TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**

Logo

Description automatically generated

**BÁO CÁO GIỮA KÌ MÔN HỌC SÂU**

*Người thực hiện:* **TRẦN MINH TRÍ - 52000815**

**LÊ VĂN VIỆT- 52000822**

**TRẦN QUỐC VINH - 52000823**

Lớp: **200503401**

Khóa: **24**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023**

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**

Logo

Description automatically generated

**BÁO CÁO GIỮA KÌ MÔN HỌC SÂU**

*Người thực hiện:* **TRẦN MINH TRÍ - 52000815**

**LÊ VĂN VIỆT- 52000822**

**TRẦN QUỐC VINH - 52000823**

Lớp: **200503401**

Khóa: **24**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2023**

LỜI CẢM ƠN

*TP. Hồ Chí Minh, ngày 31 tháng 3 năm 2024*

*Tác giả*

*Trần Minh Trí*

*Lê Văn Việt*

*Trần Quốc Vinh*

BÁO CÁO ĐƯỢC HOÀN THÀNH

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Tôi xin cam đoan đây là công trình nghiên cứu của riêng tôi. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong Khóa luận/Đồ án tốt nghiệp còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

**Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung Báo cáo cuối kỳ của mình**. Trường Đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

*TP. Hồ Chí Minh, ngày 31 tháng 3 năm 2024*

*Tác giả*

*Trần Minh Trí*

*Lê Văn Việt*

*Trần Quốc Vinh*

**TÓM TẮT**

Bài báo cáo được chia làm 3 chương. Chương 1 là phần tổng quan lý thuyết về đạo hàm. Chương 2 là thực nghiệm và nghiên cứu với các hyperparameter như learning rate, batch size và architecture. Chương 3 là xây dựng model CNN-LSTM cho bài toán Image Captioning với 2 lựa chọn: không có Attention và có Attention Mechanism

# **MỤC LỤC**

[**TÓM TẮT** v](#_Toc162784712)

[**MỤC LỤC** vi](#_Toc162784713)

[**DANH MỤC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ** viii](#_Toc162784714)

[**1** **CHƯƠNG 1 – NỀN TẢNG LÝ THUYẾT** 1](#_Toc162784715)

[1.1 Derivative 101 (Đạo hàm 101) 1](#_Toc162784716)

[*1.1.1* Tổng quát về đạo hàm 1](#_Toc162784717)

[*1.1.2* Ứng dụng 1: đạo hàm dùng để tìm giá trị lớn và nhỏ nhất 1](#_Toc162784718)

[*1.1.3* Ứng dụng 2: đạo hàm dể tính tốc độ thay đổi 1](#_Toc162784719)

[1.2 Vai trò của đạo hàm trong Machine Learning, Deep Learning 2](#_Toc162784720)

[*1.2.1* Mô hình tổng quát của bài toán dự đoán 2](#_Toc162784721)

[***1.2.2*** Các cách tính đạo hàm thông dụng 2](#_Toc162784722)

[*1.3* Automatic Differentiation trong Pytorch 6](#_Toc162784723)

[1.3.1 Vấn đề gặp phải khi đạo hàm Loss Function 6](#_Toc162784724)

[1.3.2 Thư viện Autograd 6](#_Toc162784725)

[*1.4* Forward mode và Reverse mode trong đạo hàm 7](#_Toc162784726)

[1.4.1 Forward mode (đạo hàm thuận/tiến) 7](#_Toc162784727)

[1.4.2 Reverse mode (đạo hàm ngược) 9](#_Toc162784728)

[***1.5*** Vai trò của Computational Graph (đồ thị tính toán) trong Automatic Differentiation và Backpropagation 10](#_Toc162784729)

[**2** **CHƯƠNG 2 – THỰC NGHIỆM VÀ NGHIÊN CỨU** 12](#_Toc162784730)

[**3** **CHƯƠNG 3 – XÂY DỰNG MODEL CNN-LSTM CHO BÀI TOÁN IMAGE CAPTIONING** 13](#_Toc162784731)

[**BẢNG PHÂN CHIA CÔNG VIỆC** 14](#_Toc162784732)

[**TÀI LIỆU THAM KHẢO** 15](#_Toc162784733)

**DANH MỤC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ**

[Hình 1 1 Đạo hàm biểu tượng 3](#_Toc162784541)

[Hình 1 2 Điểm yếu của đạo hàm biểu tượng 4](#_Toc162784542)

[Hình 1 3 Đạo hàm số học 4](#_Toc162784543)

[Hình 1 4 Điểm yếu của đạo hàm số học 5](#_Toc162784544)

[Hình 1 5 Đạo hàm tự đồng bằng Pytorch 6](#_Toc162784545)

[Hình 1 6 Dùng Autograd để tối ưu hàm Loss 7](#_Toc162784546)

[Hình 1 7 ví dụ về Forward Mode 8](#_Toc162784547)

[Hình 1 8 Lập danh sách các biến trung gian trong Forward Mode 8](#_Toc162784548)

[Hình 1 9 Tính đạo hàm bằng Forward Mode 9](#_Toc162784549)

[Hình 1 10 Ví dụ về Reverse Mode 10](#_Toc162784550)

[Hình 1 11 Lập các biến trung gian trong Reverse Mode 11](#_Toc162784551)

[Hình 1 12 Đạo hàm theo quy tắc chuỗi để lan truyền ngược 11](#_Toc162784552)

[Hình 1 13 Tính kết quả đạo hàm bằng Reverse Mode 12](#_Toc162784553)

[Hình 2 1 Thực nghiệm và nghiên cứu về Batch size, Learning Rate, Architecture 13](#_Toc162784554)

[Hình 3. 1 Cấu trúc folder của model CNN-LSTM 14](#_Toc162784575)

# **CHƯƠNG 1 – NỀN TẢNG LÝ THUYẾT**

## Derivative 101 (Đạo hàm 101)

### Tổng quát về đạo hàm

* Đạo hàm là một khái niệm quan trọng trong Toán học, đặc biệt trong Giải tích. Nó thể hiện tốc độ thay đổi của một hàm số tại một điểm cho trước. Nói cách khác, đạo hàm cho biết mức độ "dốc" của đồ thị hàm số tại điểm đó.
* Định nghĩa theo wiki: **"Đạo hàm của một hàm số là một đại lượng mô tả sự biến thiên của hàm tại một điểm nào đó"**.
* Tuy nhiên ta có thể nói lại theo cách đơn giản hơn: **"Đạo hàm cho biết tốc độ thay đổi (Rate of change) của hàm số tại 1 điểm cụ thể"**
* Ví dụ như sau: f(x) = 0.01x^2^ + 5 (tr VNĐ) mô tả mức lương của nhân viên dựa vào số áo bán được trong tháng. Trong đó f(x) là tiền lương nhân viên và x là số áo bán được. Nhân viên càng bán được nhiều áo thì tiền thưởng càng tăng theo hàm bậc hai. Nếu tính đạo hàm của hàm trên sẽ ra được f'(x)= 0.02x. Tức là nếu nhân viên bán được 2 áo thì mức lương là 5,04 triệu VNĐ và 0.02\*2=0.04 (tr VNĐ) cho biết mức tăng lương cho mỗi áo bán thêm tại thời điểm bán được 2 áo<br>

**=> Khi nào có sự thay đổi, ta có thể xem được hàm số đó sẽ thay đổi như thế nào bằng đạo hàm (thay đổi bao nhiêu, tốc độ thay đổi)**

### Ứng dụng 1: đạo hàm dùng để tìm giá trị lớn và nhỏ nhất

* Đạo hàm giúp xác định điểm cực đại, cực tiểu của hàm số, từ đó ta tìm được giá trị lớn nhất (GTLN) và giá trị nhỏ nhất (GTNN) của hàm số trên một khoảng xác định
* Phương pháp:
  + Tìm đạo hàm của hàm số.
  + Giải phương trình f'(x) = 0 để tìm các - điểm cực trị.
  + Xét dấu đạo hàm f'(x) trên các khoảng do các điểm cực trị chia ra.
  + So sánh giá trị hàm số tại các điểm cực trị và hai đầu khoảng xác định để tìm GTLN và GTNN.

=> Thường được dùng cho bài toán tối ưu (Optimization problem) trong nhiều lĩnh vực như logistics, thương mại điện tử

### Ứng dụng 2: đạo hàm dể tính tốc độ thay đổi

* Đạo hàm có tính chất: Nếu hàm số đang tăng đạo hàm sẽ dương, tăng càng nhanh thì đạo hàm càng lớn. Ngược lại, hàm số đang giảm, đạo hàm sẽ âm và âm càng nhiều khi hàm số giảm càng nhanh. Nhờ tính chất trên ta có thể
  + Xác định tốc độ tức thời: Ví dụ, biết được vận tốc của một vật đang chuyển động tại một thời điểm cụ thể.
  + Dự đoán xu hướng biến đổi: Dựa vào tốc độ thay đổi hiện tại, dự đoán giá trị của hàm số trong tương lai
* => Thường được dùng cho bài toán dự đoán trong lĩnh vực AI (Machine Learning, Deep Learning)

## Vai trò của đạo hàm trong Machine Learning, Deep Learning

### Mô hình tổng quát của bài toán dự đoán

Học máy (Machine Learning) là một lĩnh vực trí tuệ nhân tạo, cho phép máy tính học hỏi từ dữ liệu và đưa ra dự đoán hoặc quyết định. Để giải quyết một vấn đề bằng học máy, chúng ta thường mô hình hóa như sau

**1. Xác định mô hình**

* Ký hiệu f(θ) biểu diễn cho một mô hình cụ thể
* Biến số **θ** (theta) tượng trưng cho các tham số của mô hình
* Ví dụ mô hình dự đoán mức lương dựa vào 2 tham số là θ1 (số năm kinh nghiệm) và θ2 (giới tính): **f(x) = θ1.x1+ θ2.x2 + bias**

=> Mô hình này sẽ học hỏi từ dữ liệu và sử dụng các tham số để thực hiện dự đoán hoặc phân tích.

**2. Xác định hàm mất mát (Loss function)**

* Hàm mất mát, ký hiệu L, giúp đo lường mức độ sai lệch giữa dự đoán của mô hình và dữ liệu thực tế.
* Giá trị của hàm mất mát càng thấp thì mô hình càng chính xác. Một số hàm mất mát thông dụng là: MSE, MBE, Binary Cross Entropy

**3. Tối ưu hóa hàm mất mát**

* Mục tiêu là tìm ra bộ tham số θ (theta) tối ưu giúp giảm thiểu hàm mất mát L
* Để đạt được điều này, chúng ta cần **tính toán độ dốc (gradient) của hàm mất mát** theo từng tham số θ
* Ký hiệu ∇θL (nabla theta L) biểu diễn cho độ dốc của hàm mất mát L theo từng tham số θ. **Một số thuật toán để tối ưu loss function bao gồm: Gradient Descent, Adagrad, Adam**

=> Một model trong một bài toán machine learning sẽ bao gồm các parameters (chúng còn được gọi là “weights” hoặc "thetas"), với mục đích cố gắng giải thích bộ data có sẵn theo một công thức nào đó. Một hàm mất mát (loss function) sẽ cho chúng ta biết model của chúng ta tốt hay kém. Loss function có output thường có output là một số thực và các parameters nói trên là independent variables.

### Các cách tính đạo hàm thông dụng

Có 3 cách tính đạo hàm thường thấy là:

1. Symbolic Differentiation (đạo hàm biểu tượng)
2. Numeric differentiation (đạo hàm số)
3. Automatic Differentiation (đạo hàm tự động)

**Bảng so sánh ưu và nhược điểm của các cách tính đạo hàm**

|  |  |  |
| --- | --- | --- |
| **Phương pháp** | **Ưu điểm** | **Nhược điểm** |
| Phân biệt đạo hàm biểu tượng | - Kết quả chính xác.  - Có thể xử lý các hàm phức tạp.  - Hữu ích cho phân tích lý thuyết. | - Có thể tốn nhiều chi phí tính toán.  - Có thể không đơn giản hóa biểu thức kết quả. |
| Phân biệt đạo hàm số | - Đơn giản để thực hiện.  - Có thể sử dụng khi không thể thực hiện phân biệt đạo hàm biểu tượng. | - Dễ xảy ra lỗi làm tròn.  - Độ chính xác phụ thuộc vào bước (h). |
| Phân biệt đạo hàm tự động | - Tính toán hiệu quả đạo hàm chính xác.  - Dễ sử dụng và tích hợp với mã PyTorch.  - Hỗ trợ nhiều loại tensor và phép toán. | - Yêu cầu thư viện hoặc khuôn khổ chuyên biệt.  - Có thể không đơn giản hóa mã đạo hàm kết quả. |

=> Chúng ta sẽ khám phá cách tiếp cận cũng như ưu và nhược điểm của từng loại đạo hàm

#### Symbolic Differentiation (đạo hàm biểu tượng)

* Khái niệm: Thao tác trực tiếp trên biểu thức toán học để tìm đạo hàm. Hoạt động dựa trên bản thân biểu thức, không phải giá trị số.
* Đầu vào/Đầu ra: Biểu thức toán học (hàm) là đầu vào, kết quả là biểu thức mới biểu diễn đạo hàm.
* Ví dụ: Sử dụng thư viện Symbol trong Python để thực hiện Symbolic Differentiation  
  A screenshot of a computer

  Description automatically generated

Hình 1 1 Đạo hàm biểu tượng

* Vấn đề của Symbolic Differentiation là khi function của ta trở nên phức tạp, các ký hiệu (term) có thể tăng rất nhanh dẫn đến khó quan sát và sai sót. Ta thấy ở bảng dưới, khi áp dụng product rule thì kết quả đạo hàm sẽ rất dài  
  A white paper with black text and numbers

  Description automatically generated

Hình 1 2 Điểm yếu của đạo hàm biểu tượng

#### Numeric Differentiation (đạo hàm số)

* Khái niệm: Xấp xỉ đạo hàm bằng cách sử dụng các điểm khác biệt hữu hạn hay còn gọi là **finite/forward difference method**. Tính toán độ dốc của đường tiếp tuyến với đường cong của hàm tại một điểm.
* Đầu vào/Đầu ra: Giá trị số của hàm tại các điểm khác nhau. Đầu ra là xấp xỉ đạo hàm tại một điểm cụ thể.
* Ví dụ: tìm đạo hàm số bằng cách thế 1 điểm cụ thể trong Python

A screenshot of a computer code

Description automatically generated

Hình 1 3 Đạo hàm số học

* Đối với Numeric Differentiation thì ta không còn gặp vấn đề giống Symbolic Differentiation là các ký hiệu quá nhiều do ta có thể thay số và xấp xỉ được gradient. Tuy nhiên ta gặp phải một vài vấn đề với phương pháp này
  + Chi phí phép tính cao: Mỗi một tham số đều yêu cầu phép tính nên ta có độ phức tạp là O(n).
  + Ngoài ra khi h (h ở đây là thay đổi nhỏ nhất ta dùng để tìm gradient) quá lớn sẽ dẫn tới Truncation error (ví dụ gradient chính xác là 1 nhưng do h quá lớn nên khi tính chỉ đc 0.96 dẫn tới sai số là 0.04)
  + Tuy vậy khi h quá nhỏ thì lại dẫn tới Round off error(Lỗi làm tròn số)

A graph with a line going up

Description automatically generated

Hình 1 4 Điểm yếu của đạo hàm số học

#### Automatic Differentiation (đạo hàm tự động)

* Khái niệm: Tính toán hiệu quả đạo hàm chính xác của mã xác định một hàm. Hoạt động bằng cách lưu trữ các phép tính trung gian trong quá trình truyền trực tiếp và sử dụng chúng trong quá trình truyền ngược để tính toán đạo hàm.
* Đầu vào/Đầu ra: Mã xác định một hàm. Đầu ra là mã tính toán đạo hàm của hàm đó.
* Ví dụ: tìm đạo hàm bằng Pytorch trong Python  
  A screenshot of a computer code

  Description automatically generated

Hình 1 5 Đạo hàm tự đồng bằng Pytorch

## Automatic Differentiation trong Pytorch

### Vấn đề gặp phải khi đạo hàm Loss Function

* Khi train một model, chúng ta sẽ thay đổi các parameters sao cho model chúng ta fit bộ data đó càng khớp càng tốt, i.e. loss function càng nhỏ càng tốt.
* Nếu loss function là hàm đạo hàm được, **với một giá trị cụ thể của bộ parameters, ta có thể tính được xem loss function tại giá trị đó đang có xu hướng tăng hay giảm và tốc độ thay đổi là bao nhiêu**.
* Đối với một loss function đơn giản, ta có thể tính tay công thức gradient của nó và khi cần tính gradient tại điểm nào thì thay điểm đó vào công thức gradient là ra. Tuy nhiên nếu loss function phức tạp và nhiều tham số trong model, việc tính toán trở nên cực kỳ khó khăn và thường rất dễ sai
* Ví dụ: Ở thuật toán gradient descent, ta cần tính đạo hàm của loss function (L) với các tham số của model. Ở mô hình neural network, sẽ tính đạo hàm L với các tham số qua thuật toán backpropagation.

### Thư viện Autograd

* Gói thư viện Autograd giải quyết vấn đề này một cách nhanh chóng và hiệu quả bằng cách tự động hoá các phép tính đạo hàm (automatic differentiation). Nó giúp ta tính toán nhanh đạo hàm mà **không gặp phải các vấn để của Symbolic và Numerical Differentiation**
* Trong khi nhiều thư viện yêu cầu ta phải biên dịch một đồ thị biểu tượng (symbolic graph) để có thể tự động tính đạo hàm, autograd cho phép ta tính đạo hàm ngay lập tức thông qua các dòng lệnh thông thường.
* Mỗi khi đưa dữ liệu chạy qua mô hình, autograd xây dựng một đồ thị và theo dõi xem dữ liệu nào kết hợp với các phép tính nào để tạo ra kết quả. Với đồ thị này autograd sau đó có thể lan truyền ngược gradient lại theo ý muốn

Note: Lan truyền ngược ở đây chỉ đơn thuần là truy ngược lại đồ thị tính toán và điền vào đó các giá trị đạo hàm riêng theo từng tham số

* Ví dụ Autograd để tối ưu parameters cho hàm loss  
  A screenshot of a computer program

  Description automatically generated

Hình 1 6 Dùng Autograd để tối ưu hàm Loss

## Forward mode và Reverse mode trong đạo hàm

### Forward mode (đạo hàm thuận/tiến)

* Forward mode differentiation trong automatic differentiation là kỹ thuật tính đạo hàm bằng cách vừa đánh giá hàm (evaluate function) và lưu trữ các giá trị đó vào biến trung gian (intermediary variables) trong lúc tính toán và để cập nhật giá trị đạo hàm theo từng bước nhỏ.
* Forward mode differentiation dùng trong trường hợp size của output parameter nhiều hơn size của input parameter (size của đầu ra nhiều hơn đầu vào, ví dụ truyền 3 tham số nhưng đầu ra là 10 tham số) và **có độ phức tạp là O(n) trong đó n chính là số tham số đầu vào**

**=> Forward mode không thích hợp để tìm đạo hàm trong bài toán Machine Learning do tính chất này**

Ví dụ nếu muốn tìm đạo hàm theo x của hàm phía dưới sử dụng Forward mode differentiation với 2 tham số x,y tại

A black background with white text

Description automatically generated

A black background with white text

Description automatically generated

Hình 1 7 ví dụ về Forward Mode

1. Lập bảng các biến trung gian (intermediate values)  
   A black background with white text

   Description automatically generated

Hình 1 8 Lập danh sách các biến trung gian trong Forward Mode

1. Tìm partial derivative (đạo hàm từng phần) theo từng biến. Nếu tìm theo x thì cho x=1 (dx/dx=1) và các biến khác = 0 (ví dụ: dy/dx=0 do y và x là 2 biến độc lập), thực hiện tương tự như vậy với các tham số khác như y,...

* Tìm đạo hàm của hàm trên theo biến x sử dụng Forward mode. Ở mỗi bước nhỏ vậy ta lưu lại các giá trị trung gian kèm với tangents (tiếp tuyến) của nó, như vậy trong 1 lần pass duy nhất ta sẽ kiếm đc cả f(x,y) và f'(x,y) with respect to x  
  A black background with white text

  Description automatically generated

Hình 1 9 Tính đạo hàm bằng Forward Mode

**=> Kết quả đạo hàm của hàm trên theo x tại điểm (1.5, 0.5) là: 3,012**

### Reverse mode (đạo hàm ngược)

* Do sự hạn chế của Forward mode thì Reverse mode đã ra đời để khắc phục nhược điểm trên.
* Một cách tổng quát không phụ thuộc vào ứng dụng thì tên của nó là “phép vi phân ngược” (reverse-mode differentiation). Mặc lan truyền ngược được sử dụng cho học sâu, nhưng nó còn là công cụ tính toán mạnh mẽ cho nhiều lĩnh vực khác từ dự báo thời tiết tới phân tích tính ổn định số học, chỉ có điều là nó được sử dụng với những tên khác nhau. Thực ra nó được khai phá lại để sử dụng cho rất nhiều lĩnh vực khác nhau.
* Lan truyền ngược (backpropagation) là giải thuật cốt lõi giúp cho các mô hình học sâu có thể dễ dàng thực thi tính toán được. Với các mạng NN hiện đại, nhờ giải thuật này mà thuật toán tối ưu với đạo hàm (gradient descent) có thể nhanh hơn hàng triệu lần so với cách thực hiện truyền thống. Cứ tưởng tượng 1 mô hình với lan truyền ngược chạy mất 1 tuần thì có thể mất tới 200,000 năm để huấn luyện với phương pháp truyền thống!
* Khác với lối tiếp cận là bottom-up/from inside to outside của đạo hàm thuận, đạo hàm ngược tiếp cận bằng phương pháp top-down sử dụng chain rule cho phép ta tính được gradient của tất cả tham số trong 1 lần pass

A diagram of mathematical equations

Description automatically generated

Hình 1 10 Ví dụ về Reverse Mode

**=> Phù hợp với các task liên quan tới ML và DL**. Cách tính chi tiết bạn có thể xem ở bài viết này: https://jingnanshi.com/blog/autodiff.html#orgf5cbf62

## Vai trò của Computational Graph (đồ thị tính toán) trong Automatic Differentiation và Backpropagation

Đồ thị tính toán đóng vai trò quan trọng trong cả Automatic Differentiation và backpropagation, nó là nền tảng trong việc huấn luyện các mô hình học máy, đặc biệt là mạng nơ-ron.

Sử dụng Computational Graph để tính đạo hàm: mỗi bước đại diện cho một phép toán. Sơ đồ này được hình ảnh hóa thành dạng đồ thị với các node là các phép toán (cộng, trừ, nhân, chia, log, sin, cos,..) được kết nối bởi các edge là luồng dữ liệu đi



1. Xây dựng đồ thị bằng Forward mode: Trong quá trình forward pass, các phép tính trong đồ thị được thực hiện giống Forward mode ta đã nói phía trên  
   A diagram of a mathematical equation

   Description automatically generated

Hình 1 11 Lập các biến trung gian trong Reverse Mode

1. Tính đạo hàm bằng Reverse mode: Sử dụng chain rule để truyền ngược về và tính đạo hàm theo các tham số  
   A diagram of a mathematical equation

   Description automatically generated

Hình 1 12 Đạo hàm theo quy tắc chuỗi để lan truyền ngược

1. Tính kết quả đạo hàm của Y theo từng tham số  
   A mathematical equations on a white background

   Description automatically generated

Hình 1 13 Tính kết quả đạo hàm bằng Reverse Mode

# **CHƯƠNG 2 – THỰC NGHIỆM VÀ NGHIÊN CỨU**

* Chi tiết thực nghiệm và nghiên cứu xem thêm trong file **SimpleNNUsingPytorchAutograd.ipynb** đính kèm

A computer screen shot of a program

Description automatically generated

Hình 2 1 Thực nghiệm và nghiên cứu về Batch size, Learning Rate, Architecture

# **CHƯƠNG 3 – XÂY DỰNG MODEL CNN-LSTM CHO BÀI TOÁN IMAGE CAPTIONING**

Folder chứa model CNN-LSTM cho bài toán Image Captioning bao gồm các file dưới đây, vô từng file để xem chi tiết hướng dẫn

Link Github: [tmt203/image-captioning (github.com)](https://github.com/tmt203/image-captioning?fbclid=IwAR2u6PG-skhkETq8mqBHTfvg1aT4Tq1c7g0bKNQDGRlXCXve0iBjRn_3qLY_aem_AWegxwbQqZBjOscv568UfEhl-_rJ0s6imirFtByb6pGRnF8Z89yiU6MnAJ53wFkGM-_crMQu22qjcodriAZc6X0a)

* **image\_captioning.ipynb**: project overview, train model, load model for testing. (After training, we will use encoder-3.pkl and decoder-3.pkl in models/ to predict on test dataset)
* **data\_loader.py**: load Coco dataset (use to load captions and images data in opt/cocoapi/annotations and opt/cocoapi/images).
* **model.py**: implements CNN-LSTM model.
* **vocabulary.py**: build the vocab for the training process. (vocab.pkl)

A screenshot of a computer program

Description automatically generated

Hình 3. 1 Cấu trúc folder của model CNN-LSTM

**BẢNG PHÂN CHIA CÔNG VIỆC**

|  |  |  |  |
| --- | --- | --- | --- |
| **Số thứ tự** | **Full name** | **Student ID** | **Role** |
| 1 | Trần Quốc Vinh | 52000823 | Nền tảng lý thuyết và viết report |
| 2 | Lê Văn Việt | 52000822 | Thực nghiệm và nghiên cứu |
| 3 | Trần Minh Trí | 52000815 | Xây dựng model CNN-LSTM cho bài toán Image Captioning với 2 lựa chọn: không có attention và có attention |

**TÀI LIỆU THAM KHẢO**

1. https://nttuan8.com/bai-2-autograd/
2. https://www.tvhoang.com/articles/2019/07/autograd
3. https://jingnanshi.com/blog/autodiff.html
4. https://dominhhai.github.io/vi/2018/04/nn-bp/
5. <https://d2l.aivivn.com/chapter_multilayer-perceptrons/backprop_vn.html>
6. [Khoa học dữ liệu (phamdinhkhanh.github.io)](https://phamdinhkhanh.github.io/2019/08/10/PytorchTurtorial1.html)
7. https://github.com/OLeoghain/image\_caption-lstm/tree/master