BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC NHA TRANG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÀI TẬP LỚN**

**PLANT CLASSIFICATION**

**GVHD: ThS. NGUYỄN HẢI TRIỀU**

**SVTH: Phan Nguyễn Đình Vũ**

**MSSV: 61131562**

Khánh Hòa, năm 2024

BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC NHA TRANG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÀI TẬP LỚN**

**PLANT CLASSIFICATION**

**GVHD: ThS. NGUYỄN HẢI TRIỀU**

**SVTH: Phan Nguyễn Đình Vũ**

**MSSV: 61131562**

Khánh Hòa, năm 2024

|  |  |
| --- | --- |
| **TRƯỜNG ĐẠI HỌC NHA TRANG**  KHOA CÔNG NGHỆ THÔNG TIN |  |

**PHIẾU THEO DÕI TIẾN ĐỘ VÀ ĐÁNH GIÁ**

**ĐỒ ÁN/KHÓA LUẬN/CHUYÊN ĐỀ TỐT NGHIỆP**

**(Dùng cho CBHD và nộp cùng báo cáo ĐA/KL/CĐTN của sinh viên)**

**Tên đề tài**: PLANT CLASSIFICATION

Giảng viên hướng dẫn: Ths. Nguyễn Hải Triều

Sinh viên được hướng dẫn: Phan Nguyễn Đình Vũ MSSV: 61131562

Khóa: 61 Ngành: Công nghệ thông tin

|  |  |  |  |
| --- | --- | --- | --- |
| **Lần KT** | **Ngày** | **Nội dung** | **Nhận xét của GVHD** |
| 1 |  | Tổng quan Keras, CNN | Đạt |
| 2 |  | Nghiên cứu RNN, kết hợp 2 models CRNN | Đạt |
| 3 |  | Deploy ứng dụng | Đạt |
| 4 |  | Báo cáo, sửa bài đồ án theo yêu cầu GV | Đạt |
| **Kiểm tra giữa tiến độ của trưởng Bộ môn** | | | |
| Ngày kiểm tra:  ……………………. | | Đánh giá công việc hoàn thành: …..%  Được tiếp tục:  Không tiếp tục:  | Ký tên ………………………. |
|  | |  |  |
|  | |  |  |
|  | |  |  |

Nhận xét chung (sau khi sinh viên hoàn thành ĐA/KL/CĐTN): sinh viên Phan Nguyễn Đình Vũ có tinh thần trách nhiệm, hoàn thành tốt các nội dung được giao. Kết quả thu được thỏa mãn yêu cầu của đề tài.

Điểm hình thức: /10 Điểm nội dung: /10 Điểm tổng kết: /10

+ Đối với ĐA/KLTN:

Kết luận sinh viên: Được bảo vệ: 🗹 Không được bảo vệ: 

Khánh Hòa, ngày…….tháng 06 năm 2024

Cán bộ hướng dẫn

(Ký và ghi rõ họ tên)

Nguyễn Hải Triều

**TRƯỜNG ĐẠI HỌC NHA TRANG**

KHOA: CÔNG NGHỆ THÔNG TIN

**PHIẾU CHẤM ĐIỂM CHUYÊN ĐỀ TỐT NGHIỆP**

**(Dành cho cán bộ chấm phản biện)**

1. **Họ tên người chấm**:
2. **Sinh viên/ nhóm sinh viên thực hiện CDTN** (sĩ số trong nhóm):

(1)…………………………………………………..MSSV.............................................  
(2)…………………………………………………..MSSV.............................................  
(3)…………………………………………………..MSSV.............................................  
(4)…………………………………………………..MSSV.............................................  
Lớp:………………………..Ngành:..................................................................................

1. **Tên đề tài:**
2. **Nhận xét**

- Hình thức:

- Nội dung:

Điểm hình thức:……/10 Điểm nội dung:......./10 **Điểm tổng kết**:………/10

*Khánh Hòa, ngày…….tháng 06 năm 2024*  
**Cán bộ chấm phản biện***(Ký và ghi rõ họ tên)*

1. **LỜI CAM ĐOAN**

Tôi xin cam đoan đây là công trình nghiên cứu của tôi dưới sự hướng dẫn của ThS. Nguyễn Hải Triều.

***Nha Trang, tháng 06 năm 2024***

Sinh viên

Phan Nguyễn Đình Vũ

# **LỜI CẢM ƠN**

Em xin gửi lời cảm ơn chân thành đến Quý thầy (cô) của khoa Công nghệ thông tin, đề tài này đem lại nhiều trải nghiệm và giúp em hiểu rõ về cơ chế của hệ thống phân loại tự động. Em được học nhiều kỹ thuật, kinh nghiệm mới trong quá trình thực hiện đề tài này.

Em cũng muốn bày tỏ lòng cảm kích đến giảng viên hướng dẫn thầy Nguyễn Hải Triều, thầy đã tận tình hướng dẫn những lúc khó khăn, đóng góp những ý kiến trong quá trình thực hiện đề tài, giúp em định hướng được đề tài “***Plant Classification***”, Nhờ có sự giúp đỡ của thầy, em đã hoàn thành đề tài này một cách hoàn thiện nhất.

Em xin chân thành cảm ơn.

Sinh viên

Phan Nguyễn Đình Vũ

**MỤC LỤC**

[LỜI CAM ĐOAN i](#_Toc137723933)

[LỜI CẢM ƠN ii](#_Toc137723934)

[DANH MỤC CÁC TỪ VIẾT TẮT v](#_Toc137723935)

[DANH MỤC HÌNH ẢNH vi](#_Toc137723936)

[PHẦN MỞ ĐẦU viii](#_Toc137723937)

[Chương 1: TỔNG QUAN VỀ ĐỀ TÀI NGHIÊN CỨU 9](#_Toc137723938)

[**1.1** **LÝ DO CHỌN ĐỀ TÀI** 9](#_Toc137723939)

[**1.2** **MỤC TIÊU ĐỀ TÀI** 9](#_Toc137723940)

[Chương 2: CƠ SỞ LÝ THUYẾT 10](#_Toc137723941)

[**2.1** **TỔNG QUAN VỀ CAPTCHA** 10](#_Toc137723942)

[**2.1.1** **Captcha là gì?** 10](#_Toc137723943)

[**2.1.2** **Đặc điểm** 10](#_Toc137723944)

[**2.1.3** **Nguồn gốc** 11](#_Toc137723945)

[**2.1.4** **Nguyên tắc hoạt động** 11](#_Toc137723946)

[**2.1.5** **Ứng dụng** 12](#_Toc137723947)

[**2.2** **NGÔN NGỮ LẬP TRÌNH PYTHON** 13](#_Toc137723948)

[**2.2.1** **Giới thiệu** 13](#_Toc137723949)

[**2.2.2** **Ưu điểm** 13](#_Toc137723950)

[**2.2.3** **Nhược điểm** 13](#_Toc137723951)

[**2.2.4** **Cài đặt** 14](#_Toc137723952)

[**2.3** **THƯ VIỆN GRADIO** 14](#_Toc137723953)

[**2.3.1** **Giới thiệu** 14](#_Toc137723954)

[**2.3.2** **Ưu điểm** 15](#_Toc137723955)

[**2.3.3** **Nhược điểm** 15](#_Toc137723956)

[**2.4** **TỔNG QUAN VỀ THƯ VIỆN KERAS** 15](#_Toc137723957)

[**2.4.1** **Keras là gì?** 15](#_Toc137723958)

[**2.4.2** **Thế mạnh** 16](#_Toc137723959)

[**2.4.3** **Hạn chế** 16](#_Toc137723960)

[**2.4.4** **Cài đặt** 17](#_Toc137723961)

[**2.4.5** **Ứng dụng** 17](#_Toc137723962)

[**2.4.6** **Ví dụ về bài toán phân loại hình ảnh** 18](#_Toc137723963)

[**2.5** **TỔNG QUAN VỀ CNN** 25](#_Toc137723964)

[**2.5.1** **CNN là gì?** 25](#_Toc137723965)

[**2.5.2** **Những lớp cơ bản** 25](#_Toc137723966)

[**2.5.3** **Cấu trúc của CNN** 28](#_Toc137723967)

[**2.5.4** **Ưu và nhược điểm của CNN** 29](#_Toc137723968)

[**2.5.5** **Ứng dụng** 30](#_Toc137723969)

[**2.5.6** **Ví dụ về bài toán cho CNN** 31](#_Toc137723970)

[**2.6** **TỔNG QUAN VỀ RNN** 35](#_Toc137723971)

[**2.6.1** **RNN là gì?** 35](#_Toc137723972)

[**2.6.2** **Ví dụ và ứng dụng** 36](#_Toc137723973)

[**2.6.3** **Ưu và nhược điểm** 39](#_Toc137723974)

[Chương 3 XÂY DỰNG BÀI TOÁN GIẢI CAPTCHA TỰ ĐỘNG 40](#_Toc137723975)

[**3.1** **Dữ liệu đầu vào** 40](#_Toc137723976)

[**3.2** **Mục tiêu của bài toán** 40](#_Toc137723977)

[**3.3** **Import thư viện** 40](#_Toc137723978)

[**3.4** **Tải bộ dữ liệu** 41](#_Toc137723979)

[**3.5** **Tiền xử lý dữ liệu** 42](#_Toc137723980)

[**3.6** **Khởi tạo đối tượng trong tập dữ liệu** 44](#_Toc137723981)

[**3.7** **Hiển thị dữ liệu** 45](#_Toc137723982)

[**3.8** **Khởi tạo model C-RNN** 46](#_Toc137723983)

[**3.9** **Huấn luyện mô hình** 53](#_Toc137723984)

[**3.10** **Dự đoán** 56](#_Toc137723985)

[**3.11** **Giao diện chính** 62](#_Toc137723986)

[Chương 4: KẾT LUẬN 64](#_Toc137723987)

[**4.1 ĐÁNH GIÁ ĐỀ TÀI** 64](#_Toc137723988)

[**4.1.1 Kết quả** 64](#_Toc137723989)

[**4.1.2 Khuyết điểm** 64](#_Toc137723990)

[**4.2 HƯỚNG HOÀN THIỆN** 64](#_Toc137723991)

[TÀI LIỆU THAM KHẢO 65](#_Toc137723992)

1. DANH MỤC CÁC TỪ VIẾT TẮT

|  |  |
| --- | --- |
| **Ký Hiệu** | **Thuật ngữ** |
| API | Application Programming Interface |
| CAPTCHA | Completely Automated Public Turing test to tell Computers and Humans Apart |
| CNN | Convolutional Neural Network |
| CTC | Connectionist Temporal Classification |
| CNTK | Microsoft Cognitive Toolkit |
| CPU | Central Processing Unit |
| DDoS | Distributed Denial-of-Service |
| GUI | Graphical User Interface |
| GPU | Graphics Processing Unit |
| GAN | Generative Adversarial Network |
| IDLE | Integrated Development and Learning Environment |
| LSTM | Long Short-Term Memory |
| PIP | Preferred Installer Program |
| RNN | Recurrent Neural Network |
| SGD | Stochastic Gradient Descent |

# **DANH MỤC HÌNH ẢNH**

[*Hình 2.1 Captcha là gì?* 11](file:///D:\Study\CDTN\CNDT_PhanNguyenDinhVu_61131562.docx#_Toc137726776)

[*Hình 2.2 Nguyên tắc hoạt động của Captcha* 12](file:///D:\Study\CDTN\CNDT_PhanNguyenDinhVu_61131562.docx#_Toc137726777)

[*Hình 2.3 Ứng dụng của Captcha* 13](file:///D:\Study\CDTN\CNDT_PhanNguyenDinhVu_61131562.docx#_Toc137726778)

[*Hình 2.4 Python* 14](#_Toc137726779)

[*Hình 2.5 Trang chủ Python* 15](#_Toc137726780)

[*Hình 2.6 Gradio* 15](#_Toc137726781)

[*Hình 2.7 Keras* 16](file:///D:\Study\CDTN\CNDT_PhanNguyenDinhVu_61131562.docx#_Toc137726782)

[*Hình 2.8 Câu lệnh cài đặt Keras* 18](#_Toc137726783)

[*Hình 2.9 Ứng dụng của Keras* 18](file:///D:\Study\CDTN\CNDT_PhanNguyenDinhVu_61131562.docx#_Toc137726784)

[*Hình 2.10 CIFAR-10 dataset* 19](file:///D:\Study\CDTN\CNDT_PhanNguyenDinhVu_61131562.docx#_Toc137726785)

[*Hình 2.11 Import thư viện* 20](file:///D:\Study\CDTN\CNDT_PhanNguyenDinhVu_61131562.docx#_Toc137726786)

[*Hình 2.12 Load và chia tập dữ liệu* 20](file:///D:\Study\CDTN\CNDT_PhanNguyenDinhVu_61131562.docx#_Toc137726787)

[*Hình 2.13 Kết quả (1)* 20](file:///D:\Study\CDTN\CNDT_PhanNguyenDinhVu_61131562.docx#_Toc137726788)

[*Hình 2.14 Hiển thị hình ảnh* 20](file:///D:\Study\CDTN\CNDT_PhanNguyenDinhVu_61131562.docx#_Toc137726789)

[*Hình 2.15 Hiển thị hình ảnh* 20](file:///D:\Study\CDTN\CNDT_PhanNguyenDinhVu_61131562.docx#_Toc137726790)

[*Hình 2.16 Kết quả (2)* 21](file:///D:\Study\CDTN\CNDT_PhanNguyenDinhVu_61131562.docx#_Toc137726791)

[*Hình 2.17 Chuyển đổi kiểu và chuẩn hóa* 21](file:///D:\Study\CDTN\CNDT_PhanNguyenDinhVu_61131562.docx#_Toc137726792)

[*Hình 2.18 Xử lý dữ liệu* 21](file:///D:\Study\CDTN\CNDT_PhanNguyenDinhVu_61131562.docx#_Toc137726793)

[*Hình 2.19 Xây dựng mô hình* 22](file:///D:\Study\CDTN\CNDT_PhanNguyenDinhVu_61131562.docx#_Toc137726794)

[*Hình 2.20 Tối ưu hóa* 22](file:///D:\Study\CDTN\CNDT_PhanNguyenDinhVu_61131562.docx#_Toc137726795)

[*Hình 2.21 Kết quả (3)* 23](file:///D:\Study\CDTN\CNDT_PhanNguyenDinhVu_61131562.docx#_Toc137726796)

[*Hình 2.22 Câu lệnh và kết quả (1)* 23](file:///D:\Study\CDTN\CNDT_PhanNguyenDinhVu_61131562.docx#_Toc137726797)

[*Hình 2.23 Training* 24](file:///D:\Study\CDTN\CNDT_PhanNguyenDinhVu_61131562.docx#_Toc137726798)

[*Hình 2.24 Kết quả (4)* 24](file:///D:\Study\CDTN\CNDT_PhanNguyenDinhVu_61131562.docx#_Toc137726799)

[*Hình 2.25 Câu lệnh và kết quả (2)* 24](file:///D:\Study\CDTN\CNDT_PhanNguyenDinhVu_61131562.docx#_Toc137726800)

[*Hình 2.26 Câu lệnh lưu* 24](file:///D:\Study\CDTN\CNDT_PhanNguyenDinhVu_61131562.docx#_Toc137726801)

[*Hình 2.27 Tạo Dictionary để đưa ra dự đoán* 25](file:///D:\Study\CDTN\CNDT_PhanNguyenDinhVu_61131562.docx#_Toc137726802)

[*Hình 2.28 Kết quả (5)* 25](file:///D:\Study\CDTN\CNDT_PhanNguyenDinhVu_61131562.docx#_Toc137726803)

[*Hình 2.29 Convolutional Layer* 27](file:///D:\Study\CDTN\CNDT_PhanNguyenDinhVu_61131562.docx#_Toc137726804)

[*Hình 2.30 Max Pooling và Average Pooling* 28](file:///D:\Study\CDTN\CNDT_PhanNguyenDinhVu_61131562.docx#_Toc137726805)

[*Hình 2.31 Fully Connected Layer* 29](#_Toc137726806)

[*Hình 2.32 Cấu trúc của CNN* 30](#_Toc137726807)

[*Hình 2.33 Ứng dụng của CNN* 31](file:///D:\Study\CDTN\CNDT_PhanNguyenDinhVu_61131562.docx#_Toc137726808)

[*Hình 2.34 Dữ liệu trong MNIST Dataset* 32](file:///D:\Study\CDTN\CNDT_PhanNguyenDinhVu_61131562.docx#_Toc137726809)

[*Hình 2.35 Model cho bài toán CNN* 32](file:///D:\Study\CDTN\CNDT_PhanNguyenDinhVu_61131562.docx#_Toc137726810)

[*Hình 2.36 Softmax Function* 33](file:///D:\Study\CDTN\CNDT_PhanNguyenDinhVu_61131562.docx#_Toc137726811)

[*Hình 2.37 Ví dụ cho hàm mất mát* 34](file:///D:\Study\CDTN\CNDT_PhanNguyenDinhVu_61131562.docx#_Toc137726812)

[*Hình 2.38 Các giá trị* 35](file:///D:\Study\CDTN\CNDT_PhanNguyenDinhVu_61131562.docx#_Toc137726813)

[*Hình 2.39 Biểu đồ hàm mất mát L* 35](file:///D:\Study\CDTN\CNDT_PhanNguyenDinhVu_61131562.docx#_Toc137726814)

[*Hình 2.40 Mô hình RNN* 36](file:///D:\Study\CDTN\CNDT_PhanNguyenDinhVu_61131562.docx#_Toc137726815)

[*Hình 2.41 Các mô hình của RNN* 37](#_Toc137726816)

[*Hình 2.42 Kết quả (6)* 38](file:///D:\Study\CDTN\CNDT_PhanNguyenDinhVu_61131562.docx#_Toc137726817)

[*Hình 2.43 Ứng dụng của RNN* 39](file:///D:\Study\CDTN\CNDT_PhanNguyenDinhVu_61131562.docx#_Toc137726818)

[*Hình 3.1 Input ảnh Captcha định dạng png, kích thước 200x50 pixel* 41](file:///D:\Study\CDTN\CNDT_PhanNguyenDinhVu_61131562.docx#_Toc137726819)

[*Hình 3.2 Import thư viện* 41](#_Toc137726820)

[*Hình 3.3 Tải bộ dữ liệu* 42](#_Toc137726821)

[*Hình 3.4 In thông tin tập dữ liệu* 42](#_Toc137726822)

[*Hình 3.5 Thiết lập các thông số* 43](#_Toc137726823)

[*Hình 3.6 Kết quả (1)* 43](#_Toc137726824)

[*Hình 3.7 Ánh xạ* 43](#_Toc137726825)

[*Hình 3.8 Chia dữ liệu* 44](#_Toc137726826)

[*Hình 3.9 Encode* 45](#_Toc137726827)

[*Hình 3.10 Khởi tạo đối tượng* 45](#_Toc137726828)

[*Hình 3.11 Hiện ảnh và nhãn* 46](#_Toc137726829)

[*Hình 3.12 Kết quả (2)* 47](file:///D:\Study\CDTN\CNDT_PhanNguyenDinhVu_61131562.docx#_Toc137726830)

[*Hình 3.13 Cấu trúc model* 48](#_Toc137726831)

[*Hình 3.14 Khởi tạo lớp CTC* 48](#_Toc137726832)

[*Hình 3.15 Phương thức call* 49](#_Toc137726833)

[*Hình 3.16 Xây dựng model* 49](#_Toc137726834)

[*Hình 3.17 Các khối Conv* 50](#_Toc137726835)

[*Hình 3.18 Tính toán kích thước mới* 51](#_Toc137726836)

[*Hình 3.19 RNN* 51](#_Toc137726837)

[*Hình 3.20 Lớp Dense* 52](#_Toc137726838)

[*Hình 3.21 Lớp CTC* 52](#_Toc137726839)

[*Hình 3.22 Định nghĩa và biên dịch mô hình* 52](#_Toc137726840)

[*Hình 3.23 Lấy và tóm tắt mô hình* 53](#_Toc137726841)

[*Hình 3.24 Kết quả (3)* 53](#_Toc137726842)

[*Hình 3.25 Huấn luyện mô hình* 54](#_Toc137726843)

[*Hình 3.26 Kết quả huấn luyện* 54](#_Toc137726844)

[*Hình 3.27 Kết quả (4)* 55](#_Toc137726845)

[*Hình 3.28 Trực quan hóa đồ thị* 56](#_Toc137726846)

[*Hình 3.29 Đồ thị* 56](#_Toc137726847)

[*Hình 3.30 Khởi tạo mô hình dự đoán* 57](#_Toc137726848)

[*Hình 3.31 Greedy Search* 57](#_Toc137726849)

[*Hình 3.32 Hàm duyệt và trả về văn bản* 58](#_Toc137726850)

[*Hình 3.33 Dự đoán và so sánh với nhãn thực thế* 59](#_Toc137726851)

[*Hình 3.34 Dự đoán đầu ra* 59](#_Toc137726852)

[*Hình 3.35 Dùng hàm decode để giải mã* 59](#_Toc137726853)

[*Hình 3.36 Tạo danh sách lưu trữ* 59](#_Toc137726854)

[*Hình 3.37 Vòng lặp for duyệt và giải mã thành văn bản* 60](#_Toc137726855)

[*Hình 3.38 Tạo biểu đồ* 60](#_Toc137726856)

[*Hình 3.39 Độ chính xác* 61](#_Toc137726857)

[*Hình 3.40 Kết quả (5)* 61](#_Toc137726858)

[*Hình 3.41 Kết quả (6)* 62](#_Toc137726859)

[*Hình 3.42 Giao diện chính* 62](#_Toc137726860)

[*Hình 3.43 Thao tác trên giao diện* 63](#_Toc137726861)

[*Hình 3.44 Kết quả (7)* 63](#_Toc137726862)

# **PHẦN MỞ ĐẦU**

Trong thời đại công nghệ số, việc áp dụng các phương pháp tự động hóa vào những công việc đòi hỏi nhiều thời gian và công sức đang trở nên ngày càng phổ biến. Một trong những ứng dụng đáng chú ý là việc phân loại các loài thực vật, đặc biệt là các loài hoa. Việc xác định loài hoa yêu cầu kiến thức sâu rộng của các nhà thực vật học và quá trình này có thể rất tốn thời gian cũng như chi phí.

Nhằm giải quyết thách thức này, em đã phát triển một hệ thống phân loại tự động các loài hoa dựa trên hình ảnh. Mục tiêu là xây dựng một bộ phân loại có thể tự động nhận dạng và phân loại các loài hoa từ hình ảnh, giúp giảm bớt khối lượng công việc thủ công. Việc phát triển một hệ thống phân loại tự động không chỉ giúp tiết kiệm thời gian và chi phí mà còn nâng cao độ chính xác và nhất quán trong việc phân loại các loài hoa. Điều này có ý nghĩa quan trọng không chỉ đối với các bảo tàng và vườn bách thảo mà còn đối với các lĩnh vực khác như nông nghiệp và bảo tồn thiên nhiên, nơi việc xác định chính xác các loài thực vật đóng vai trò then chốt.

Em đã sử dụng các phương pháp học máy tiên tiến kết hợp với histogram màu để xây dựng mô hình phân loại. Bằng cách này, em đã tạo ra một hệ thống có khả năng phân loại các loài hoa một cách chính xác và hiệu quả. Mô hình này không chỉ là một công cụ mạnh mẽ trong việc phân loại hoa mà còn là một ví dụ điển hình về việc áp dụng công nghệ hiện đại vào giải quyết các vấn đề truyền thống trong khoa học và đời sống.

Vì vậy em lựa chọn đề tài “**Plant Classification**”. Dùng ngôn ngữ Python, thư viện OpenCV, Scikit-learn và sử dụng thư viện PyQt để thiết kế giao diện.

Nội dung bài báo cáo gồm 4 chương:

**Chương 1: Tổng quan về đề tài nghiên cứu**

**Chương 2: Cơ sở lý thuyết**

**Chương 3: Xây dựng bài toán giải Captcha tự động**

**Chương 4: Kết luận**

# **Chương 1: TỔNG QUAN VỀ ĐỀ TÀI NGHIÊN CỨU**

* 1. **LÝ DO CHỌN ĐỀ TÀI**

Phân loại các loài hoa từ lâu đã là một công việc quan trọng trong nhiều lĩnh vực như thực vật học, nông nghiệp, và bảo tồn thiên nhiên. Tuy nhiên, việc này đòi hỏi nhiều kiến thức chuyên môn và thời gian của các nhà thực vật học. Một số lý do chính để chọn đề tài này bao gồm:

* **Tiết Kiệm Thời Gian và Chi Phí**: Việc phân loại hoa bằng phương pháp truyền thống rất tốn thời gian và chi phí. Bằng cách áp dụng các phương pháp tự động hóa, chúng ta có thể giảm đáng kể thời gian và nguồn lực cần thiết.
* **Tăng Độ Chính Xác và Nhất Quán**: Phân loại thủ công có thể dẫn đến sai sót do yếu tố con người. Sử dụng máy học giúp đảm bảo tính nhất quán và độ chính xác cao hơn.
* **Ứng Dụng Công Nghệ Mới**: Việc áp dụng công nghệ học máy và xử lý hình ảnh vào lĩnh vực thực vật học không chỉ là một bước tiến quan trọng mà còn mở ra nhiều cơ hội nghiên cứu và ứng dụng mới.
* **Hỗ Trợ Nghiên Cứu Khoa Học**: Hệ thống phân loại tự động giúp các nhà nghiên cứu tập trung vào các nghiên cứu khoa học sâu hơn, thay vì tốn thời gian vào các công việc phân loại thủ công.
  1. **MỤC TIÊU ĐỀ TÀI**

Đề tài nghiên cứu này được xây dựng với các mục tiêu cụ thể sau:

1. **Phát Triển Hệ Thống Phân Loại Tự Động**: Xây dựng một hệ thống có khả năng tự động phân loại các loài hoa từ hình ảnh một cách chính xác và hiệu quả.
2. **Ứng Dụng Histogram Màu và Học Máy**: Sử dụng histogram màu 3D RGB để đặc trưng hóa màu sắc của các cánh hoa, kết hợp với các thuật toán học máy để xây dựng mô hình phân loại.
3. **Đánh Giá Hiệu Suất Hệ Thống**: Thử nghiệm và đánh giá hiệu suất của hệ thống trên các tập dữ liệu khác nhau để đảm bảo độ chính xác cao và tính khả thi của mô hình.
4. **Tối Ưu Hóa và Mở Rộng**: Tối ưu hóa hệ thống để đạt được hiệu suất tốt nhất, đồng thời mở rộng khả năng phân loại cho nhiều loài hoa hơn và áp dụng cho các tập dữ liệu lớn hơn.

# **Chương 2: CƠ SỞ LÝ THUYẾT**

* 1. **TỔNG QUAN VỀ OPENCV**

### **Giới thiệu**



*Hình 2.1 Captcha là gì?*

OpenCV (Open Source Computer Vision Library) là một thư viện mã nguồn mở được phát triển nhằm cung cấp các công cụ và chức năng hỗ trợ cho việc xử lý và phân tích hình ảnh, video, cũng như các bài toán thị giác máy tính khác. Được phát hành lần đầu vào năm 1999 bởi Intel, OpenCV hiện nay đã trở thành một trong những thư viện phổ biến và mạnh mẽ nhất trong lĩnh vực thị giác máy tính.

### **Các tính năng chính**

OpenCV cung cấp một loạt các công cụ và thuật toán phục vụ cho nhiều mục đích khác nhau trong xử lý hình ảnh và thị giác máy tính. Một số tính năng chính của OpenCV bao gồm:

* **Xử Lý Ảnh**: Bao gồm các công cụ để đọc, ghi, hiển thị và biến đổi hình ảnh. Các thao tác xử lý ảnh phổ biến như chuyển đổi không gian màu, làm mờ, lọc ảnh, phát hiện cạnh, và tạo histogram cũng được hỗ trợ.
* **Nhận Dạng Đối Tượng**: OpenCV hỗ trợ các thuật toán nhận dạng đối tượng như Haar Cascade, HOG (Histogram of Oriented Gradients), và Deep Learning.
* **Theo Dõi Đối Tượng**: Các kỹ thuật theo dõi đối tượng trong video, bao gồm các thuật toán KLT (Kanade-Lucas-Tomasi) tracker, MeanShift, và CAMShift.
* **Học Máy**: Thư viện cung cấp các công cụ cho học máy và học sâu, bao gồm các thuật toán phân loại, hồi quy, clustering, và các mô hình mạng nơ-ron nhân tạo.
* **Xử Lý Video**: Bao gồm các công cụ để đọc và ghi video, cũng như các kỹ thuật để xử lý video theo khung hình.

### **Histogram Màu**

*Hình 2.2 Nguyên tắc hoạt động của Captcha*

Histogram màu là một công cụ quan trọng trong việc mô tả và phân loại hình ảnh dựa trên màu sắc. OpenCV cung cấp các hàm hỗ trợ tạo và xử lý histogram màu. Trong nghiên cứu của Charles, histogram màu 3D RGB được sử dụng để đặc trưng hóa màu sắc của các cánh hoa, giúp phân loại các loài hoa khác nhau. Một số bước chính để tạo histogram màu trong OpenCV bao gồm:

* **Chuyển Đổi Không Gian Màu**: Chuyển đổi hình ảnh từ không gian màu gốc sang không gian màu RGB.
* **Tạo Histogram**: Sử dụng hàm **cv2.calcHist** để tính toán histogram màu từ hình ảnh.
* **Chuẩn Hóa Histogram**: Sử dụng hàm **cv2.normalize** để chuẩn hóa giá trị histogram.

### **Học máy**

OpenCV tích hợp các công cụ học máy, cho phép xây dựng và huấn luyện các mô hình học máy để giải quyết các bài toán thị giác máy tính. Em đã sử dụng RandomForestClassifier từ thư viện scikit-learn, một thư viện học máy phổ biến, để xây dựng mô hình phân loại hoa. Các bước chính trong việc sử dụng học máy với OpenCV bao gồm:

* **Mã Hóa Nhãn Lớp**: Sử dụng **LabelEncoder** để mã hóa các nhãn lớp từ chuỗi ký tự sang số nguyên.
* **Chia Tập Dữ Liệu**: Sử dụng hàm **train\_test\_split** để chia tập dữ liệu thành tập huấn luyện và kiểm tra.
* **Huấn Luyện Mô Hình**: Sử dụng **RandomForestClassifier** để huấn luyện mô hình trên tập dữ liệu huấn luyện.
* **Đánh Giá Mô Hình**: Sử dụng **classification\_report** để đánh giá hiệu suất của mô hình trên tập dữ liệu kiểm tra.

### **Ứng dụng**

*Hình 2.3 Ứng dụng của Captcha*

OpenCV được ứng dụng rộng rãi trong nhiều lĩnh vực khác nhau, từ nghiên cứu học thuật đến các ứng dụng thực tiễn trong công nghiệp:

* **Nhận Diện Khuôn Mặt**: OpenCV hỗ trợ nhận diện và phát hiện khuôn mặt trong hình ảnh và video.
* **Thị Giác Robot**: Sử dụng trong các hệ thống robot để nhận dạng và theo dõi đối tượng.
* **Xe Tự Lái**: Hỗ trợ các hệ thống nhận dạng và phân tích hình ảnh để điều khiển xe tự lái.
* **Y Tế**: Phân tích hình ảnh y khoa để hỗ trợ chẩn đoán bệnh.
* **Nông Nghiệp Thông Minh**: Theo dõi và phân loại cây trồng, kiểm tra chất lượng nông sản.

## **NGÔN NGỮ LẬP TRÌNH PYTHON**

### **Giới thiệu**



*Hình 2.4 Python*

Python là ngôn ngữ lập trình bậc cao được ra mắt vào năm 1991 do Guido van Rossum tạo ra. Python được tạo ra cho nhiều mục đích lập trình đa năng với các ưu điểm là dễ đọc, dễ nhớ, dễ học.

Sau phiên bản đầu tiên, phiên bản Python 2 đã được ra mắt vào tháng 10 năm 2000. Và sau 8 năm, phiên bản Python 3 đã được cập nhật vào tháng 12 năm 2008. Và cho đến nay, phiên bản hiện tại là Python 3.11.2.

### **Ưu điểm**

* **Không có cú pháp phức tạp:** Khá dễ đọc và dễ học nên người dùng có thể chú tâm vào xây dựng logic.
* **Miễn phí:** Có thể tải xuống cả mã nguồn của Python, thay đổi mã nguồn và thậm chí là phân phối mã nguồn đã thay đổi.
* **Hệ sinh thái thư viện đa dạng và phong phú:** Dùng cho các mục đích khác nhau, được sử dụng phổ biến.
* **Tích hợp nhiều ngôn ngữ:** Trong khi phát triển dự án, người dùng có thể tích hợp thêm nhiều ngôn ngữ khác qua các thư viện như Cython hay Jython,…

### **Nhược điểm**

* **Rất đơn giản:** Nếu đã quá quen sử dụng nó thì sẽ khá khó để tiếp cận với ngôn ngữ khác.
* **Khó phát hiện lỗi:** Các lỗi chỉ xuất hiện khi chạy chương trình, việc này khiến việc kiểm tra và sửa lỗi trở nên khó khăn.
* **Bảo mật kém:** Python không được coi là ngôn ngữ được đánh giá cao về mặt an toàn và có một số nguy cơ liên quan đến bảo mật. Vì thế nên có một số hạn chế về việc truy cập cơ sở dữ liệu.
* **Tiêu tốn nhiều dung lượng:** Để lưu trữ cấu trúc, với tốc độ thực thi chậm hơn so với một số ngôn ngữ khác và khả năng tiêu tốn nhiều dung lượng, có thể không phù hợp cho việc phát triển những ứng dụng di động và trò chơi yêu cầu hiệu năng cao.

### **Cài đặt**

Ở Windows có rất nhiều phương pháp để cài đặt Python như cài đặt trực tiếp hay thông qua các IDLE. Tuy nhiên, cách đơn giản nhất là tải gói cài đặt từ trang chủ của Python.



*Hình 2.5 Trang chủ Python*

## **THƯ VIỆN GRADIO**

### **2.3.1 Giới thiệu**



*Hình 2.6 Gradio*

PyQt là một bộ công cụ (framework) phát triển ứng dụng giao diện đồ họa (GUI) mạnh mẽ và linh hoạt cho Python, dựa trên thư viện Qt của C++. Được phát triển và duy trì bởi Riverbank Computing, PyQt cung cấp các lớp và phương thức giúp lập trình viên dễ dàng tạo ra các ứng dụng GUI chuyên nghiệp và phức tạp. PyQt hỗ trợ nhiều phiên bản của Qt, bao gồm cả Qt 4 và Qt 5, và cung cấp các tính năng cần thiết để xây dựng các ứng dụng desktop mạnh mẽ.

### **Ưu điểm**

* **Thiết lập nhanh chóng, dễ dàng:** Có thể cài đặt Gradio bằng PIP. Tạo giao diện Gradio chỉ yêu cầu thêm một vài dòng mã vào dự án. Ta có thể chọn từ nhiều loại giao diện khác nhau cho chức năng của mình.
* **Trình bày và chia sẻ:** Gradio có thể được nhúng trong Python hoặc được trình bày dưới dạng trang web. Giao diện Gradio có thể tự động tạo liên kết công khai mà có thể chia sẻ với người khác để cho phép họ tương tác với mô hình trên máy tính từ xa từ thiết bị của chính họ.
* **Lưu trữ vĩnh viễn:** Khi đã tạo giao diện, ta có thể lưu trữ vĩnh viễn giao diện đó trên Hugging Face. Hugging Face Spaces sẽ lưu trữ giao diện trên các máy chủ của nó và cung cấp một liên kết có thể chia sẻ.

### **2.3.3 Nhược điểm**

* **Hạn chế tuỳ chỉnh:** Gradio có một số hạn chế về tuỳ chỉnh giao diện, không cho phép tùy chỉnh chi tiết hoặc phức tạp hơn.
* **Phụ thuộc vào trình duyệt:** Gradio yêu cầu người dùng có trình duyệt web để tương tác với giao diện người dùng, giới hạn sự tiện ích trên các nền tảng không có trình duyệt.
  1. **TỔNG QUAN VỀ THƯ VIỆN KERAS**

### Keras là gì ? giới thiệu về keras **Keras là gì?**

*Hình 2.7 Keras*

Keras là thư viện mã nguồn mở để tạo ra các mô hình mạng Neural và Deep Learning, phát triển bởi François Chollet và phát hành lần đầu vào năm 2015.

Giao diện thân thiện để tạo ra các mô hình mạng Neural, để tương thích với thư viện NumPy để hỗ trợ, xử lý các kiến trúc mạng khác như: mạng Neural tách lớp, mạng hồi quy, và mạng tích chập.

Keras là một trong các thư viện phổ biến cho Deep Learning đang được phổ biến rộng trong phân loại hình, nhận dạng âm thanh, và xử lý ngôn từ.

### **Thế mạnh**

* **Dễ sử dụng:** Các API của Keras rất trực quan và dễ hiểu, dễ dàng tạo ra các mô hình Deep Learning mà không cần có nhiều kiến thức chuyên môn.
* **Tích hợp linh hoạt:** tích hợp với nhiều thư viện khác như TensorFlow, Theano, CNTK,…
* **Tính toán nhanh:** Keras dùng nhiều thư viện tính toán số học tối ưu để giảm thời gian tính toán và tiết kiệm thời gian huấn luyện.
* **Đa dạng trong xây dựng mô hình:** Keras hỗ trợ nhiều kiểu kiến trúc mô hình Deep Learning như: CNN, RNN và GAN.
* **Linh hoạt trong việc tùy biến mô hình:** Keras cho phép tùy biến các siêu tham số và thông số để hợp với nhiều loại bài toán.

### **Hạn chế**

* **Giới hạn trong việc tùy chỉnh mô hình:** Việc tùy chỉnh các thành phần của mô hình như hàm mất mát hoặc tối ưu hóa có thể bị giới hạn. Keras không là lựa chọn thích hợp cho những bài toán Deep Learning phức tạp đòi hỏi nhiều tùy chỉnh.
* **Không thể chạy trên CPU thấp:** Đòi hỏi một hệ thống có GPU để có được tốc độ tính toán nhanh của GPU. Do đó, nếu sử dụng Keras trên một hệ thống chỉ có CPU, thời gian huấn luyện sẽ rất chậm.
* **Có thể gặp vấn đề với phiên bản:** Luôn được phát triển và liên tục cập nhật các phiên bản, nên việc sử dụng phiên bản cũ có thể gây ra một số lỗi trong việc huấn luyện mô hình. Do đó, người dùng cần đảm bảo sử dụng phiên bản phù hợp và cập nhật thường xuyên để tránh gặp vấn đề này.
* **Khó khăn trong việc Debug:** Vì Keras là một thư viện rất cao cấp và tổng hợp nhiều thành phần, nên việc Debug khi gặp lỗi có thể sẽ gây cản trở và tiêu tốn nhiều thời gian.

### **Cài đặt**

* **Cài đặt thư viện TensorFlow:** Keras được tích hợp với các thư viện học máy phổ biến như TensorFlow, ta cần cài đặt thư viện này trước khi cài đặt Keras. Ta có thể cài đặt TensorFlow bằng pip, công cụ quản lý gói Python.
* **Cài đặt Keras:** Ta có thể cài đặt Keras bằng pip:

**A blue background with white text

Description automatically generated with low confidence**

*Hình 2.8 Câu lệnh cài đặt Keras*

* **Kiểm tra cài đặt:** Kiểm tra xem Keras đã được cài đặt bằng lệnh sau:

**from tensorflow import keras**

Nếu không có xuất hiện lỗi, thì Keras đã được cài đặt thành công.

### **Ứng dụng**

Áp dụng X-ray để chẩn đoán ung thư ở bệnh nhân. Ngoài ra, có thể phân loại và phát hiện những ký tự và số viết tay để đọc văn bản.

*Hình 2.9 Ứng dụng của Keras*

### **Ví dụ về bài toán phân loại hình ảnh**

#### **Giới thiệu**

*Hình 2.10 CIFAR-10 dataset*

Là bài toán phân loại CIFAR-10 đã có sẵn.

Bài toán này yêu cầu chúng ta áp dụng mô hình Deep Learning để phân loại hình thuộc vào một trong 10 lớp khác nhau: chim, ô tô, chó, ếch, mèo,… CIFAR-10 có 60,000 ảnh huấn luyện và 10,000 ảnh kiểm tra với kích thước 32x32 pixel.

Để xử lý bài toán này, ta dùng Keras để xây dựng một mạng tích chập. Mạng này có một lớp input để chứa ảnh đầu vào, sau đó là một số lớp như kích hoạt và tích chập để trích xuất các thuộc tính của ảnh, và một lớp đầu ra với hàm kích hoạt để phân loại các hình đầu vào một trong 10 lớp.

Một số thông số và siêu tham số cần được thiết lập như số lớp, số lượng, kích thước bộ lọc, hàm kích hoạt, hàm mất mát, tốc độ học, và số lần lặp lại huấn luyện.

Sau khi xây dựng và thiết lập những tham số, chúng ta huấn luyện bằng cách áp dụng bộ dữ liệu đã huấn luyện và kiểm tra để điều chỉnh các trọng số.

Cuối cùng, chúng ta có thể đánh giá hiệu năng trên bộ dữ liệu kiểm tra để đánh giá độ chính xác trong việc phân loại các hình thuộc vào những lớp của nó.

#### **2.4.6.2 Xây dựng Model**

* Text

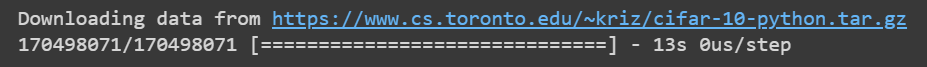
  Description automatically generatedImport các thư viện cần thiết.

*Hình 2.11 Import thư viện*

* Text

  Description automatically generatedLoad và chia tập dữ liệu từ Keras.

*Hình 2.12 Load và chia tập dữ liệu*

Kết quả nhận được:

*Hình 2.13 Kết quả (1)*

* Text

  Description automatically generatedHiển thị một số hình ảnh từ tập dữ liệu để cho ta thấy một cách trực quan.

*Hình 2.14 Hiển thị hình ảnh*

*Hình 2.0.15 Hiển thị hình ảnh*

Graphical user interface, application, website

Description automatically generatedKết quả nhận được:

*Hình 2.16 Kết quả (2)*

* Text

  Description automatically generated with low confidenceChuyển đổi các giá trị pixel thành kiểu float và chuẩn hóa tập dữ liệu.

*Hình 2.17 Chuyển đổi kiểu và chuẩn hóa*

* Text

  Description automatically generatedDùng One-Hot Encoding để xử lý dữ liệu.

*Hình 2.18 Xử lý dữ liệu*

* Text

  Description automatically generatedTạo Model Sequential để thêm vào các layer.

*Hình 2.19 Xây dựng mô hình*

Giải thích:

- Khởi tạo model Sequential.

- Để tạo các lớp tích chập, ta dùng Conv2D để trích xuất những đặc điểm từ hình với các tham số sau:

+ Filters: có 32 số lượng bộ lọc, các bộ lọc có kích thước 3x3, input có kích thước (32, 32, 3) là chiều rộng, chiều cao và số kênh màu.

+ Kernel\_constraint: sử dụng ràng buộc với 3 là trọng số.

+ Sử dụng hàm kích hoạt ReLU.

+ Padding: = "same" để output có cùng kích thước với input.

- Các lớp gộp (Pooling Layers) để giảm kích thước của input bằng cách chọn giá trị lớn nhất trong các vùng 2x2, mà vẫn giữ được các đặc điểm:

+ Pool\_size: là kích thước được sử dụng để chọn giá trị lớn nhất hoặc trung bình.

- Lớp Dense() tạo một lớp với các tham số sau:

+ Activation: lựa chọn hàm kích hoạt softmax để đưa ra xác suất dự đoán cho mọi lớp và sử dụng hàm ReLu để ràng buộc.

- Lớp Flatten() để làm phẳng chuyển đổi dữ liệu input từ tensor 2 chiều sang vector 1 chiều.

* Text

  Description automatically generatedCấu hình tối ưu hóa và biên dịch mô hình.

*Hình 2.20 Tối ưu hóa*

- Hàm compile() sử dụng thuật toán tối ưu SGD:

+ Learning\_rate: là tỉ lệ học cho quá trình cập nhật trọng số.

+ Momentum: chỉ định mức độ được dùng trong quá trình cập nhật.

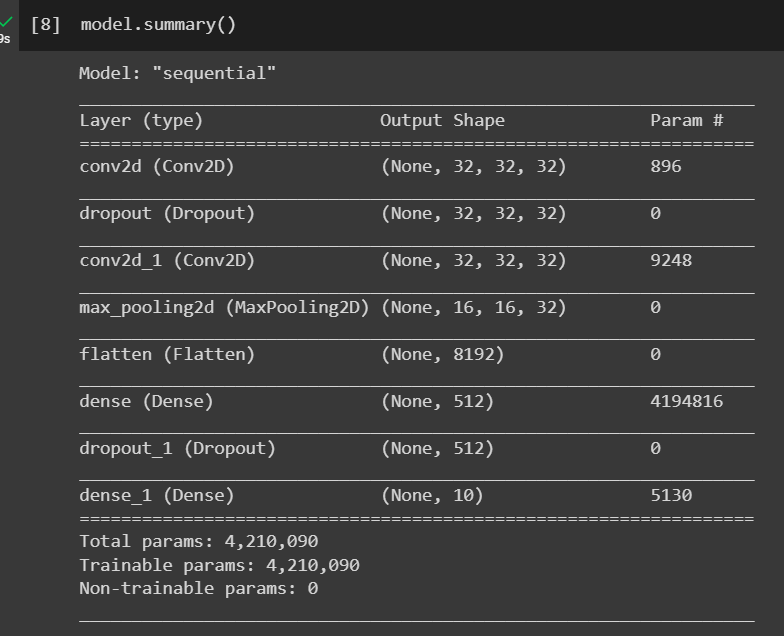
+ Decay: mức giảm tỉ lệ học theo thời gian để điều chỉnh việc học.

+ Loss: hàm mất mát thường dùng trong bài toán phân loại đa lớp.

+ Metric: đo độ chính xác.

Kết quả:

*Hình 2.21 Kết quả (3)*

* Xem tóm tắt để rõ hơn về kiến trúc của mô hình.

*Hình 2.22 Câu lệnh và kết quả (1)*

* Training model.

*Hình 2.23 Training*

- Hàm fit ():

+ Tập train và test được cho vào trong quá trình training.

+ Batch\_size là kích thước của các tập dữ liệu và sẽ được cập nhật sau mỗi batch.

+ Epoch là số lượng lần duyệt qua toàn bộ dữ liệu trong lúc huấn luyện.

Text

Description automatically generatedKết quả:

*Hình 2.24 Kết quả (4)*

* Graphical user interface, text

  Description automatically generatedĐo hiệu năng của tập test.

*Hình 2.25 Câu lệnh và kết quả (2)*

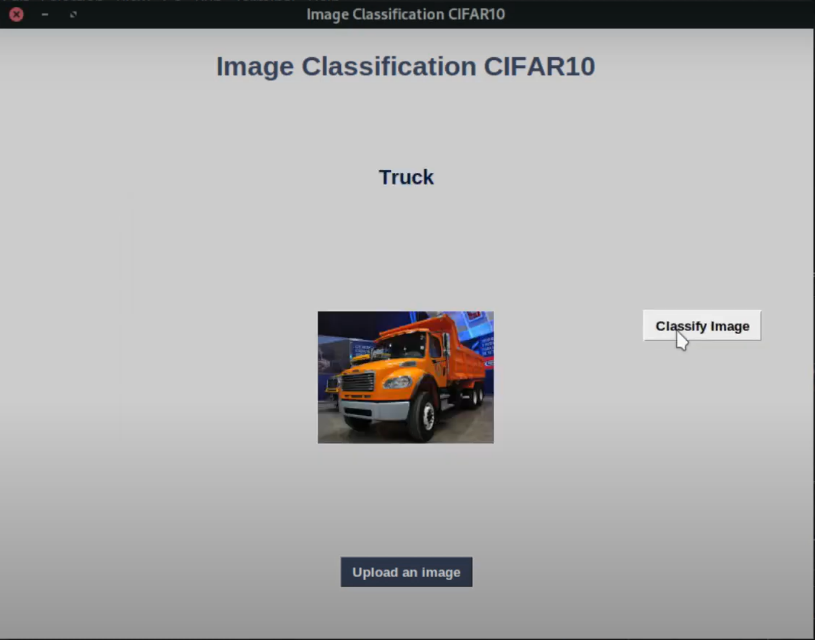
* Lưu lại mô hình.

*Hình 2.26 Câu lệnh lưu*

* Text

  Description automatically generatedTạo Dictionary để ánh xạ đến các output để dự đoán, để tăng độ chính xác ta có thể tăng số epoch lên để train.

*Hình 2.27 Tạo Dictionary để đưa ra dự đoán*

Kết quả đạt được:

*Hình 2.28 Kết quả (5)*

## 

## **2.5 TỔNG QUAN VỀ CNN**

### **2.5.1 CNN là gì?**

CNN tức mạng Neural tích chập. Là một kiến trúc mạng Neural được áp dụng rộng rãi trong xử lý ảnh và video, là một mạng Neural được thiết kế để học cách trích xuất đặc trưng từ các hình ảnh một cách tự động. Kiến trúc này có thể phân loại hình ảnh, định vị, nhận dạng vật thể, và thậm chí còn có thể tạo ảnh mới. CNN sử dụng phép tích chập để tìm kiếm những đặc điểm của hình để dự đoán.

### **Những lớp cơ bản**

**1. Convolution Layer:** là thành phần quan trọng, vì nó làm tất cả các tính toán. Những yếu tố quan trọng gồm: Filter Map, Stride, Feature Map và Padding.

- CNN dùng bộ lọc để sử dụng lên các ma trận của hình. Được biểu thị ở dạng ma trận 3D và chứa những tham số được gọi là Parameters.

- Stride là khoảng cách giữa các vị trí mà Filter Map được áp dụng trên input, được tính bằng số pixel.

- Padding là việc thêm những giá trị 0 vào input để đảm bảo rằng kích thước của output không đổi sau khi sử dụng filter map. Thông thường, các giá trị 0 được thêm vào những khu vực xung quanh của ma trận để có thể thực hiện phép tính mà không thay đổi đến kích thước ma trận của hình.

- Feature Map là kết quả của việc tính toán sau khi Filter Map sử dụng input. Feature Map thể hiện những đặc điểm đã trích xuất từ hình. Sau khi Filter Map quét qua input, Convolutional Layer sẽ tiến hành tính toán và tạo ra Feature Map tương ứng.

Cho input của một lớp tích chập có kích thước H \* W \* D.

Kernel có kích thước F \* F \* D, Kernel có độ sâu bằng độ sâu input, F là số lẻ, bước nhảy là S, đệm là P, lớp tích chập sử dụng K Kernel.

Text

Description automatically generated with low confidence=> Output là một Tensor 3D có kích thước:

Diagram

Description automatically generated

*Hình 2.29 Convolutional Layer*

**2. ReLU Layer**

Dùng để mô phỏng việc truyền tín hiệu qua các neural trong não. Nó được coi là hàm kích hoạt phổ biến và thông dụng nhất, mặc dù cũng có các hàm khác như Maxout, Sigmoid,...

Hàm ReLU được sử dụng để huấn luyện mạng Neural với ưu điểm chính là giúp tính toán nhanh. Khi sử dụng, cần chú ý thay đổi Learning Rate và Dead Unit. Sau khi áp dụng Filter Map đã tính toán và dùng ReLU lên các giá trị của bộ lọc.

**3. Pooling Layer**

Khi kích thước input ảnh lớn, lớp gộp sẽ đặt vào giữa các lớp tích chập để làm giảm số lượng tham số.

Lớp Pooling thường áp dụng giữa những lớp tích chập để làm giảm kích thước dữ liệu mà vẫn giữ lại những đặc điểm cần thiết. Việc giảm kích thước giúp giảm đi số lượng tính toán.

Kích thước của Pooling được gọi là K\*K. Input lớp gộp có kích thước H\*W\*D, chia thành D ma trận có kích thước H\*W. Ở từng ma trận, ở K\*K trên ma trận, thì ta phải tìm giá trung bình hoặc lớn nhất và ghi vào kết quả. Quy định về bước nhảy và đệm được dùng giống phép tính tích chập trên hình.

Khi dùng lớp gộp, kích thước thường là (2,2), bước nhảy là 2 và không có đệm, chiều cao và chiều rộng của đầu ra sẽ giảm ½ và độ sâu không thay đổi.

Diagram

Description automatically generatedA picture containing text, clock

Description automatically generatedHiện nay, Max Pooling và Average Pooling là 2 loại thông dụng.

*Hình 2.30 Max Pooling và Average Pooling*

*Hình 2.30 Pooling Layer (2\*2)*

1. **Fully Connected Layer**

Khi Convolutional và Pooling Layer đã có input ảnh, nó sẽ tính toán và ra kết quả. Khi mô hình đã đọc thông tin trong hình ảnh, thì sẽ thêm ra các liên kết để đưa ra nhiều đầu ra hơn. Đó là lúc ta dùng Fully Connected Layer. Nếu các Fully Connected Layer này có những dữ liệu của hình, thì nó được đổi sang một mục chưa được phân loại.

Sau khi ảnh được xử lý qua các lớp gộp và lớp tích chập, mô hình sẽ học những đặc trưng quan trọng trong hình như khuôn mặt, tai, miệng…, thì Tensor output của lớp cuối có kích thước H\*W\*D sẽ trở thành Vector có kích thước (H\*W\*D) và những lớp Fully Connected sẽ được sử dụng để hợp nhất những đặc trưng của hình để tạo ra output.

Graphical user interface

Description automatically generated with medium confidence

*Hình 2.31 Fully Connected Layer*

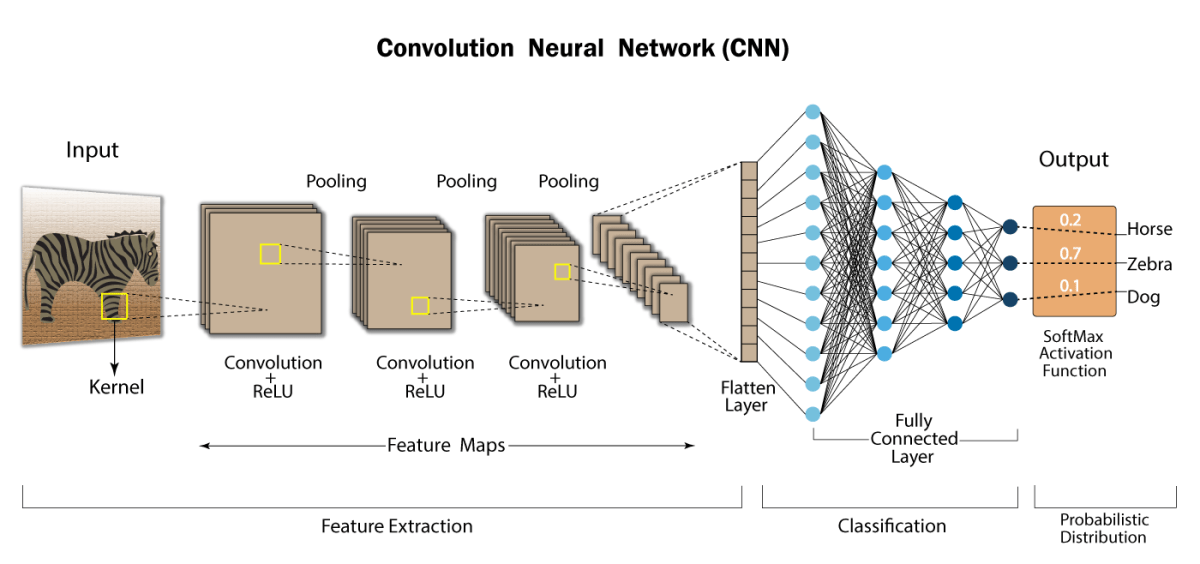
### **Cấu trúc của CNN**

Gồm các lớp Convolution được liên kết lại và sử dụng những hàm như tanh, ReLU để sử dụng trọng số trong các Neural để xây dựng những thông tin trừu tượng. Tính kết hợp và tính bất biến là các đặc tính quan trọng. Tính bất biến đảm bảo rằng mô hình không bị ảnh hưởng bởi các phép dịch chuyển, co giãn và quay của đối tượng bằng cách sử dụng Pooling Layer. Tính kết hợp cục bộ cho phép hiển thị các thông tin từ nhỏ đến lớn qua nhiều lớp tích chập từ những bộ lọc.

Mỗi lớp đều kết nối thông qua cơ chế Convolution, trong đó mỗi Neural trong các lớp kế tiếp được xây dựng từ kết quả của bộ lọc thuộc lớp trước đó áp dụng lên khu vực ảnh cục bộ của Neural phù hợp. Subsampling Layer và Pooling được dùng để lọc những thông tin có lợi và xóa đi các thông tin đã bị nhiễu đi. CNN học các giá trị thông qua lớp Filter tự động, sử dụng các mẫu tương tự như cách mà não người nhận diện ra được những vật thể trong môi trường xung quanh.

Ba thành phần chính của một mạng CNN bao gồm: Trường cục bộ, Lớp tổng hợp, Trọng số và bias.

* **Trường cục bộ:** Dùng để lọc và phân tích dữ liệu để tìm ra những nơi mà hình có giá trị cao nhất.
* **Trọng số và bias:** Giúp giảm số lượng tham số có vai trò lớn trong mạng CNN bằng cách sử dụng nhiều Feature Map khác nhau để nhận diện các đặc trưng của hình.
* **Lớp tổng hợp:** Có khả năng đơn giản hóa thông tin output bằng cách bỏ đi các thông tin không thừa và tối ưu output để đưa ra kết quả đúng với ý định của mình.



*Hình 2.32 Cấu trúc của CNN*

### **Ưu và nhược điểm của CNN**

* **Ưu điểm:**

- **Thích ứng với nhiều bài toán:** Học và tương thích với các bài toán mới bằng cách dùng lại trọng số đã huấn luyện từ trước, giúp tăng tính thích nghi và giảm thời gian trong khi huấn luyện.

- **Hiệu năng cao:** Xử lý hình ảnh với độ phân giải và độ phức tạp cao, đem đến tỉ lệ đúng hơn với các kỹ thuật truyền thống, đảm bảo hiệu suất cao.

- **Học tự động:** Những đặc điểm của ảnh được tự học một cách tự nhiên qua việc lặp lại các lớp tích chập, giúp cho khả năng phân loại và nhận diện đối tượng tốt hơn mà không phải xác định các đặc điểm trước đó.

- **Tính đa dụng:** CNN có tính đa dụng, dùng cho nhiều bài toán trong thị giác máy tính, gồm phân loại hình, nhận dạng chữ viết tay, v.v.

* **Nhược điểm:**

- **Cần nhiều dữ liệu:** Để có được hiệu quả tốt, CNN yêu cầu nhiều dữ liệu để học những đặc điểm của ảnh một cách tự nhiên.

- **Chi phí cao:** Có nhiều lớp Fully Connected và lớp Convolutional, do đó cần nhiều tài nguyên để tính toán và thời gian để huấn luyện, gây ra chi phí tính toán cao.

- **Ít hiệu quả trên bài toán đơn giản:** Khi dùng cho những bài toán đơn giản, nó sẽ gây ra sự quá khớp dữ liệu, khiến hiệu suất của mô hình không được tốt.

- **Khó hiểu:**  Vì CNN có nhiều lớp và tham số, việc hiểu và giải thích hoạt động của nó là khá khó.

### **Ứng dụng**

Mặc dù CNN thường dùng cho những việc liên quan tới thị giác máy tính, nhưng khả năng xử lý nhiều vấn đề tự học khác, đặc biệt liên quan tới chuỗi dữ liệu. Chứng minh là hoạt động hiệu quả trên các văn bản, video và âm thanh, thường được thêm vào với những mạng khác. Việc áp dụng kiến trúc mạng CNN trên dữ liệu chuỗi sẽ giải quyết nhiều vấn đề dữ liệu cụ thể, bao gồm việc dịch văn bản bằng máy, và những vấn đề khác.

*Hình 2.33 Ứng dụng của CNN*

### **Ví dụ về bài toán cho CNN**

#### **2.5.6.1 Xây dựng bài toán**

A picture containing qr code

Description automatically generatedTa có một bức hình có kích thước 28x28 chứa các chữ số 1-9, và ta muốn dự đoán chữ số đó là gì. Ví dụ:

*Hình 2.34 Dữ liệu trong MNIST Dataset*

#### **Chuẩn bị dữ liệu**

MNIST là bộ dữ liệu cơ bản về chữ số viết tay, bao gồm 2 tập: tập huấn luyện với 60.000 ảnh chứa những chữ số viết tay và tập kiểm tra với 10.000 ảnh chứa các chữ số. Trong đó, tập huấn luyện được chia làm hai tập, gồm 50.000 dữ liệu cho tập train và 10.000 dữ liệu cho tập validation, tập test vẫn là 10.000 dữ liệu.

#### **Xây dựng Model**

Diagram

Description automatically generatedModel cho bài toán là: Input ảnh -> Lớp tích chập (Conv) + Lớp gộp (Pool) -> Fully connected layer (FC) -> Đầu ra.

*Hình 2.35 Model cho bài toán CNN*

#### **Softmax Function**

Mỗi lớp trong mạng thực hiện hai bước là tính tổng tuyến tính của các nút ở lớp trước và áp dụng hàm kích hoạt (ví dụ như hàm sigmoid, hàm softmax). Vì output của mỗi nút là giá trị thực, ta cần dùng hàm softmax để biến giá trị đầu ra thành phần trăm.

Diagram

Description automatically generatedVì mỗi ảnh trong MNIST chỉ thuộc vào một trong 10 lớp từ 0 đến 9, lớp đầu ra có 10 nút giống với xác suất của hình thuộc vào mỗi lớp. Ví dụ, nếu node thứ 6 trong output layer có giá trị cao nhất, thì ảnh sẽ được dự đoán là số 5 (lưu ý rằng số node từ 0, và số của các số ở MNIST là từ 0).

*Hình 2.36 Softmax Function*

Text

Description automatically generated with medium confidence

Tổng quát hàm activation là:

A close-up of a clock

Description automatically generated with low confidence

Nhận xét:

Do đó, ta có thể xem giá trị của node thứ i trong output layer là xác suất của ảnh thuộc lớp (i-1).

Nếu chỉ có hai lớp thì hàm kích hoạt ở lớp đầu ra sẽ là hàm sigmoid, và nếu có nhiều hơn hai lớp thì hàm kích hoạt ở lớp đầu ra sẽ là hàm softmax.

Vì trong MNIST có 10 lớp giống với các số 0-9, nên output layer sẽ có 10 nút và hàm kích hoạt sẽ là hàm softmax.

#### **Loss Function**

Diagram

Description automatically generatedĐể xác định hàm mất mát (loss function), ta sử dụng phương pháp mã hóa one-hot để biến nhãn của hình từ số sang giá trị vector để giống với kích thước đầu ra .Ví dụ:

*Hình 2.37 Ví dụ cho hàm mất mát*

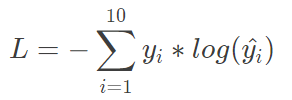
Lưu ý rằng nhãn của dữ liệu là số i được biểu diễn bằng giá trị vector v có kích cỡ 10\*1, trong đó vi+1 = 1 và những giá trị khác là 0. Với quy định % như trên, mã hóa one-hot là để đảm bảo hình trên là số 5 (nếu vi+1 = 1).

Bây giờ, chúng ta có giá trị thực (lable) được mã hóa và giá trị dự đoán ở lớp đầu ra với hàm kích hoạt Softmax có cùng cỡ 10\*1, ta sẽ xác định hàm mất mát để đánh giá hiệu suất mô hình.

Chart, histogram, scatter chart

Description automatically generated

*Hình 2.38 Các giá trị*

Để giá trị a6 gần bằng 1 trong khi các giá trị khác gần bằng 0, để dự đoán chính xác rằng hình input là số 5. Chúng ta sẽ xác định hàm mất mát như sau:

Giả sử ảnh đầu vào là số 5, hàm mất mát L sẽ được tính bằng công thức:

*Hình 2.39 Biểu đồ hàm mất mát L*

Chart

Description automatically generatedL = -log(y^6), y^6 là giá trị dự đoán tương ứng với lớp số 5 (lưu ý rằng chỉ số của node bắt đầu từ 0).

**Nhận xét:**

- Hàm L giảm dần 0-1.

- Lúc dự đoán giá trị y^6 gần bằng 1, nghĩa là giá trị dự đoán gần đúng với giá trị thật y6, hàm mất mát L sẽ nhỏ, gần bằng 0.

- Lúc dự đoán giá trị y^6 gần bằng 0, hàm mất mát sẽ tăng cao.

=> Hàm mất mát sẽ giảm khi giá trị dự đoán gần với giá trị thật và cao thì ra cho ra dự đoán sai. Hàm mất mát thấp thì dự đoán gần đúng với giá trị thật.

## **TỔNG QUAN VỀ RNN**

### **2.6.1 RNN là gì?**

Là một loại mạng Neural nhân tạo để giải quyết vấn đề về dữ liệu có tính tuần tự. RNN được tạo để lưu lại và dùng thông tin cho những bước trước đó để xử lý dữ liệu ở bước hiện tại.

RNN được xây dựng trên cơ sở của kiến trúc mạng Neural tiêu chuẩn bằng cách thêm một số đơn vị đặc biệt gọi là "Cell" để lưu lại trạng thái trước đó. Những thông tin được truyền tiếp thông qua nhiều bước khác nhau, giúp RNN có khả năng giải quyết vấn đề về dữ liệu dựa vào những điều đã xảy ra từ trước.

Diagram

Description automatically generatedRNN được dùng phổ biến về các lĩnh vực như dịch máy, nhận dạng và nhiều ứng dụng khác.

*Hình 2.40 Mô hình RNN*

Để giải thích, ta so sánh với mạng Neural Network thông thường, trong đó các input được truyền qua các Hidden Layer và đưa ra output. Tuy nhiên, những đầu vào xt phối hợp với Hidden Layer ht-1 thông qua fW để tính ra ht và đầu ra yt suy ra từ ht. W đại diện cho tập trọng số sử dụng trong các layer, và chúng được sử dụng ở các cụm L1, L2,..,Lt đại diện cho những hàm mất mát. Việc kết hợp h với ht-1 trong việc tính toán cho phép mạng RNN ghi nhớ thông tin từ những việc tính toán trước đó, giúp tăng tính chính xác cho dự đoán hiện tại. Công thức toán học:

A black text on a white background

Description automatically generated with medium confidence

Hàm fW thì dùng hàm tanh, công thức trên sẽ là:

A black text on a white background

Description automatically generated with low confidence

Có ba thành phần mới xuất hiện ở đây là Wxh, Whh và Why. Trong khi mạng Neural Network chỉ dùng một ma trận trọng số, thì RNN lại dùng tới ba ma trận trọng số khác nhau để thực hiện hai việc tính toán. Ma trận Whh phối hợp với ht-1, còn ma trận Wxh phối hợp với xt để tính toán ra ht, và phối hợp với Why để tính ra giá trị yt. Ngoài kiểu mô hình Many to Many, RNN còn có những mô hình khác như:

A picture containing diagram

Description automatically generated

*Hình 2.41 Các mô hình của RNN*

### **Ví dụ và ứng dụng**

#### **2.6.2.1 Ví dụ về bài toán RNN**

Ta có 1 tập dữ liệu đầu vào x = [h,e,l,o] và muốn dùng RNN để sinh ra 1 từ có ý nghĩa. Ta dùng mã hóa one-hot để biểu diễn các chữ cái trong từ.

Diagram

Description automatically generatedKết quả:

*Hình 2.42 Kết quả (6)*

Sau khi phát hiện ký tự bắt đầu là "h", chúng ta sử dụng mô hình RNN để tìm ra chữ cái tiếp theo có xác suất lớn nhất là "e". Sau đó, "e" được sử dụng như input cho cụm tiếp theo và tiếp tục thực hiện đến khi sinh ra 1 từ có nghĩa, trong trường hợp này là từ "hello".

#### **2.6.2.2 Ứng dụng của RNN**

**Diagram

Description automatically generated**Mạng Neural tích chập (CNN) thường dùng để phát hiện những đối tượng trong hình, sau đó mạng hồi quy (RNN) sẽ tạo ra các câu mô tả bức ảnh đó có ý nghĩa.

*Hình 2.43 Ứng dụng của RNN*

Ảnh trên là đã được áp dụng mạng CNN VGG16, mạng RNN để tạo ra hình ảnh này, bằng cách loại bỏ 2 lớp FC-1000 và Softmax của mạng CNN.

### **Ưu và nhược điểm**

**Ưu điểm:**

* + Lưu trữ thông tin từ các bước trước đó, giúp xử lý dữ liệu chuỗi có liên quan với các thời điểm trước đó.
  + Có thể mô hình hóa kết hợp với những thành phần có ở chuỗi, giúp dự đoán các thành phần tiếp theo.
  + Làm việc tốt với các tác vụ dự đoán chuỗi thời gian.
  + Có thể xử lý đầu vào có độ dài khác nhau.

**Nhược điểm:**

* Có thể bị mất thông tin vì phương pháp lan truyền ngược chỉ tập trung vào việc điều chỉnh các trọng số mới nhất của mạng. Do đó, các thông tin từ những bước trước có thể bị mất đi theo thời gian.
* Không thể xử lý được các chuỗi rất dài vì các trọng số của mạng có thể bị phân rã hoặc phát triển quá nhanh trong quá trình lan truyền tiến (Forward Propagation).
* Có thể mắc phải vấn đề Vanishing Gradient, khi đạo hàm của hàm mất mát về phía những bước trước kia trở nên rất nhỏ và gần như bằng 0, dẫn đến các trọng số sẽ không cập nhật tốt trong việc huấn luyện.

# **Chương 3 XÂY DỰNG BÀI TOÁN GIẢI CAPTCHA TỰ ĐỘNG**

## **Dữ liệu đầu vào**

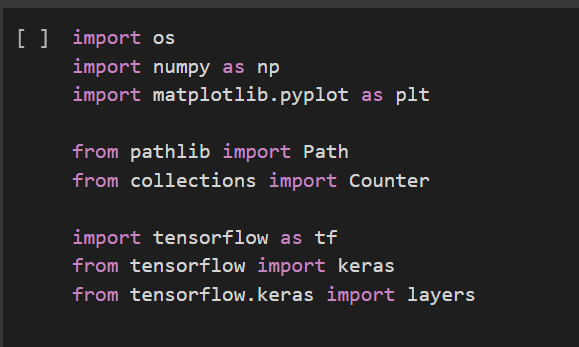
Gồm 1040 ảnh được lấy nguồn từ Kaggle captcha\_images\_v2.

*Hình 3.1 Input ảnh Captcha định dạng png, kích thước 200x50 pixel*

## **Mục tiêu của bài toán**

Dùng Deep Learning với mô hình kết hợp C-RNN để nhận dạng được Captcha, cho thấy được tỉ lệ và xác suất thành công.

## **Import thư viện**



*Hình 3.2 Import thư viện*

## **Tải bộ dữ liệu**

Đầu tiên ta dẫn đường dẫn đến thư mục dữ liệu sau đó lấy danh sách tất cả các hình ảnh gồm:

* **Images**: danh sách đường dẫn tới những file hình ảnh (.png) trong thư mục dữ liệu.
* **Labels**: danh sách các nhãn của các hình ảnh captcha.
* **Characters**: tập hợp các ký tự duy nhất xuất hiện trong các nhãn captcha.

A screen shot of a computer code

Description automatically generated with low confidence

*Hình 3.3 Tải bộ dữ liệu*

In thông tin về tập dữ liệu:

* + **len(images)** in ra số lượng hình ảnh được tìm thấy.
  + **len(labels)** in ra số lượng nhãn được tìm thấy.
  + **len(characters)** in ra số lượng ký tự duy nhất.
  + **characters** in ra các ký tự hiện có trong tập dữ liệu.

A picture containing text, screenshot, font

Description automatically generated

*Hình 3.4 In thông tin tập dữ liệu*

Thiết lập các thông số huấn luyện:

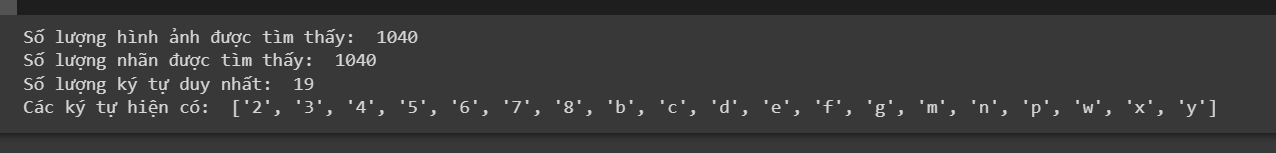
* + **batch\_size** là kích thước lô (batch size) cho quá trình huấn luyện và xác thực.
  + **img\_width** và **img\_height** là kích thước ảnh mong muốn cho quá trình huấn luyện.
  + **downsample\_factor** là tỷ lệ giảm mẫu của ảnh bằng các khối tích chập, Ta sẽ sử dụng hai khối các khối tích chập và mỗi khối sẽ có một lớp gộp (pooling layer) giảm mẫu đặc trưng đi một tỷ lệ là 2. Do đó, tổng tỷ lệ giảm mẫu sẽ là 4.
  + **max\_length** là độ dài tối đa của bất kỳ captcha nào trong tập dữ liệu.

A picture containing text, screenshot, font

Description automatically generated

*Hình 3.5 Thiết lập các thông số*

Kết quả:



*Hình 3.6 Kết quả (1)*

## **Tiền xử lý dữ liệu**

Trong nhãn của hình chứa các chuỗi gồm ký tự và số nên phải ánh xạ ký tự sang số nguyên để huấn luyện. Tiếp đến, khi ta dự đoán phải ánh xạ số nguyên trở lại các ký tự ban đầu.

A picture containing text, screenshot, font

Description automatically generated

*Hình 3.7 Ánh xạ*

Đoạn code trên là:

- Tạo hai ánh xạ dùng để chuyển đổi ký tự thành số nguyên và ngược lại. Đầu tiên, ánh xạ "char\_to\_num" được tạo bằng cách sử dụng lớp "StringLookup" từ gói "layers" của TensorFlow. Nó ánh xạ các ký tự trong danh sách "characters" thành các số nguyên. Biến "mask\_token" được đặt thành None để không ánh xạ các ký tự đặc biệt.

- Ánh xạ "num\_to\_char" tạo ra bằng cách dùng cùng "StringLookup", nhưng dùng phương thức "get vocabulary()" của ánh xạ "char\_to\_num" để lấy danh sách ký tự gốc và sử dụng nó làm từ vựng.

Tiếp theo, ta dùng hàm split\_data để lấy tổng kích thước của tập dữ liệu, tạo một mảng chỉ số (indices array) và trộn nó, nếu cần thiết. Tiếp đến, ta lấy kích thước mẫu, chia dữ liệu thành tập huấn luyện và xác thực.

A screen shot of a computer program

Description automatically generated with low confidence

*Hình 3.8 Chia dữ liệu*

Đầu vào gồm các mảng "images" và "labels", kích thước huấn luyện được xác định bằng tham số "train\_size" (mặc định là 0.9), và nếu "shuffle" được đặt thành True (mặc định là True), dữ liệu sẽ được trộn ngẫu nhiên trước khi chia. Hàm trả về các mảng tương ứng với tập huấn luyện và xác thực.

Tiếp theo, Ta dùng hàm encode\_single\_sample để xử lý dữ liệu. Đầu vào gồm đường dẫn “img\_path” của ảnh và nhãn. Các bước gồm:

* + Đọc ảnh từ đường dẫn.
  + Giải mã và chuyển đổi ảnh thành ảnh xám.
  + Chuyển đổi thành kiểu dữ liệu float32 từ khoảng 0 đến 1.
  + Thay đổi kích thước ảnh thành kích thước mong muốn.
  + Hoán vị ảnh để chiều thời gian tương ứng với chiều rộng của ảnh.
  + Ánh xạ các ký tự trong nhãn thành các số bằng cách sử dụng ánh xạ "char\_to\_num".
  + Trả về một từ điển chứa ảnh và nhãn, đáp ứng yêu cầu của mô hình với hai đầu vào "image" và "label".

A picture containing text, screenshot, font

Description automatically generated

*Hình 3.9 Encode*

## **Khởi tạo đối tượng trong tập dữ liệu**

Ta tạo những đối tượng từ các mảng để dùng trong lúc huấn luyện và cung cấp dữ liệu cho mô hình trong những vòng lặp.

A screen shot of a computer program

Description automatically generated with low confidence

*Hình 3.10 Khởi tạo đối tượng*

Các bước trong mã là như sau:

* Tạo một đối tượng từ các cặp dữ liệu huấn luyện. Đối tượng này sẽ chứa các mẫu huấn luyện và nhãn tương ứng.
* Dùng phương thức **map** để ánh xạ mỗi mẫu và nhãn thông qua hàm **mã hóa**. Hàm này có nhiệm vụ mã hóa (encode) các mẫu thành dạng phù hợp để huấn luyện. Tham số được dùng để tăng tốc độ xử lý bằng việc tự động điều chỉnh số lượng tiến trình song song.
* Dùng phương thức **batch** để tạo các batch (nhóm) dữ liệu có kích thước. Điều này giúp tiện lợi cho việc huấn luyện mô hình bằng cách xử lý một lượng lớn mẫu cùng một lúc.
* Dùng phương thức **prefetch** để tạo một bộ đệm (buffer) có kích thước. Điều này giúp tối ưu hóa việc đọc dữ liệu từ ổ đĩa hoặc các nguồn dữ liệu từ xa bằng cách trước đoán và chuẩn bị dữ liệu tiếp theo trong quá trình huấn luyện.

## **Hiển thị dữ liệu**

Dùng thư viện matplotlib để hiện một lưới ảnh và nhãn tương ứng của chúng.

A picture containing text, screenshot, font

Description automatically generated

*Hình 3.11 Hiện ảnh và nhãn*

Giải thích:

* Dòng đầu là khởi tạo một lưới gồm 4 hàng và 4 cột (tổng cộng 16 ô) để hiển thị ảnh và nhãn. Biến **ax** là một mảng 2 chiều các đối tượng trục (axis) đại diện cho từng ô trong lưới.
* Vòng lặp for lấy ra một batch (nhóm) dữ liệu từ **train\_dataset**. **train\_dataset** là một tập dữ liệu đào tạo (training dataset) và được giả định là một tập dữ liệu chứa các ảnh và nhãn tương ứng.
* Trích xuất các ảnh và nhãn từ batch dữ liệu. Các ảnh được lưu trong biến **images**, và các nhãn được lưu trong biến **labels**.
* Thêm vòng lặp for lặp qua từng phần tử trong khoảng từ 0 đến 15 (16 ô trong lưới).
* Chuyển đổi ảnh từ định dạng tensor thành định dạng mảng numpy và định dạng giá trị pixel từ dạng thực về dạng nguyên từ 0 đến 255.
* Chuyển đổi nhãn từ dạng số thành dạng chuỗi bằng cách sử dụng hàm **num\_to\_char** và sau đó chuyển đổi từ tensor thành mảng numpy và dùng decode (utf-8) để chuyển đổi sang dạng chuỗi (string).
* Hiển thị ảnh trong ô thứ i của lưới. **imshow** là hàm trong thư viện matplotlib để hiển thị ảnh. **img[:, :, 0]** là lấy kênh đầu tiên của ảnh (nếu ảnh màu), và **.T** là để chuyển đổi chiều của ma trận ảnh (nếu cần). **cmap="gray"** chỉ định sử dụng màu xám để hiển thị ảnh.
* Đặt tiêu đề cho ô hiển thị ảnh bằng chuỗi nhãn **label**.
* Tắt các trục (axis) của ô hiển thị ảnh.
* Hiển thị lưới ảnh và nhãn tương ứng trên màn hình.

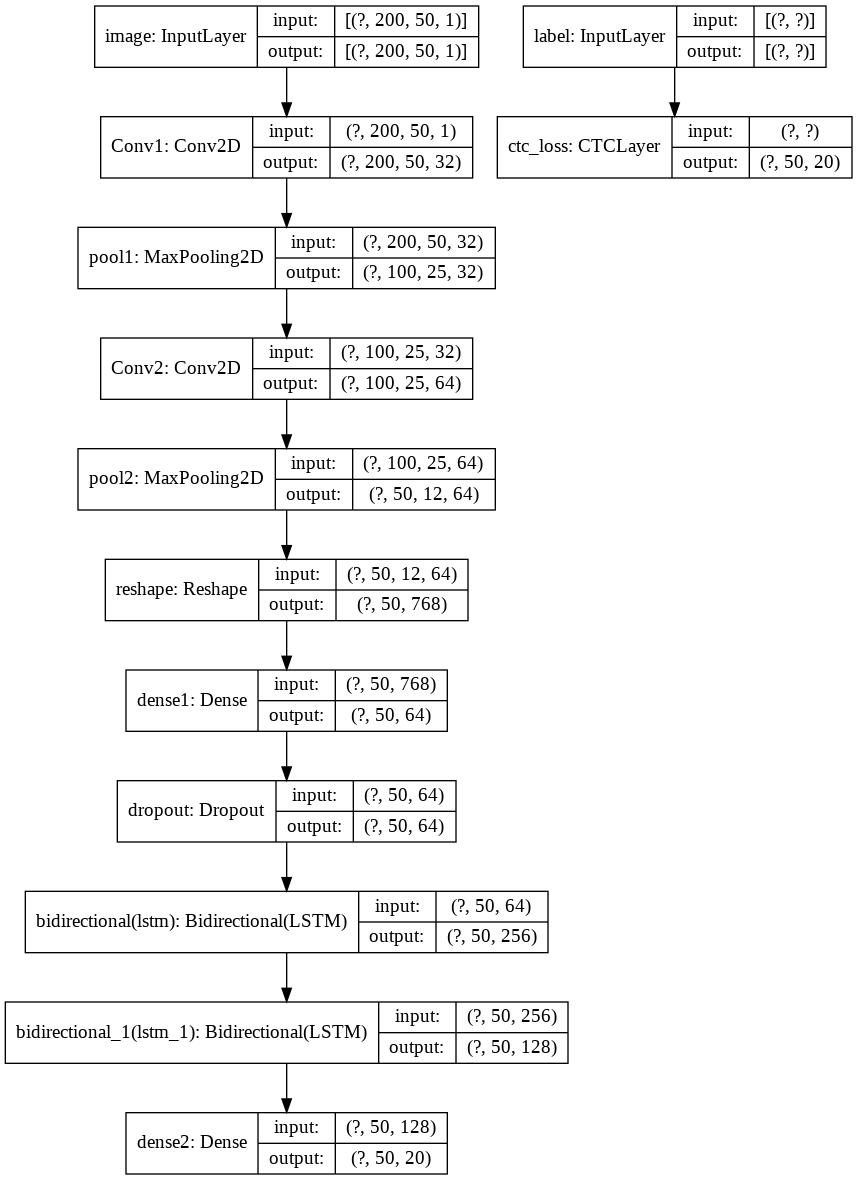
A screenshot of a computer

Description automatically generated with low confidenceKết quả:

*Hình 3.12 Kết quả (2)*

## **Khởi tạo model C-RNN**

Ta xây dựng một mạng kết hợp C-RNN với output là một lớp CTC loss. Cấu trúc của model:



*Hình 3.13 Cấu trúc model*

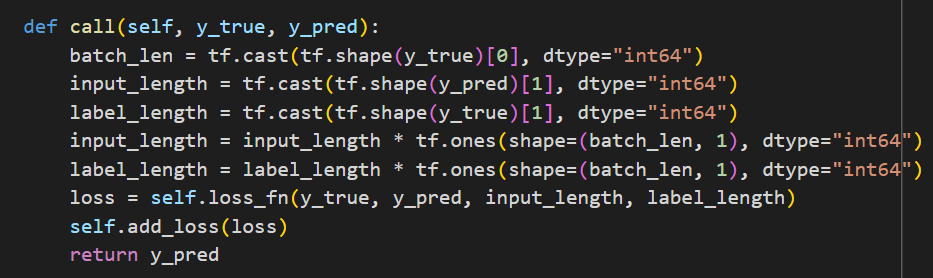
Khởi tạo lớp CTC là một lớp con của lớp layers.Layer trong TensorFlow/Keras. Dùng phương thức khởi tạo của lớp là hàm tính toán giá trị mất mát CTC.

A screen shot of a computer program

Description automatically generated with low confidence

*Hình 3.14 Khởi tạo lớp CTC*

Tiếp theo ta dùng phương thức call của lớp để tính độ mất mát trong lúc huấn luyện và thêm nó vào model. Thêm giá trị mất mát trong quá trình huấn luyện vào lớp bằng cách sử dụng self.add\_loss() và ở thời điểm kiểm tra, chỉ cần trả về các dự đoán đã tính toán.



*Hình 3.15 Phương thức call*

Đầu vào là **y\_true** (nhãn thực tế) và **y\_pred** (dự đoán của mô hình). Trong phương thức này, các giá trị cần thiết như **batch\_len**, **input\_length**, và **label\_length** được tính toán và định dạng phù hợp. Sau đó, giá trị mất mát được tính bằng cách gọi **self.loss\_fn** với các đối số tương ứng và sau đó được thêm vào mô hình bằng **self.add\_loss()**. Cuối cùng, dự đoán **y\_pred** được trả về.

Tiếp theo ta tạo và cấu hình model định nghĩa cho các đầu vào là hình và nhãn và có kiểu dữ liệu float32.

A picture containing text, screenshot, font, line

Description automatically generated

*Hình 3.16 Xây dựng model*

Xây dựng hai khối Conv sử dụng lớp Conv2D và lớp MaxPooling2D.

A picture containing text, screenshot, font

Description automatically generated

*Hình 3.17 Các khối Conv*

- Khối Conv đầu tiên dùng 32 bộ lọc có kích thước (3, 3) để trích xuất đặc điểm từ đầu vào, hàm kích hoạt relu được sử dụng sau khi tính toán. Tham số padding được đặt là same để đảm bảo kích thước đầu ra của khối giống với kích thước đầu vào. Một lớp pooling được thêm vào để giảm kích thước của đầu ra sử dụng kích thước (2, 2) và sử dụng max pooling để giữ lại đặc trưng quan trọng.

- Khối Conv thứ hai sủ dụng 64 bộ lọc để tiếp tục trích xuất đặc điểm từ đầu vào, là đầu ra conv đầu tiên cũng tương tự như các bước trước ở khối conv đầu, lớp pooling được áp dụng để giảm kích thước đầu ra.

Sau khi sử dụng hai lớp max pooling với kích thước và bước nhảy (strides) là 2, các feature map đã được giảm mẫu xuống còn 1/4 kích thước ban đầu. Số bộ lọc trong lớp cuối cùng là 64. Ta cần thay đổi kích thước phù hợp trước khi truyền đầu ra vào phần RNN. Để tính toán kích thước mới cho việc reshape feature map trước khi truyền vào phần RNN của mô hình.

A picture containing text, screenshot, font

Description automatically generated

*Hình 3.18 Tính toán kích thước mới*

* Đây là một phép tính để tính toán kích thước mới của đầu vào **x** sau khi áp dụng các lớp convolution và pooling trước đó. Kích thước mới được tính bằng cách chia chiều rộng (**img\_width**) và chiều cao (**img\_height**) của đầu vào đi cho 4, và nhân chiều cao mới với 64.
* Lớp Reshape được dùng để thay đổi hình dạng của đầu vào **x** thành **new\_shape**. Để chuẩn bị cho các lớp Dense (fully connected) tiếp theo trong mô hình. Lớp Reshape có tham số **target\_shape** được đặt là **new\_shape**, và được đặt tên là "reshape".
* Lớp Dense là một lớp fully connected trong mạng neural network. Đầu vào của lớp này là **x**, là đầu ra của lớp Reshape trước đó. Lớp Dense có 64 đơn vị đầu ra và sử dụng hàm kích hoạt "relu" để tạo ra những đặc trưng phi tuyến tính. Lớp này được đặt tên là "dense1".
* Lớp Dropout được sử dụng để ngẫu nhiên loại bỏ một phần tử của đầu vào trong lúc huấn luyện, với tỷ lệ dropout là 0.2 (20%). Điều này giúp tránh việc mô hình trở nên quá phụ thuộc vào một số đặc trưng cụ thể và giúp kháng nhiễu.

Tiếp theo, hai lớp RNNs được thêm vào.

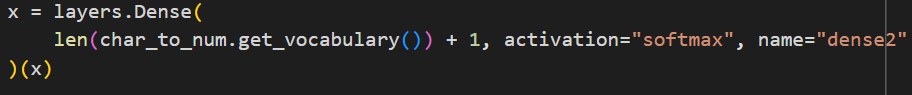


*Hình 3.19 RNN*

- Đầu tiên, một lớp LSTM được áp dụng lên đầu vào **x**. Lớp LSTM này có 128 đơn vị đầu ra và được đặt để trả về các chuỗi đầu ra (**return\_sequences=True**). Một đặc điểm của lớp LSTM là khả năng duy trì thông tin trong quá trình xử lý chuỗi. Lớp Bidirectional được sử dụng để bọc lớp LSTM, cho phép nó xử lý chuỗi đầu vào không chỉ theo thứ tự ban đầu mà còn theo thứ tự đảo ngược. Tham số dropout có giá trị 0.25, cho biết tỷ lệ dropout là 25%, nghĩa là 25% các đơn vị trong quá trình huấn luyện sẽ bị tắt ngẫu nhiên để tránh overfitting.

- Một lớp LSTM khác được áp dụng lên đầu ra của lớp LSTM trước đó (**x**). Lớp LSTM này cũng có 64 đơn vị đầu ra và trả về các chuỗi đầu ra. Tương tự như trước, lớp Bidirectional và dropout được dùng để cải thiện hiệu suất và khả năng kháng nhiễu.

Tiếp theo, lớp đầu ra được định nghĩa bằng lớp dense với trong đó **char\_to\_num.get\_vocabulary()** là một đối tượng từ điển (dictionary) được dùng để ánh xạ ký tự vào số nguyên hàm kích hoạt softmax để tạo ra một phân phối xác suất trên những đơn vị đầu ra, đại diện cho xác suất của mỗi ký tự trong từ điển.



*Hình 3.20 Lớp Dense*

Tiếp theo, thêm lớp CTC (Connectionist Temporal Classification) để tính toán mất mát CTC tại mỗi bước.



*Hình 3.21 Lớp CTC*

Định nghĩa mô hình được biên dịch và trả về với tối ưu **Adam**.

A screen shot of a computer code

Description automatically generated with low confidence

*Hình 3.22 Định nghĩa và biên dịch mô hình*

Lấy mô hình và dùng hàm summary() được gọi để in ra tóm tắt của model.



*Hình 3.23 Lấy và tóm tắt mô hình*

Kết quả:

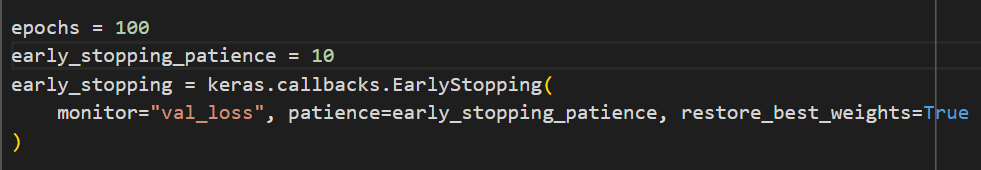
A screenshot of a computer program

Description automatically generated with medium confidence

*Hình 3.24 Kết quả (3)*

## **Huấn luyện mô hình**

Ta sẽ training 100 epochs, có thể training nhiều hơn nếu muốn tăng độ chính xác. thêm chức năng dừng sớm (early stopping) vào việc huấn luyện. Chức năng dừng sớm giúp giảm thiểu quá trình huấn luyện khi không có sự tiến bộ và giúp tránh việc mô hình bị quá khớp (overfitting).



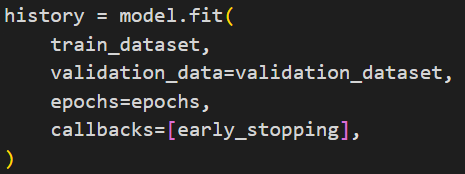
*Hình 3.25 Huấn luyện mô hình*

Trong đoạn mã trên:

- Tham số "patience" của dừng sớm (early stopping). Nếu không có cải thiện trong **patience** epoch liên tiếp trên tập validation, quá trình huấn luyện sẽ dừng lại.

- Đây là một callback của Keras để thực hiện dừng sớm. Nó sẽ theo dõi giá trị mất mát trên tập validation (**monitor="val\_loss"**) và dừng huấn luyện nếu không có cải thiện trong **patience** epoch liên tiếp. Tham số sẽ khôi phục trọng số của mô hình về trạng thái tốt nhất (có mất mát nhỏ nhất trên tập validation).

Tiếp theo, là lúc huấn luyện của mô hình. Mô hình được huấn luyện trên **train\_dataset** và đánh giá trên **validation\_dataset**. Số lượng epoch là **epochs**. Callback **early\_stopping** được dùng để kiểm soát quá trình dừng sớm. Kết quả của quá trình huấn luyện được lưu vào biến **history**.

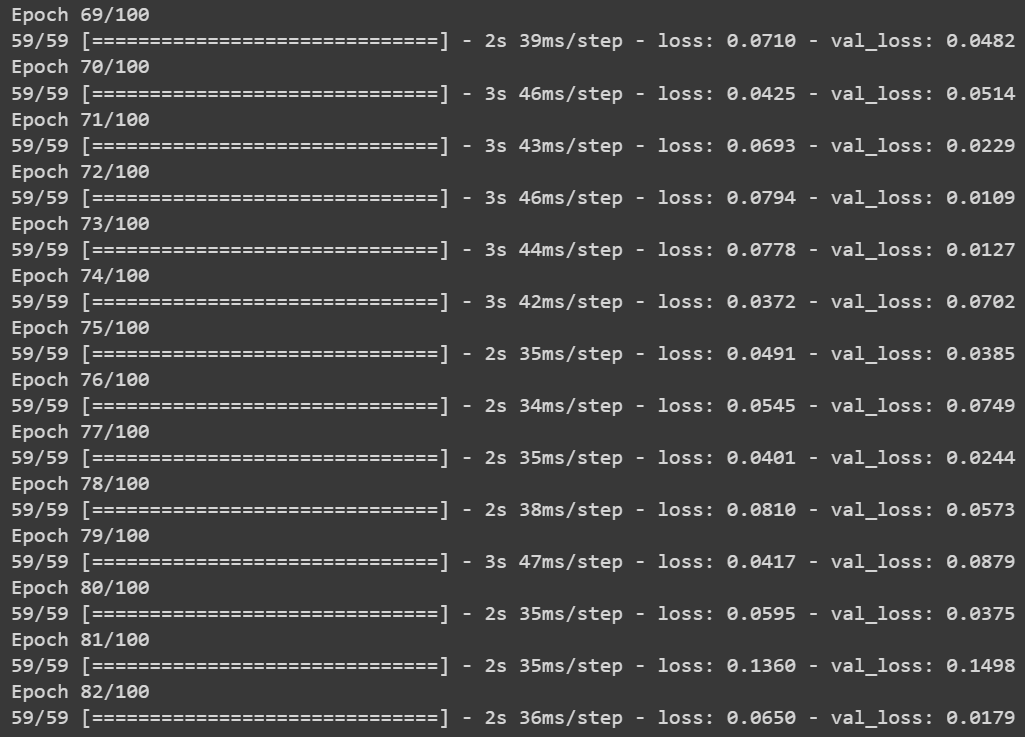


*Hình 3.26 Kết quả huấn luyện*

Đoạn mã mô tả quá trình huấn luyện với dữ liệu huấn luyện (**train\_dataset**) và dữ liệu validation (**validation\_dataset**). Các tham số huấn luyện đã được định nghĩa trước đó bao gồm số lượng epoch (**epochs**) và đối tượng **early\_stopping** để kiểm tra dừng sớm.

* **model.fit()**: Phương thức này khởi động quá trình huấn luyện. Nó nhận vào dữ liệu huấn luyện và dữ liệu validation, và thực hiện việc cập nhật trọng số dựa trên dữ liệu huấn luyện để tối ưu hóa hàm mất mát.
* **train\_dataset**: Dữ liệu huấn luyện, được đưa vào để huấn luyện. Đây có thể là một đối tượng Dataset của TensorFlow chứa những cặp (đầu vào, nhãn) để huấn luyện mô hình.
* Dữ liệu validation, được dùng để đánh giá hiệu suất sau mỗi epoch. Đây cũng có thể là một đối tượng Dataset của TensorFlow.
* Số lượng epoch (vòng lặp) sẽ được huấn luyện cho mô hình.
* Danh sách các callback được áp dụng trong lúc huấn luyện. Trong trường hợp này, chỉ có callback **early\_stopping** được sử dụng để kiểm tra dừng sớm.
* Biến **history** lưu trữ thông tin về quá trình huấn luyện, bao gồm những chỉ số và đồ thị của hàm mất mát và các độ đo hiệu suất (nếu có) sau mỗi epoch.

Kết quả:



*Hình 3.27 Kết quả (4)*

Tiếp theo, ta tạo ra một đồ thị để hiển thị sự biến thiên của mất mát trong lúc huấn luyện và kiểm tra mô hình. Giúp đánh giá hiệu suất của mô hình và phát hiện sự tăng giảm của mất mát qua các epoch.

A picture containing text, font, screenshot

Description automatically generated

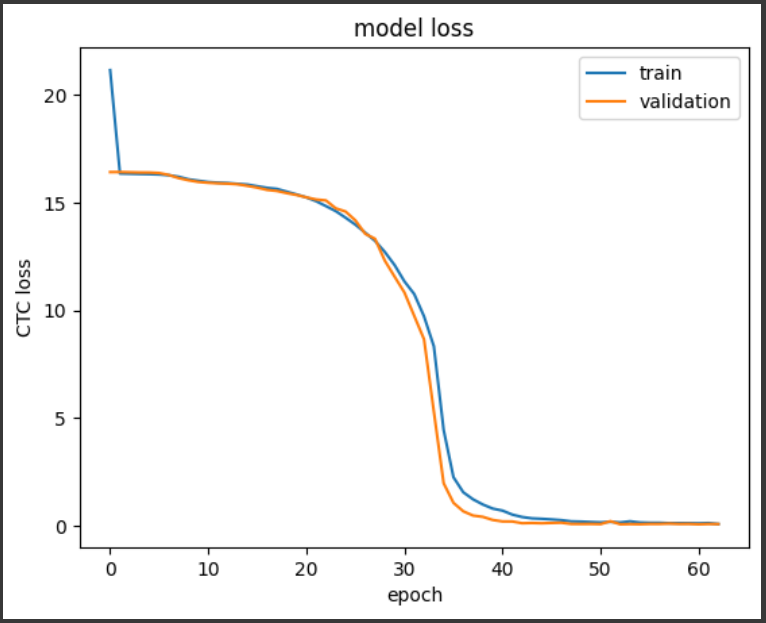
*Hình 3.28 Trực quan hóa đồ thị*

- Dòng đầu tiên trong đoạn mã. Nó tạo ra một đường đồ thị cho giá trị mất mát của mô hình trong quá trình huấn luyện. Trong trường hợp này, giá trị mất mát được lưu trữ trong biến **history.history['loss']**. Dòng này tạo ra một đường cong màu xanh đại diện cho mất mát của quá trình huấn luyện.

**-** Tương tự như dòng trước, nhưng nó vẽ đường đồ thị cho giá trị mất mát trên tập dữ liệu validation (kiểm tra) trong quá trình huấn luyện. Giá trị mất mát trên tập validation được lưu trữ trong biến **history.history['val\_loss']**. Đường cong này được tạo ra màu cam và đại diện cho mất mát trên tập validation.

**-** Đặt tiêu đề, đặt nhãn cho trục y và trục x của đồ thị. Tạo ra một chú thích cho đồ thị, hiển thị tên của từng đường đồ thị.

Kết quả:



*Hình 3.29 Đồ thị*

## **Dự đoán**

Ta sẽ tạo mô hình dự đoán các Captcha chứa trong tập dữ liệu dựa trên mô hình đã huấn luyện, mô hình dự đoán này chỉ gồm một phần của mô hình ban đầu, từ layer đầu vào đến layer đầu ra. Đầu tiền ta sẽ trích xuất các lớp cho đến lớp đầu ra.

A screen shot of a computer program

Description automatically generated with low confidence

*Hình 3.30 Khởi tạo mô hình dự đoán*

* Đoạn mã này sử dụng lớp **Model** trong keras để tạo một mô hình mới (**prediction\_model**). Đối số đầu tiên là layer đầu vào của mô hình gốc, được truy cập bằng cách dùng phương thức **get\_layer()** và truyền vào tên của layer ("image"). Đối số thứ là layer đầu ra của mô hình gốc, cũng được truy cập bằng cách dùng **get\_layer()** và truyền vào tên của layer ("dense2").
* Phương thức **summary()** được gọi trên **prediction\_model** để hiển thị tóm tắt của mô hình dự đoán. Tóm tắt này cho biết thông tin về các layer trong mô hình, kích thước của output shape của mỗi layer và tổng số tham số của mô hình.

Tiếp theo, ta dùng hàm tiện ích để giải mã kết quả đầu ra của mạng dùng kỹ thuật Greedy Search để giải mã các dự đoán. Đầu vào của hàm là **pred**, một tensor chứa các kết quả dự đoán.

A picture containing text, screenshot, font, line

Description automatically generated

*Hình 3.31 Greedy Search*

* Biến **input\_len** được khởi tạo là một mảng numpy với kích thước bằng số hàng của **pred** và giá trị của mỗi phần tử là **pred.shape[1]**. Điều này đại diện cho chiều dài đầu vào của mỗi mẫu trong batch dự đoán.
* Hàm **decode** được dùng để giải mã những dự đoán CTC (Connectionist Temporal Classification). Đầu vào của hàm bao gồm **pred** (dự đoán từ mô hình), chiều dài đầu vào của mỗi mẫu và dùng kỹ thuật greedy để giải mã
* Hàm này trả về kết quả giải mã dưới dạng một danh sách những chuỗi ký tự có thể. Mỗi phần tử trong danh sách đại diện cho một mẫu trong batch dự đoán.
* Phần tử đầu tiên trong danh sách là kết quả giải mã cho mẫu đầu tiên trong batch.
* Một phần của kết quả giải mã được cắt đi để đảm bảo rằng độ dài của kết quả không vượt quá, giới hạn này được định nghĩa trước đó.
* Hàm trả về biến **results**, chứa các kết quả giải mã của các dự đoán từ mô hình.

Hàm này duyệt qua các kết quả và trả về văn bản tương ứng.

A picture containing screenshot, text, font

Description automatically generated

*Hình 3.32 Hàm duyệt và trả về văn bản*

Đoạn mã trên thực hiện một số thao tác để chuyển đổi kết quả giải mã từ dạng số sang dạng văn bản:

* Tạo một danh sách rỗng để lưu trữ kết quả văn bản giải mã.
* Duyệt qua từng phần tử trong danh sách **results**, đại diện cho kết quả giải mã của từng mẫu trong batch.
* Dùng hàm **num\_to\_char** để chuyển đổi các số trong **res** thành các ký tự tương ứng. Để chắc rằng những dự đoán đã được biến đổi thành dạng văn bản.
* Hàm này được dùng để kết hợp các ký tự trong mỗi dự đoán thành một chuỗi văn bản duy nhất. Chuyển đổi chuỗi kết quả từ định dạng Tensor sang định dạng chuỗi numpy, sau đó giải mã chuỗi từ định dạng utf-8.
* Thêm chuỗi kết quả giải mã **res** vào danh sách.
* Trả về danh sách, chứa các kết quả giải mã đã được biến đổi thành dạng văn bản.

Tiếp theo, ta kiểm tra kết quả trên một số mẫu trong vòng lặp **for,** đoạn mã này lặp qua một lô mẫu xác nhận để thực hiện dự đoán và so sánh với nhãn thực tế.

A black background with white text

Description automatically generated with low confidence

*Hình 3.33 Dự đoán và so sánh với nhãn thực thế*

* Lấy 1 batch từ tập validation\_dataset. Đây là một phương thức của dataset trong TensorFlow, nó cho phép lấy một số lượng nhất định các phần tử từ dataset.
* Duyệt qua từng batch trong tập validation\_dataset. Vì đã sử dụng **.take(1)**, vòng lặp chỉ thực hiện một lần với một batch duy nhất.
* Lấy dữ liệu hình ảnh từ batch hiện tại. Trong đoạn mã này, "image" là một khóa (key) trong batch và nó chứa dữ liệu hình ảnh.
* Lấy dữ liệu nhãn (label) từ batch hiện tại. Tương tự như trên, "label" là một khóa trong batch và chứa dữ liệu nhãn tương ứng với các hình ảnh trong batch.
* Các biến **batch\_images** và **batch\_labels** sẽ được dùng để đánh giá mô hình trên batch dữ liệu validation này.

Dự đoán đầu ra từ mô hình dự đoán bằng cách truyền một lô hình ảnh vào.



*Hình 3.34 Dự đoán đầu ra*

Giải mã các dự đoán thành văn bản tương ứng bằng cách sử dụng hàm **decode.**



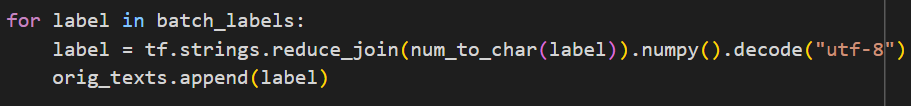
*Hình 3.35 Dùng hàm decode để giải mã*

Tạo một danh sách để lưu trữ các nhãn thực tế.



*Hình 3.36 Tạo danh sách lưu trữ*

Trong vòng lặp **for** đoạn mã này duyệt qua các nhãn thực tế trong lô dữ liệu và giải mã chúng thành văn bản tương ứng.



*Hình 3.37 Vòng lặp for duyệt và giải mã thành văn bản*

Tạo một biểu đồ hiển thị các hình ảnh gồm dữ liệu và kết quả dự đoán.

A picture containing text, screenshot, font, software

Description automatically generated

*Hình 3.38 Tạo biểu đồ*

* Tạo một lưới đa dạng với kích thước 4x4 và kích thước tổng thể của hình ảnh là (15, 5). Kết quả trả về từ hàm này bao gồm một figure object và một mảng chứa các axis objects (biến **ax**) tương ứng với mỗi ô trong lưới.
* Duyệt qua từng phần tử trong danh sách, đại diện cho các dự đoán văn bản tương ứng với từng hình ảnh.
* Lấy hình ảnh từ batch\_images. Đoạn mã này sử dụng chỉ số **i** để truy cập hình ảnh thứ **i** trong batch\_images. Dòng tiếp theo nhân với 255 để chuyển đổi giá trị từ [0, 1] thành [0, 255], và cuối cùng chuyển đổi thành kiểu dữ liệu **np.uint8**.
* Chuyển đổi hình ảnh từ định dạng (width, height) sang (height, width). Điều này là cần thiết vì matplotlib mong đợi hình ảnh được cung cấp dưới dạng (height, width).
* Tạo một tiêu đề cho hình ảnh hiển thị, bao gồm dự đoán văn bản tương ứng.
* Hiển thị hình ảnh trong ô tương ứng của lưới. chọn axis object tương ứng với ô tại hàng và cột được dùng để hiện ảnh **img**, với cmap="gray" chỉ định rằng sử dụng một mức xám để hiển thị.
* Đặt tiêu đề cho ô hiển thị hình ảnh tương ứng.
* Tắt các trục tọa độ (axis) trong ô hiển thị hình ảnh.
* Hiển thị figure chứa các hình ảnh và tiêu đề tương ứng.

Từ thư viện SKLearn ta dùng để tính toán những độ đo đánh giá mô hình như độ chính xác và in ra màn hình.



*Hình 3.39 Độ chính xác*

Kết quả:

A screenshot of a computer program

Description automatically generated with medium confidence

*Hình 3.40 Kết quả (5)*

A screenshot of a computer

Description automatically generated with medium confidence

*Hình 3.41 Kết quả (6)*

## **Giao diện chính**

A screenshot of a computer

Description automatically generated with medium confidence

*Hình 3.42 Giao diện chính*

Truyền ảnh vào và ấn submit.

A screenshot of a computer

Description automatically generated with medium confidence

*Hình 3.43 Thao tác trên giao diện*

Kết quả:

A screenshot of a phone

Description automatically generated with low confidence

*Hình 3.44* *Kết quả (7)*

# **Chương 4: KẾT LUẬN**

## **4.1 ĐÁNH GIÁ ĐỀ TÀI**

### **4.1.1 Kết quả**

* Mô hình đã có thể dự đoán chính xác.
* Đã tạo được một giao diện tương tác người dùng.
* Bổ sung và làm vững hơn nền tảng kiến thức về ngôn ngữ Python.

### **4.1.2 Khuyết điểm**

* Các chức năng còn đơn giản, chưa hoàn toàn tối ưu.
* Giao diện đơn giản, các chức năng còn ít.
* Chưa dự đoán được các dạng hình Captcha khác nhau.

## **4.2 HƯỚNG HOÀN THIỆN**

* Phát triển dự đoán đa dạng Captcha: Việc phát triển theo hướng này sẽ giúp dự đoán được nhiều Captcha. Ngoài ra, quá trình huấn luyện sẽ tốn thời gian, tài nguyên và nhân lực để có thể hoàn thành.
* Phát triển toàn diện: hướng phát triển này thì mô hình sẽ trở nên tốt nhất. Nhưng để hoàn thiện được thì việc bỏ ra thời gian, tài nguyên và nhân lực cũng sẽ rất lớn.

# **TÀI LIỆU THAM KHẢO**

[1] “Build & Share Delightful Machine Learning Apps” [online]

<<https://gradio.app/docs/>>

[2] “Cấu trúc mạng Convolutional Neural Network” [online] <<https://nttuan8.com/bai-6-convolutional-neural-network/>>

[3] “How CAPTCHAs work | What does CAPTCHA mean?” [online]

<<https://www.cloudflare.com/learning/bots/how-captchas-work/>>

[4] Nguyễn Hải Triều (2022). Bài giảng Lập trình Python. Trường ĐH Nha Trang.

[5] “OCR model for reading Captchas” [online]

<<https://keras.io/examples/vision/captcha_ocr/>>

[6] “Recurrent Neural Network (Phần 1): Tổng quan và ứng dụng” [online]

<<https://viblo.asia/p/recurrent-neural-networkphan-1-tong-quan-va-ung-dung-jvElaB4m5kw>>