**TRƯỜNG ĐẠI HỌC PHENIKAA**

**KHOA KHOA HỌC CƠ BẢN**

⸎⸎⸎⸎⸎

**Logo, company name

Description automatically generated**

BÀI GIỮA KỲ

**Deep Learning**

Đề tài: Car-Truck Classiffication

(Phân loại xe)

Lớp: Học-sâu-1-2-22(N01)

|  |  |
| --- | --- |
| Giảng viên hướng dẫn | : **TS. Lê Minh Huy** |
| Sinh viên thực hiện- MSV | : Nguyễn Quốc Khánh – 21011495  Đỗ Trọng Hiệp – 21011492 |
|  |  |
|  |  |

Hà Nội, tháng 3 năm 2023

# **MỞ ĐẦU**

AI (Artificial Intelligence) bao gồm nhiều lĩnh vực nghiên cứu, từ thuật toán di truyền đến các hệ thống chuyên gia và cung cấp phạm vi cho các lập luận về những gì cấu thành AI.

Trong lĩnh vực nghiên cứu AI, Machine Learning đã đạt được thành công đáng kể trong những năm gần đây – cho phép máy tính vượt qua hoặc tiến gần đến việc kết hợp hiệu suất của con người trong các lĩnh vực từ nhận dạng khuôn mặt đến nhận dạng giọng nói và ngôn ngữ.

Deep Learning (hay còn gọi là Học sâu) là một tập hợp con của Machine Learning, có khả năng khác biệt ở một số khía cạnh quan trọng so với Machine Learning nông truyền thống, cho phép máy tính giải quyết một loạt các vấn đề phức tạp không thể giải quyết được.

Dưới đây là triển khai đơn giản của một mô hình học sâu, dưa trên những gì đã học được, chúng em triện khai một mạng phân loại xe bằng một mạng Deep Neural Network(github[[1]](#footnote-1)) gồm 7 layers được triển khai từ numpy và math.

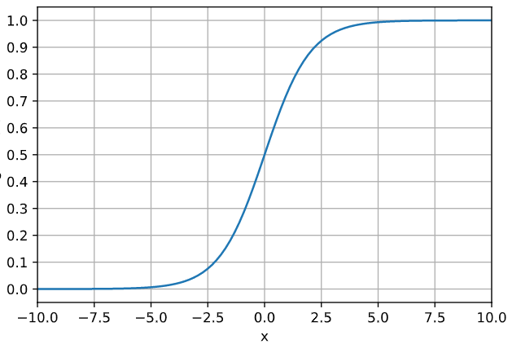
# **NỘI DUNG**

## **Phần I: Thu thập, xử lý, chọn dữ liệu, lựa chọn mô hình**

Mô hình phân loại của chúng em sử dụng là mô hình phân loại nhị phân (Binary Classification). Sẽ phân biệt 2 loại xe: Xe con và Xe tải. Tập dữ liệu sẽ chia thành 3 phần *Train/Validation/Test.*

Mô hình được xây dựng huấn luyện hai bộ dataset : KaggleCarTruck & CarTruck. Trong đó, Kaggle\_Car\_Truck là bộ dữ liệu được tải từ Kaggle tổng gồm 1267 ảnh chia theo tỷ lệ 78/16/8(%) và CarTruck là bộ dữ liệu với các ảnh được tải xuống từ trên google. Qua quá trình xử lý và làm sạch dữ liệu, chúng em thu được bộ dữ liệu hoàn chỉnh với 4400 ảnh trên 82/9/9(%).

Với việc chọn bài toán phân loại nhị phân. Chúng em chọn mô hình Deep Neural Network với 7 layers fully connected triển khai bằng các thư viện numpy, math, matplotlib,... Do các ảnh có kích thước lớn, không đồng nhất. Nên trước khi đưa vào huấn luyện chúng sẽ được giảm chiều xuống thành (64,64,3), và chuẩn hóa về các giá trị trong khoảng từ -1 đến 1. Dữ liệu sẽ được đưa vào mô hình. VÌ đây là bài toán phân loại nhị phân nên output sẽ được đi qua hàm *Sigmoid* để đưa về khoảng giá trị [0; 1]. Ngưỡng được lấy là 0.5 tức là những ảnh có output >= 0.5 sẽ được dự đoán là 1 (Car) và các ảnh có output < 0.5 được dự đoán là 0 (Truck).



Hình 1. Activation function Sigmoid

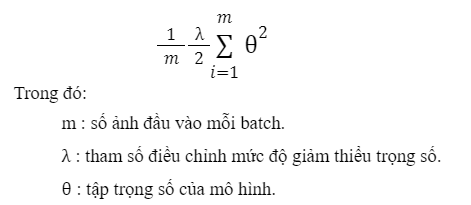
Optimizer chúng em lựa chọn cho mô hình là Adam bởi những ưu điểm sau:

* Adam kết hợp các ưu điểm của optimizer RMSProp và Momentum, giúp tăng tốc độ hội tụ và giảm hiện tượng rơi vào cục bộ tối ưu hóa.
* Tính ổn định: Adam có thể giảm thiểu ảnh hưởng của các giá trị gradient nhiễu và đảm bảo sự ổn định trong quá trình tối ưu hóa.
* Tốc độ hội tụ nhanh: Vì Adam tự động điều chỉnh tỷ lệ học (learning rate) trong quá trình tối ưu hóa, nó có thể đạt tốc độ hội tụ nhanh hơn so với các phương pháp tối ưu hóa khác.

Khi huấn luyện một mô hình học sâu với bộ data hạn chế, một hiện tượng dễ dàng gặp phải đó chính là Overfitting. Khi đó, mô hình học các quy tắc và mối quan hệ bên trong trong tập huấn luyện, mà không thể tổng quát hóa cho các dữ liệu mới dẫn đến việc dự đoán không còn khách quan.

Để ngăn chặn hiện tượng Overfitting, mô hình của chúng em được áp dụng các phương pháp Regularization, Drop Out.

Regularization cố gắng giảm thiểu Overfitting bằng cách thêm một số hàm mất mát vào quá trình huấn luyện. Các hàm mất mát này giúp kiểm soát độ phức tạp của mô hình và giúp các tham số trọng số của mô hình không trở nên quá lớn. Kỹ thuật Regularization được bọn em sử dụng là L2 regularization có công thức:

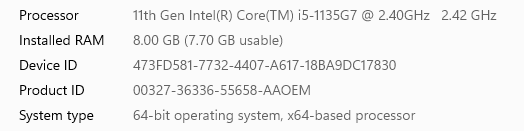


Dropout là một kỹ thuật khác được sử dụng để giảm thiểu hiện tượng Overfitting. Khi áp dụng Dropout, một số unit (node) trong mạng neural sẽ được chọn ngẫu nhiên và loại bỏ trong quá trình huấn luyện. Kết quả là mạng neural sẽ không dựa quá nhiều vào một số unit cụ thể để thực hiện quá trình học, mà sẽ học được các đặc trưng của dữ liệu đầu vào. Kỹ thuật này chỉ áp dụng trong quá trình training mà không áp dụng trong lúc đánh giá mô hình bằng tập validation hay tập test để đảm bảo tính khách quan.

  Trong phần code, biến list *keep\_probs* chứa tỉ lệ unit (node) được giữ lại ở mỗi layer trừ layer cuối. Thường thì layer cuối cùng trong mạng neural được thiết kế để tính toán đầu ra cụ thể cho bài toán. Vì vậy, việc áp dụng dropout trên layer cuối cùng có thể dẫn đến mất mát thông tin quan trọng và làm giảm chất lượng của đầu ra.

## **Phần II: Huấn luyện mô hình**

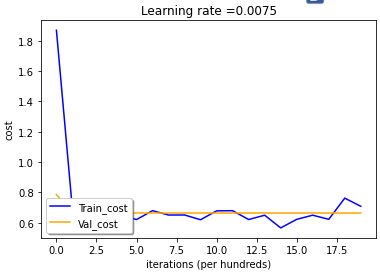
Trước hết, chúng em huấn luyện mô hình với KaggleCarTruck dataset. Với 200 epochs (iterations). Lúc này mô hình Deep Neural mới chỉ có 5 layes được huấn luyện trên máy Laptop cá nhân với các thông số:



Kết quả thu được với loss = 0.66470. Cùng với đó là dự đoán cũng không được tốt khi giá trị dự đoán *y\_pred* luôn ra kết quả bằng 1 (Hình 2).

A, 

Hình 2. A, Mô hình dự đoán với label = “Car” trên Kaggle dataset

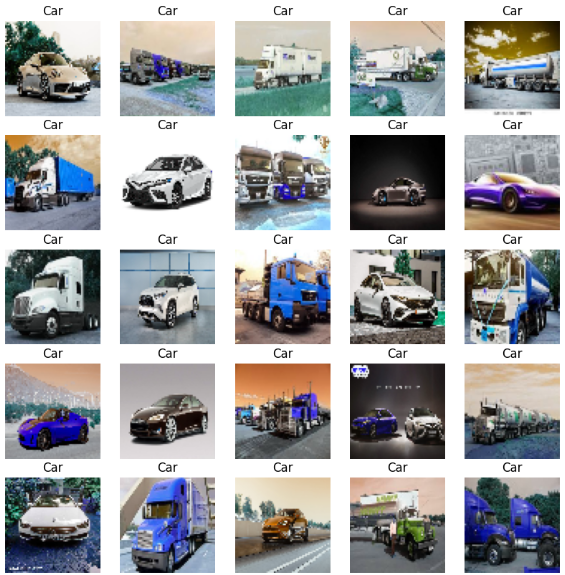
B,

Hình 2. B, Biểu đồ loss sau 100 epochs

Có 3 nguyên nhân có thể dẫn đến hiện tượng dự đoán duy nhất một class như trên:

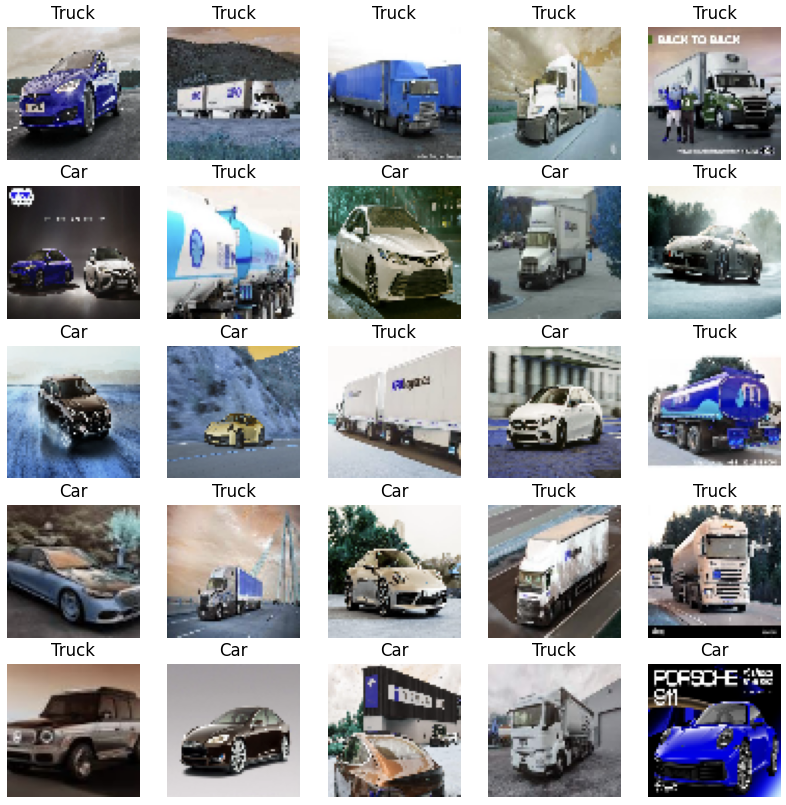
* Tập dữ liệu quá ít và bị lệch quá nhiều về bên *Car* khiến cho mô hình trở nên thiên vị.
* Chất lượng ảnh không tốt, từ góc chụp, độ nét đến tính đặc trưng đều không đồng đều.
* Mô hình không sử dụng các lớp ConvNet nên không thể trích xuất những đặc trưng khó của ảnh, do đó yêu cầu ảnh đầu vào phải có những đặc trưng dễ dàng phân biệt hơn.

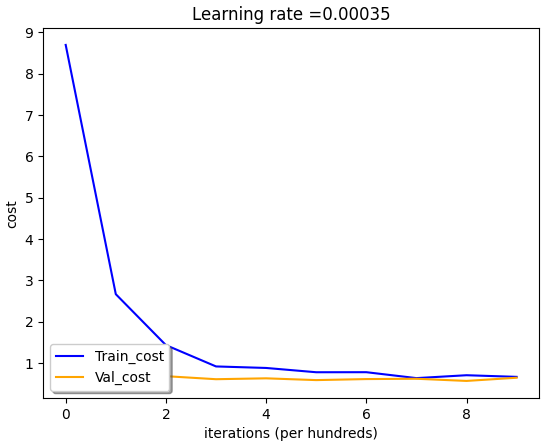
Để giải quyết nguyên nhân đầu tiên, chúng em thu thập thêm dữ liệu và tạo ra bộ dữ mới CarTruck với số ảnh *Truck* và ảnh *Car* ngang nhau (2200 ảnh mỗi loại). Mô hình huấn luyện với 100 epochs. Tuy nhiên kết quả vẫn không tốt, *y\_pred* vẫn chỉ ra kết quả bằng 1 (Hình 3).  Do đó, chúng em quyết định điều chỉnh các siêu tham số (hyperparameters) để giúp mô hình hiệu quả hơn.



Hình 3. Mô hình dự đoán với label = “Car” trên CarTruck dataset

Sau hàng loạt thử nghiệm với việc thay đổi số layer, số node mỗi layer, *learning\_rate*, *lambda*, *keep\_probs*,... mô hình cuối cùng là mô hình Deep Neural Network với 7 layers (như đã nói). Với các siêu tham số như *learning\_rate* = 0.00035, *lambda* = 0.45, *mini\_batch\_size* = 128 mô hình thu được *loss* = 0.641405 ở epoch 100 (Hình 4.B). Accuracy = 79%. Đây là một kết quả có thể chấp nhận được với một model đơn giản và bộ data tự thu thập trong thời gian ngắn.

A, 

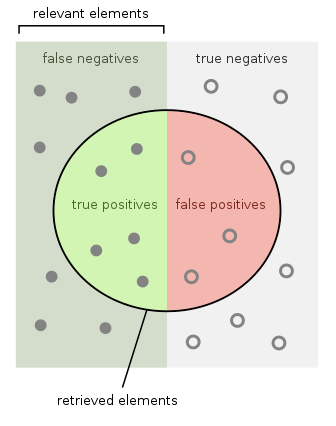
B, 

Hình 4. A, Mô hình dự đoán trên CarTruck dataset

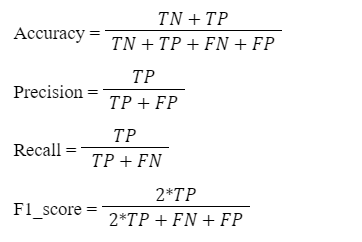
B, Biểu đồ loss sau 100 epochs

## **Phần III: Đánh giá mô hình**

Để đánh giá mô hình, chúng em sử dụng 4 độ đo: Accuray, Precision, Recall, F1\_Score với công thức dựa trên hình dưới đây.

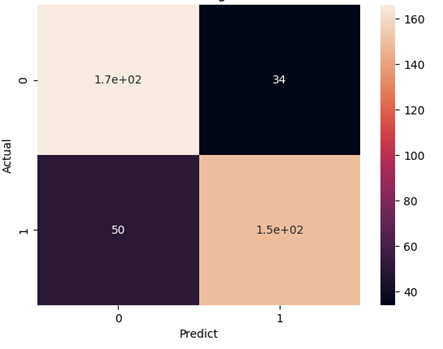


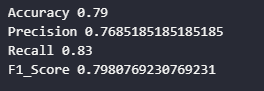
Ta có:



Trong đó TP, TN, FP, FN lần lượt là True Positive, True Negative, False Positive, False Negative.

Để thuận tiện cho việc tính toán, bọn em sử dụng pandas để vẽ Confusion matrix. Từ đó thu được các kết quả ở hình 4. Trong đó, Positive = 0, Negative = 1.





Hình 5. Cofusion Matrix cùng kết quả của Accuray, precision, recall, f1score

# **KẾT LUẬN**

Trong quá trình thực hiện bài tập, chúng em đã phần nào hiểu được bản chất của một mạng Deep Nuera, từ đó thực hiện các phương pháp học sâu để cải thiện tốc độ huấn luyện tăng độ chính xác.Cùng lúc đó, đã giải quyết được Overffiting – một vấn đề vẫn thường thấy trong một mô hình học máy.

Chúng em đã học được cách thu thập và tiền xử lý dữ liệu, nhận ra tầm quan trọng của khâu thu thập dữ liệu. Một bộ dữ liệu không tốt có thể khiến người xây dựng mô hình mất rất nhiều thời gian và công sức nhưng không thể tăng hiệu quả của mô hình lên được.

Những kinh nghiệm rút ra được là vô cùng quý giá để chúng em có thể phát triển hơn trong bài tập lớn cuối kỳ cũng như trong sự nghiệp sau này.

# **TÀI LIỆU THAM KHẢO**

1. [Precision and recall - Wikipedia](https://en.wikipedia.org/wiki/Precision_and_recall)
2. [Deep Learning Coursera](https://l.facebook.com/l.php?u=https%3A%2F%2Fwww.coursera.org%2Fprograms%2Ffpt-university-jan-may-2023-yfdrt%3FcollectionId%3D%26currentTab%3DMY_COURSES%26productId%3DW62RsyrdEeeFQQqyuQaohA%26productType%3Ds12n%26showMiniModal%3Dtrue&h=AT0lQDDzPtTSeTxMaQbA-a8ufD_dOGHSeuppoGXmhSal-9qnMRSfN9DJ7aeVDkVT4KkUtvmwfsYNhDmnlZQtaGyTQgKvP6D5LTpIakDJ5xpfUbjw3ATNZJHDWk-7yeYQ8IS1n9wwYUWLdDs&s=1)

**BẢNG PHÂN CÔNG VIỆC**

|  |  |  |
| --- | --- | --- |
| Họ và Tên | MSV | Phân chia công việc |
| Nguyễn Quốc Khánh | 21011495 | Thu thập, xử lý dữ liệu  Viết code:  Forward Propagation  Chia Mini-batch size  Adam Optimization  Evaluation  Chọn tham số fit mô hình  Viết report, làm slide, thuyết trình. |
| Đỗ Trọng Hiệp | 21011492 | Thu thập, xử lý dữ liệu  Viết code load data  Backward Propagation  Dropout  Regularization  Hàm Train  Chọn tham số fit mô hình  Viết report, làm slide, thuyết trình. |

1. [qkhanh711/DeepLearning\_University (github.com)](https://github.com/qkhanh711/DeepLearning_University) [↑](#footnote-ref-1)