|  |  |  |
| --- | --- | --- |
|  | **TRƯỜNG ĐẠI HỌC BÁCH KHOA**  **KHOA CÔNG NGHỆ THÔNG TIN** |  |

BÁO CÁO

PBL4 - ĐỒ ÁN HỆ THỐNG THÔNG MINH

**TÊN ĐỀ TÀI**

**Xây dựng hệ thống phát hiện và nhận diện khuôn mặt khi đeo hoặc không đeo khẩu trang**

Giảng viên hướng dẫn: Nguyễn Văn A

|  |  |
| --- | --- |
| STT NHÓM: …  HỌ VÀ TÊN SINH VIÊN | LỚP HỌC PHẦN ĐỒ ÁN |
| … | 17N16A |
| … | 17N16B |
| … | 17N16A |
| … | 17N16C |

*(SV lấy thông tin ở link để điền vào bảng:* [*https://dutudn-my.sharepoint.com/:x:/g/personal/pcthang\_dut\_udn\_vn/Ecc5Pf0CQz9NoQ9Mass78KIB56U7OIiols5o7qxssuJksA?e=BSdceV*](https://dutudn-my.sharepoint.com/:x:/g/personal/pcthang_dut_udn_vn/Ecc5Pf0CQz9NoQ9Mass78KIB56U7OIiols5o7qxssuJksA?e=BSdceV)

*Phần này chỉ để ghi chú, SV sẽ bỏ đi ko đưa vào trang bìa này của ĐA).*

ĐÀ NẴNG, …./2021

**TÓM TẮT ĐỒ ÁN**

Sinh viên mô tả tóm tắt đồ án trong 01 đoạn văn bản, từ 7 đến 10 dòng, nêu 03 ý lớn: vấn đề cần giải quyết, phương pháp giải quyết và kết quả đạt được. Phần này trình bày trên 01 trang riêng, ngay sau trang bìa.

**BẢNG PHÂN CÔNG NHIỆM VỤ**

|  |  |  |
| --- | --- | --- |
| Sinh viên thực hiện | Các nhiệm vụ | Tự đánh giá theo 3 mức  (Đã hoàn thành/Chưa hoàn thành/Không triển khai) |
| Nguyễn A | -…   * … | -…  -… |
| Trần B |  |  |
|  |  |  |

**MỤC LỤC**

**(tối đa 3 cấp)**

**CÁC YÊU CẦU VIẾT BÁO CÁO**

**VỀ HÌNH THỨC:**

* Định dạng: căn chỉnh lề kiểu Justify; font chữ 13, times new roman; giãn dòng 1.5, cách trước 6pt; hình ảnh phải đánh số, có chú thích bên dưới hình; bảng phải đánh số, có chú thích bên trên bảng.
* Báo cáo có độ dài khoảng 20 trang và không đóng bìa gương (chỉ cần bìa mềm có màu) gồm: trang bìa (trang 1 của file hướng dẫn này), tóm tắt, bảng phân công nhiệm vụ,mục lục, và các trang nội dung.

**VỀ NỘI DUNG:**

Khi viết báo cáo cần chú ý:

* Báo cáo để người khác đọc hiểu được logic vấn đề và giải pháp, và sau khi đọc xong thì người đọc có thể làm lại được sản phẩm đồ án theo nội dung báo cáo.
* Báo cáo không phải để kể lể các việc đã triển khai và/hoặc để liệt kê mã nguồn.
* Báo cáo gồm 04 phần sau.

# 1. Giới thiệu

1.1 Tổng quan ( Viết thêm)

Trong đề tài, nhóm chúng em sẽ xây dựng hệ thống nhận diện khuôn mặt có đeo khẩu trang hoặc không đeo khẩu trang để đưa cảnh báo, là tiền đề cở sở để phát triển hệ thống cảnh báo mang khẩu trang đối với mọi người đặt biệt là ở nơi công cộng tập trung đông người như trung tâm thương mại, trường học, bệnh viện,…

Trong đề này nhóm chúng em xây dựng **“Ứng dụng nhận diện khuôn mặt đeo khẩu trang”** với thư viện OpenCV, Keras/Tensorflow, và Deep Learning.

1.2 Mục đích

Xây dựng hệ thống phát hiện và nhận diện khuôn mặt khi đeo hoặc không đeo khẩu trang , từ đó đưa ra cảnh báo trên mang hình giúp cho giám sát màn hình có thể giám sát tình hình đảm bảo an toàn trong giai đoạn dịch bệnh nguy hiểm hiện nay.

Tìm hiểu về thư viện OpenCV khi sử dụng trong môi trường lập trình bằng ngôn ngữ Python.

Tìm hiểu về Keras/Tensorflow.

Tìm hiểu về Deep Learning: cách hoạt động và ứng dụng của Deep Learning trong đề tài này.

1.3 Phương pháp

1.3.1 Đối tượng

* Các phương pháp, thuật toán phục vụ cho việc phát hiện và nhận dạng khuôn mặt người trên ảnh.
* Bộ thư viện xử lý ảnh OpenCv
* Bộ thư viện xử lý và nhận dạng Tensorflow
* Phương pháp nhận diện và học sâu Deep Learning
* (nhóm đài thêm nếu thiếu)

1.3.2 Phạm vi nghiên cứu

* Tập trung tìm hiểu nhận dạng khuôn mặt (Face Recognition)
* Công nghệ theo dõi nhận dạng tập trung ROI (Region Of Interest)
* Xử lí ảnh khuôn mặt từ bộ dữ liệu đầu vào có sẵn và huấn luyện để nhận diện.
* Phát hiện khuôn mặt với webcam và nhận diện mang khẩu trang hoặc không .
* Nếu không mang khẩu trang thì thực hiện nhận diện khuôn mặt bằng (Face Recognition)

1.4 Cấu trúc đồ án

1. Giới thiệu

2. Giải pháp

3. Kết quả

4 Kết luận

# 2. Giải pháp

2.1 Cơ sở lý thuyết

2.1.1 Công cụ thực thiện

a) Giới thiệu về ngôn ngữ Python

Python là một ngôn ngữ lập trình bậc cao cho các mục đích lập trình đa năng, do Guido van Rossum tạo ra và lần đầu ra mắt vào năm 1991. Python được thiết kế với ưu điểm mạnh là dễ đọc, dễ học và dễ nhớ. Python là ngôn ngữ có hình thức rất sáng sủa, cấu trúc rõ ràng, thuận tiện cho người mới học lập trình. Cấu trúc của Python còn cho phép người sử dụng viết mã lệnh với số lần gõ phím tối thiểu.

Python hoàn toàn tạo kiểu động và dùng cơ chế cấp phát bộ nhớ tự động; do vậy nó tương tự như Perl, Ruby, Scheme, Smalltalk, và Tcl. Python được phát triển trong một dự án mã mở, do tổ chức phi lợi nhuận Python Software Foundation quản lý.

Ban đầu, Python được phát triển để chạy trên nền Unix. Nhưng rồi theo thời gian, Python dần mở rộng sang mọi hệ điều hành từ MS-DOS đến Mac OS, OS/2, Windows, Linux và các hệ điều hành khác thuộc họ Unix. Mặc dù sự phát triển của Python có sự đóng góp của rất nhiều cá nhân, nhưng Guido van Rossum hiện nay vẫn là tác giả chủ yếu của Python. Ông giữ vai trò chủ chốt trong việc quyết định hướng phát triển của Python.

b )Giới thiệu về thư viện OpenCV

OpenCv (Open Source Computer Vision) là một thư viện mã nguồn mở về thị giác máy với hơn 500 hàm và hơn 2500 các thuật toán đã được tối ưu về XLA, và các vấn đề liên quan tới thị giác máy. OpenCv được thiết kế một cách tối ưu, sử dụng tối đa mạnh của các dòng chip đa lõi… để thực hiện các phép tính toán trong thời gian thực, nghĩa là tốc độ đáp ứng của nó thể đủ nhanh cho các ứng dụng thông thường.

OpenCv là thư viện được thiết kế để chạy trên nhiều nền tảng khác nhau (cross-platform),nghĩa là nó có thể chạy trên hệ điều hành Window, Linux, Mac, iOS… Việc sử dụng thư viện OpenCv tuân theo các quy định về sử dụng phần mềm mã nguồn mở BSD do đó bạn có thể sử dụng thư viện này một cách miễn phí cho các mục đích phi thương mại lẫn thương mại.

Dự án về OpenCv được khởi động từ những năm 1999, đến năm 2000 nó được giới thiệu trong một hội nghị của IEEE về các vấn đề trong thị giác máy và nhận dạng, tuy nhiên bản OpenCV 1.0 mãi tới tận năm 2006 mới chính thức được công bố và năm 2008 bản 1.1 (prerelease) mới được ra đời. Tháng 10 năm 2009, bản OpenCV thế hệ thứ hai ra đời (thường gọi là phiên bản 2.x), phiên bản này có giao diện của C++ (khác với phiên bản trước có giao diện của C) và có nhiều điểm khác biệt so với phiên bản thứ nhất.

Thư viện OpenCV ban đầu được sự hỗ trợ từ Intel, sau đó được hỗ trợ bởi Willow Garage, một phòng thí nghiệm chuyên nghiên cứu về công nghệ robot. Cho đến nay, OpenCV vẫn là thư viện mở, được phát triển bởi nguồn quỹ không lợi nhuận (none-profit foundation) và được sự hưởng ứng rất lớn của cộng đồng.

c) Deep Learning

Trí tuệ nhân tạo đang len lỏi vào trong cuộc sống và ảnh hưởng sâu rộng tới mỗi chúng ta, các cụm từ “Artificial Intelligence”, “Machine Learing”, “Deep Learning” đã không còn quá xa lạ gì. Chúng ta cùng xem hình vẽ để mô tả lại mối quan hệ giữa artificial intelligence, machine learning, và deep learning:

Deep learning đã và đang là một chủ đề AI được bàn luận sôi nổi. Là một phạm trù nhỏ của machine learning, deep learning tập trung giải quyết các vấn đề liên quan đến mạng thần kinh nhân tạo nhằm nâng cấp các công nghệ như nhận diện giọng nói, tầm nhìn máy tính và xử lý ngôn ngữ tự nhiên. Deep learning đang trở thành một trong những lĩnh vực hot nhất trong khoa học máy tính. Chỉ trong vài năm, deep learning đã thúc đẩy tiến bộ trong đa dạng các lĩnh vực như nhận thức sự vật (object perception), dịch tự động (machine translation), nhận diện giọng nói,… – những vấn đề từng rất khó khăn với các nhà nghiên cứu trí tuệ nhân tạo.

Để hiểu hơn về deep learning, hãy nhìn lại một số khái niệm cơ bản về trí tuệ nhân tạo.

Trí tuệ nhân tạo có thể được hiểu đơn giản là được cấu thành từ các lớp xếp chồng lên nhau, trong đó mạng thần kinh nhân tạo nằm ở dưới đáy, machine learning nằm ở tầng tiếp theo và deep learning nằm ở tầng trên cùng.

### d) Thư viện của Deep Learning – Keras/Deep Learning

Keras được coi là một thư viện ‘high-level’ với phần ‘low-level’ (còn được gọi là backend) có thể là TensorFlow, CNTK, hoặc Theano. Keras có cú pháp đơn giản hơn TensorFlow rất nhiều. Với mục đích giới thiệu về các mô hình nhiều hơn là các sử dụng các thư viện deep learning, tôi sẽ chọn Keras với TensorFlow là ‘backend’.

Những lý do nên sử dụng Keras để bắt đầu:

* Keras ưu tiên trải nghiệm của người lập trình
* Keras đã được sử dụng rộng rãi trong doanh nghiệp và cộng đồng nghiên cứu
* Keras giúp dễ dàng biến các thiết kế thành sản phẩm
* Keras hỗ trợ huấn luyện trên nhiều GPU phân tán
* Keras hỗ trợ đa backend engines và không giới hạn bạn vào một hệ sinh thái

Tensorflow - Với sự bùng nổ của lĩnh vực Trí Tuệ Nhân Tạo – A.I. trong thập kỷ vừa qua, machine learning và deep learning rõ ràng cũng phát triển theo cùng. Và ở thời điểm hiện tại, TensorFlow chính là thư viện mã nguồn mở cho machine learning nổi tiếng nhất thế giới, được phát triển bởi các nhà nghiên cứu từ Google. Việc hỗ trợ mạnh mẽ các phép toán học để tính toán trong machine learning và deep learning đã giúp việc tiếp cận các bài toán trở nên đơn giản, nhanh chóng và tiện lợi hơn nhiều.

Kiến trúc TensorFlow hoạt động được chia thành 3 phần:

– Tiền xử lý dữ liệu

– Dựng model

– Train và ước tính model

Cách TensorFlow hoạt động:

TensorFlow cho phép các lập trình viên tạo ra dataflow graph, cấu trúc mô tả làm thế nào dữ liệu có thể di chuyển qua 1 biểu đồ, hay 1 sê-ri các node đang xử lý. Mỗi node trong đồ thị đại diện 1 operation toán học, và mỗi kết nối hay edge giữa các node là 1 mảng dữ liệu đa chiều, hay còn được gọi là ‘tensor’.

TensorFlow cung cấp tất cả những điều này cho lập trình viên theo phương thức của ngôn ngữ Python. Vì Python khá dễ học và làm việc, ngoài ra còn cung cấp nhiều cách tiện lợi để ta hiểu được làm thế nào các high-level abstractions có thể kết hợp cùng nhau. Node và tensor trong TensorFlow là các đối tượng Python, và các ứng dụng TensorFlow bản thân chúng cũng là các ứng dụng Python.

Các operation toán học thực sự thì không được thi hành bằng Python. Các thư viện biến đổi có sẵn thông qua TensorFlow được viết bằng các binary C++ hiệu suất cao. Python chỉ điều hướng lưu lượng giữa các phần và cung cấp các high-level abstraction lập trình để nối chúng lại với nhau.

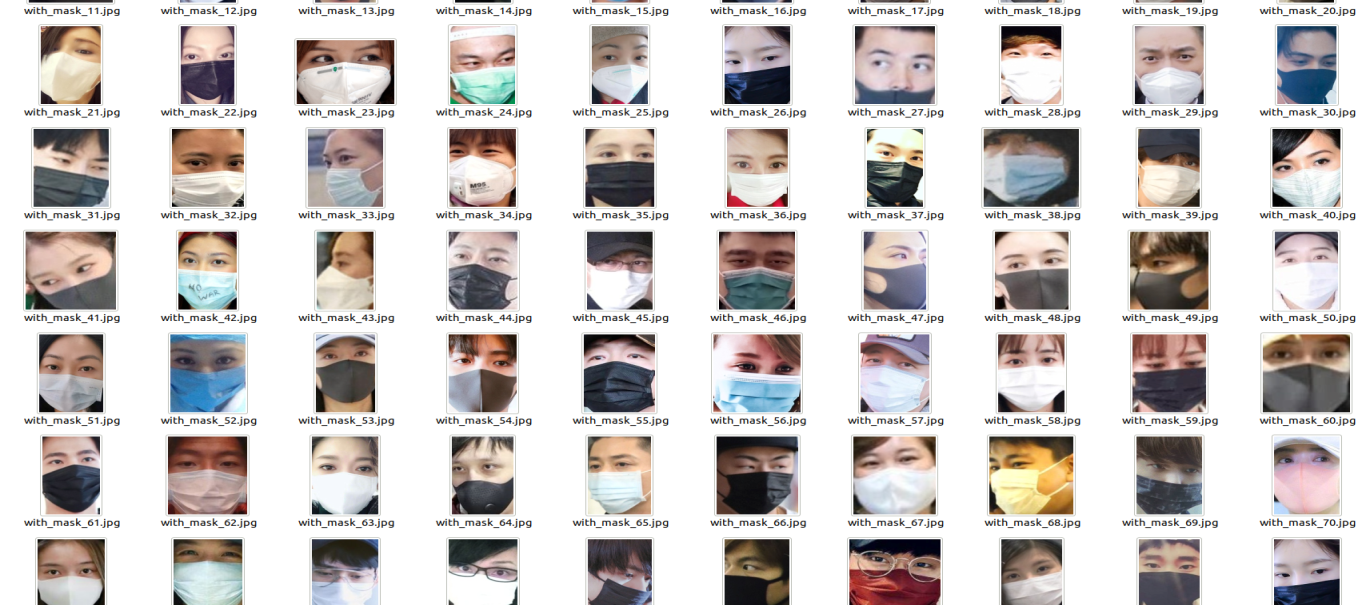
TensorFlow 2.0, được ra mắt vào tháng 10 năm 2019, cải tiến framework theo nhiều cách dựa trên phản hồi của người dùng, để dễ dàng và hiệu quả hơn khi làm việc cùng nó (ví dụ: bằng cách sử dụng các Keras API liên quan đơn giản cho việc train model). Train phân tán dễ chạy hơn nhờ vào API mới và sự hỗ trợ cho TensorFlow Lite cho phép triển khai các mô hình trên khá nhiều nền tảng khác nhau. Tuy nhiên, nếu đã viết code trên các phiên bản trước đó của TensorFlow thì bạn phải viết lại, đôi lúc 1 ít, đôi lúc cũng khá đáng kể, để tận dụng tối đa các tính năng mới của TensorFlow 2.0.

**2.2 Nhận diện khẩu trang**

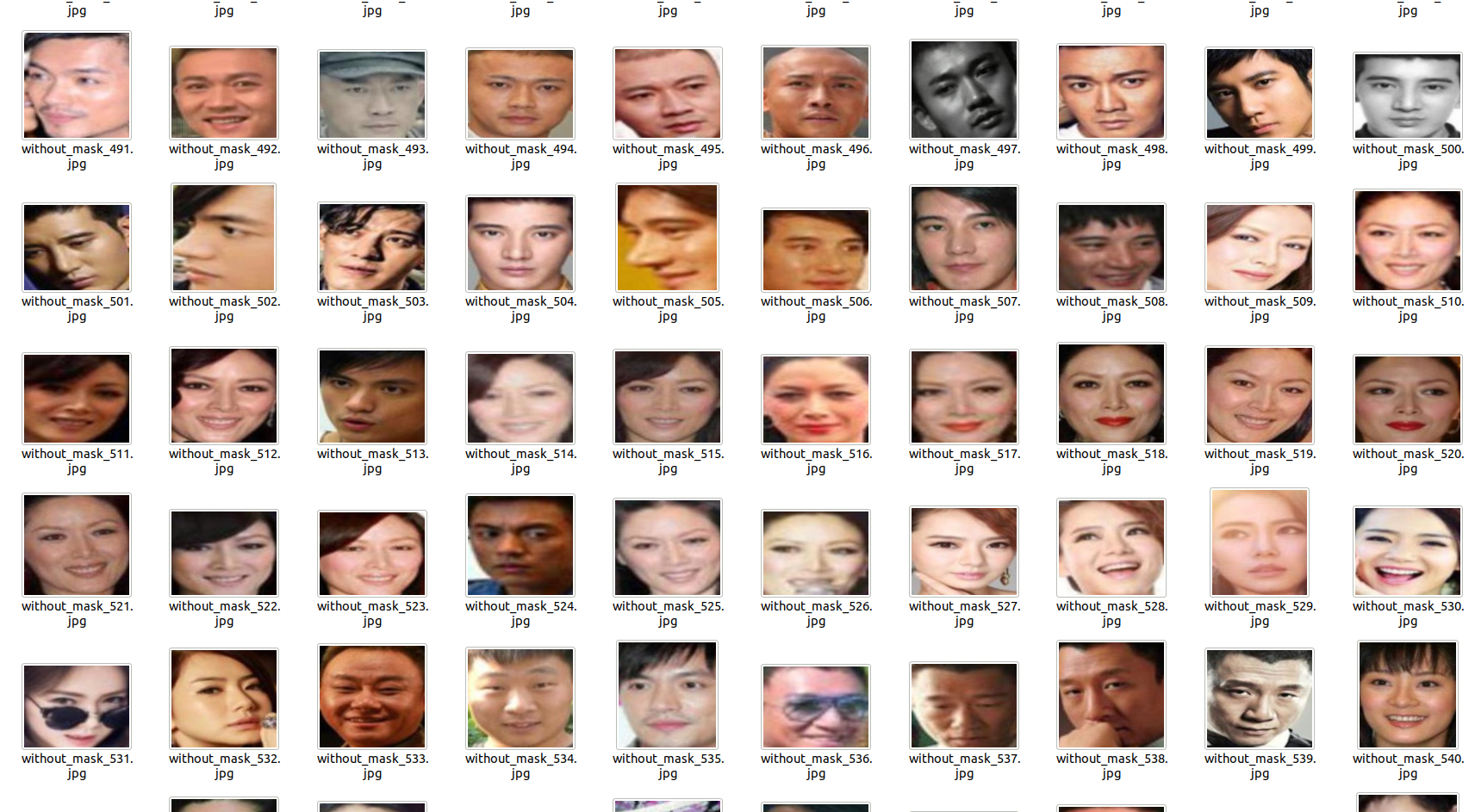
2.2.1 Tiền xử lý ảnh

Để xây dựng tập dữ liệu phục vụ cho việc huấn luyện (train) dữ liệu. - Tập ảnh chân dung với khuôn mặt không đeo khẩu trang (without mask): chúng ta tạo ra một kho ảnh chụp chân dung của nhiều người khác nhau, 3828 ảnh, sau đó training tập này với định danhh là không đeo khẩu trang - Tập ảnh chân dung với khuôn mặt đeo khẩu trang (with mask): chúng ta tạo ra một kho ảnh chụp chân dung của nhiều người khác nhau, 3725 ảnh, sau đó training tập này với định danhh là đeo khẩu trang.

Các bước xử lý trước bao gồm thay đổi kích thước thành 224 × 224 pixel, chuyển đổi sang định dạng mảng, chuyển kênh màu BGR sang RGB và chia tỷ lệ cường độ pixel trong hình ảnh đầu vào thành phạm vi [-1, 1]. Sau đó sử dụng scikit-learn One-HotEncoding để tạo nhãn lớp cho mỗi hình ảnh. Trong chiến lược này, mỗi véc-tơ giá trị nhãn đầu ra được chuyển đổi thành dạng mới, trong đó chỉ có 1 đầu ra bằng “1” ứng với mã phân loại của véc-tơ đầu vào tương ứng còn các đầu ra khác đều bằng “0”.



Hình . Tập dataset người mang khẩu trang



Hình . Tập dataset người không khẩu trang

2.2.2 Huấn luyện dữ liệu

Sử dụng MobileNetV2 một phiên bản của CNN(Mạng nơron tích chập) là mô hình hướng di động, nhẹ và hiệu quả do số lượng tham số ít. MobileNetV2 xây dựng dựa trên các ý tưởng từ MobileNetV1 [10], sử dụng tích chập có thể phân tách theo chiều sâu với các đặc trưng mới cho kiến trúc.

Trong bước này, chia dữ liệu thành tập huấn luyện chứa các hình ảnh mà mô hình CNN sẽ được huấn luyện và tập kiểm tra với các hình ảnh mà mô hình sẽ kiểm tra. Cụ thể lấy split\_size = 0.8, có nghĩa là 80% tổng số hình ảnh cho huấn luyện và 20% còn lại của hình ảnh cho kiểm tra một cách ngẫu nhiên. Sau đó, xây dựng mô hình CNN với các lớp khác nhau như AveragePooling2D với trọng số 7×7, Flatten, Dropout và Dense. Trong lớp Dense cuối cùng, sử dụng hàm softmax để xuất ra một véc - tơ thể hiện xác suất của mỗi lớp.

Tiền xử lý ảnh

Trích chọn đặc trưng

Huấn luyện mô hình

Hình 1 . Huấn luyện mô hình

Phẳng hóa ma trận bằng vecto

Pooling theo giá trị trung bình

MobileNetV2

Ảnh nguồn

Kích hoạt phi tuyến Relu 128 Kênh ra

Hàm kích hoạt sortmax 2 kênh ra

Dropout p=0,5

Kết quả

Hình 2. Mô hình huấn luyện chi tiết

2.2.3 Vận hành mô hình

Tải hình ảnh, video đầu vào sau đó phát hiện khuôn mặt trong ảnh. Áp dụng bộ phát hiện khẩu trang để phân loại khuôn mặt đeo khẩu trang (with\_ mask) hoặc là không đeo khẩu trang (without\_ mask). Tiền xử lý được thực hiện là chuẩn hóa kích thước và hoán đổi kênh màu. Để giảm nhiễu cho ảnh do chiếu sáng cần chuyển ảnh sang dạng ảnh blob thông qua hàm blobFromImage của OpenCV.

Load ảnh/video cần kiểm tra

Phát hiện mặt từ ảnh/video

Áp dụng mô hình dữ liệu

Đưa ra kết quả dự đoán

Hình 2. Mô hình vận hành

2.2.3 Nhận diện khẩu trang

* Đối với vấn đề xác định người trong video/webcam có đeo khẩu trang hay không cần xác định các khuôn mặt trong webcam và phân loại khuôn mặt đeo khẩu trang. Đầu tiên, lặp qua các frame từ video và thay đổi kích thước để có chiều rộng tối đa 400 pixel, phát hiện khuôn mặt trong frame và xác định xem họ có đang đeo khẩu trang hay không?

Ảnh khuôn mặt

Load phát hiện khuôn mặt

Trích xuất ảnh từ các frame

Khởi tạo video

Chuyển kết quả vào frame

Tiền xử lí ảnh

Load mô hình MobileNetV2

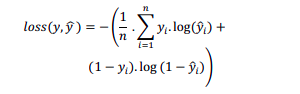
Hiển thị kết quả

Hình 4. Sơ đồ phát hiện người đeo khẩu trang trên video/webcam

2.2.4 Kết quả đạt được

2.2.4.1 Kết quả phần huấn luyện Model

* Để đánh giá tập trọng số cần xác định lỗi cho cả huấn luyện (loss) và kiểm tra (val\_loss) ta sử dụng hàm Cross Entropy là nhị phân chéo binary\_crossentropy. Cụ thể, tính toán loss của mỗi trường hợp bằng cách tính giá trị trung bình như sau:



Với n là số lượng giá trị vô hướng trong đầu ra của mô hình, hàm loss trả về một số thực không âm thể hiện sự chênh lệch giữa hai đại lượng yˆ là xác suất nhãn được dự đoán và y là xác suất của nhãn đúng. Sau đó sử dụng thuật toán gradient descent “adam” (Adaptive Mô - ment Estimator) để tối ưu [13].

Hơn nữa, để kiểm định hiệu năng của mô hình phân loại, cần tính toán tỷ lệ chính xác trung bình trên tất cả các dự đoán sử dụng thang đo ma trận nhầm lẫn như sau:

Bảng 1. Ma trận nhầm lẫn

|  |  |  |
| --- | --- | --- |
|  | Dự đoán là Positive | Dự đoán là Negative |
| Thực tế là Positive | TP | FN |
| Thực tế là Negative | FP | TN |

Trong đó:

Các hàng của ma trận là nhãn lớp thực tế, các cột của ma trận là nhãn lớp dự đoán.

- TN: Số lượng khuôn mặt không đeo khẩu trang được phân loại chính xác.

- FN: Số lượng khuôn mặt đeo khẩu trang bị phân loại nhầm là khuôn mặt không đeo khẩu trang.

- TP: Số lượng khuôn mặt đeo khẩu trang được phân loại chính xác.

- FP: Số lượng khuôn mặt không đeo khẩu trang bị phân loại nhầm là khuôn mặt đeo khẩu trang.

Từ đó, độ chính xác của mô hình được tính theo công thức sau:

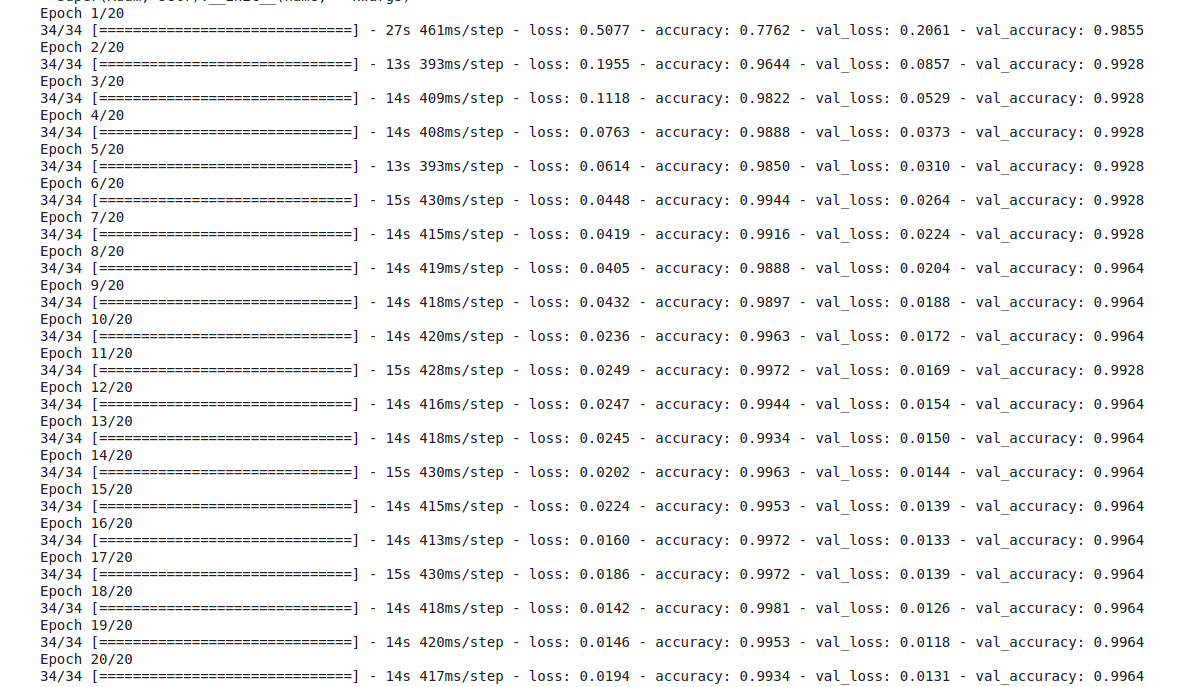
Accuracy =

Đây là tỉ lệ của tất cả trường hợp phân loại đúng (không phân biệt negative/positive) trên toàn bộ trường hợp trong mẫu kiểm tra. Một độ đo cũng thường được dùng để đánh giá mô hình phân lớp đó là F-measure hay F-core được tính dựa trên 2 độ đo khác là precision và recall, và được tính như sau:

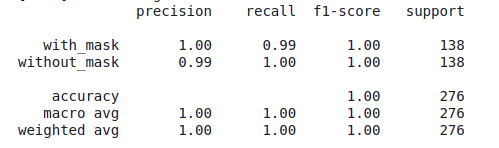
Precision =

*Recall =*

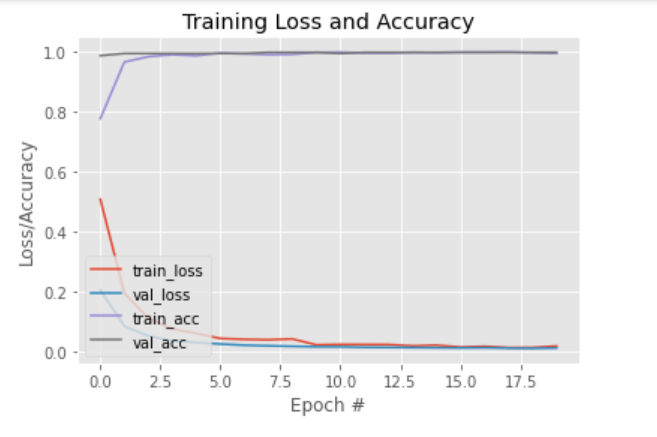
F score = =



Hình 5. Đánh giá huấn luyện mô hình qua các Epoch



Hình 6. Đánh giá mô hình tổng quan



Hình 7. Đồ thị độ chính xác và giá trị lỗi

2.3. Nhận diện khuôn mặt

2.3.1 Tiền xử lý ảnh

Phần này áp dụng một số phương pháp tiền xử lý trên hình ảnh đầu vào, bao gồm phát hiện và cắt xén để lấy vùng ảnh chứa khuôn mặt, cải thiện chất lượng ảnh. Trong thực tế ứng dụng, ảnh đầu vào thường được trích xuất từ camera nên bao gồm cả không gian nền, do đó, ta phải thực hiện giai đoạn tìm kiếm và phát khuôn mặt (gọi là face detection) nhằm xác định vùng ảnh chứa đúng khuôn mặt cần xử lý và cắt bỏ không gian nền của ảnh. Để thực hiện điều này, các tác giả sử dụng phương pháp phát hiện vùng ảnh có chứa khuôn mặt dựa vào DNN.

Khi ảnh khuôn mặt được phát hiện, thực hiện cắt vùng ảnh khuôn mặt đó từ nền, cải thiện chất lượng ảnh khuôn mặt này bằng việc chuyển đổi hình ảnh đầu vào thành hình ảnh đa cấp độ xám và áp dụng phép cân bằng mức xám, co giãn về kích thước đúng với đầu vào của mạng nơron đã thiết kế để thực hiện trích chọn đặc trưng và phân lớp.





Hình: Cắt xén ảnh

2.3.2 Huấn luyện dữ liệu

Facenet lấy hình ảnh khuôn mặt của một người làm đầu vào và xuất ra một vectơ gồm 128 thể hiện những đặc điểm quan trọng nhất của khuôn mặt. Trong học máy, vector này được gọi là embedding. Tất cả các thông tin quan trọng của một bức hình đều được nhúng trong vector này. FaceNet lấy khuôn mặt của một người và nén nó thành một vector gồm 128 số.

Embedding là vectơ và chúng ta có thể giải thích vectơ là điểm trong tọa độ Đề-cát. Có nghĩa là có thể vẽ hình ảnh của một khuôn mặt trong hệ tọa độ bằng cách sử dụng các embedding của nó.

Chúng tôi không trực tiếp cho FaceNet biết các số trong vectơ sẽ đại diện cho những gì trong quá trình đào tạo, chúng tôi chỉ yêu cầu các vectơ embedding của các khuôn mặt của cùng một người cũng tương tự nhau (tức là gần nhau). Tùy thuộc vào FaceNet để tìm ra cách biểu diễn khuôn mặt bằng vectơ sao cho vectơ của những người giống nhau và vectơ của những người khác nhau thì không. Để điều này trở thành sự thật, FaceNet cần xác định các đặc điểm chính trên khuôn mặt của một người để tách biệt nó với các khuôn mặt khác nhau. FaceNet đang thử nhiều cách kết hợp khác nhau của những tính năng này trong quá trình đào tạo cho đến khi tìm thấy những tính năng hoạt động tốt nhất. FaceNet (hoặc mạng nơ-ron nói chung) không thể hiện các tính năng trong hình ảnh giống như cách chúng ta làm (khoảng cách, kích thước, v.v.). Đó là lý do tại sao thật khó để giải thích những vectơ này, nhưng chúng tôi khá chắc chắn rằng một cái gì đó như khoảng cách giữa các mắt được ẩn sau các con số trong một vectơ nhúng

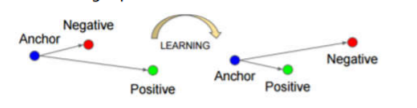
Những vector embedding sẽ làm sẽ làm đầu vào cho hàm loss function đánh giá khoảng cách giữa các vector. Để áp dụng triple loss, quá trình học được thực hiện với mỗi bộ ba mẫu học gồm, trong đó là hình ảnh của một người cụ thể (gọi là ảnh neo - anchor), là ảnh khác của cùng một người với ảnh (gọi là ảnh dương - positive) và là hình ảnh của bất kỳ một người khác (gọi là ảnh âm - negative). Mục tiêu ở đây là học mạng nơron (điều chỉnh trọng số mạng) sao cho phản hồi của mạng nơron với cặp mẫu là gần nhau hơn.

Hàm triple loss luôn lấy 3 bức ảnh làm input trong mọi trường hợp kì vọng:

d(A, P) < d(A, N) (1)

Để làm cho khoảng cách giữa vế trái và vế phải lớn hơn, ta sẽ cộng thêm vào vế trái một hệ số α không âm rất nhỏ. Khi đó (1) trở thành:

ǁf(A) – f(P) + α ≤ ǁf(A) – f(N)

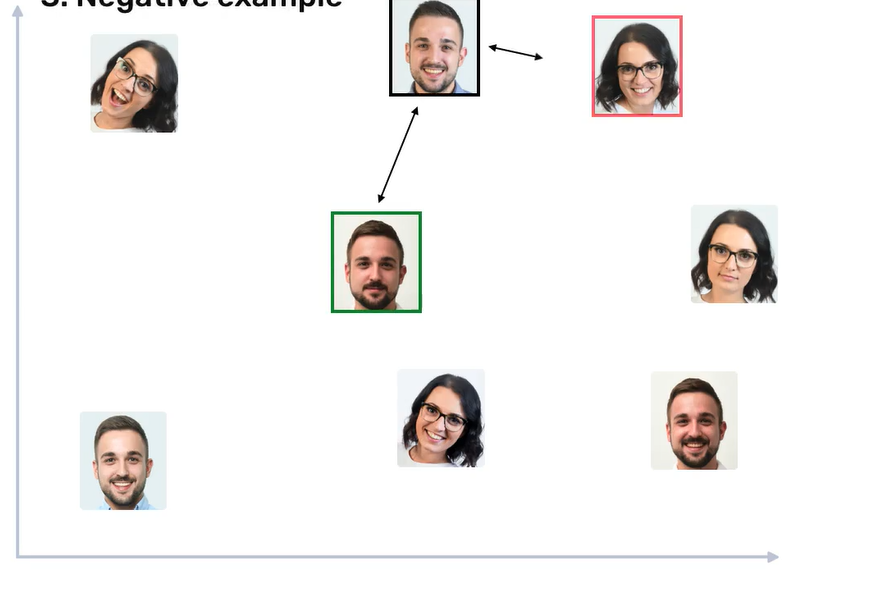


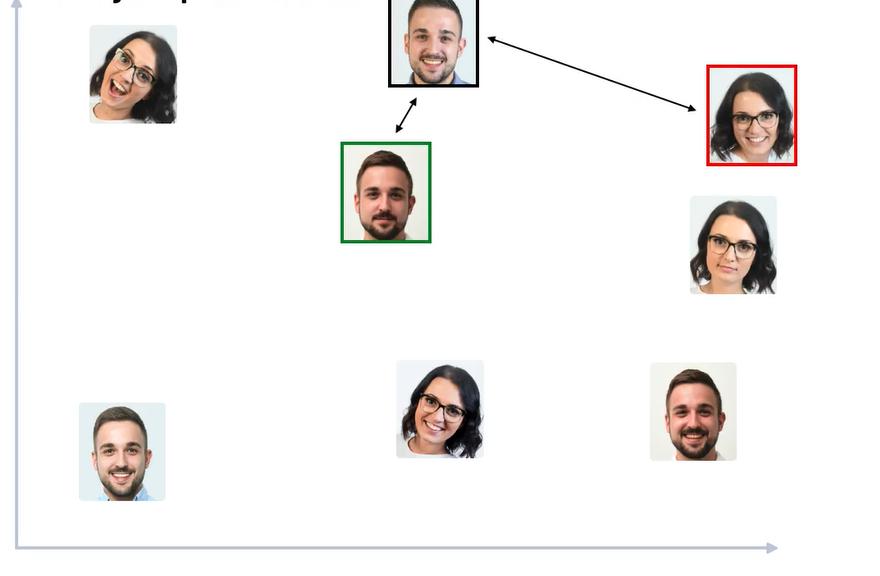
Như vậy khi áp dụng Triple loss vào các mô hình convolutional neural network ta có thể tạo ra các biểu diễn vector tốt nhất cho mỗi một bức ảnh. Những biểu diễn véctơ này sẽ phân biệt tốt các ảnh Negative rất giống ảnh Positive. Và đồng thời các bức ảnh thuộc cùng một label sẽ trở nên gần nhau hơn trong không gian chiếu Euclidean.

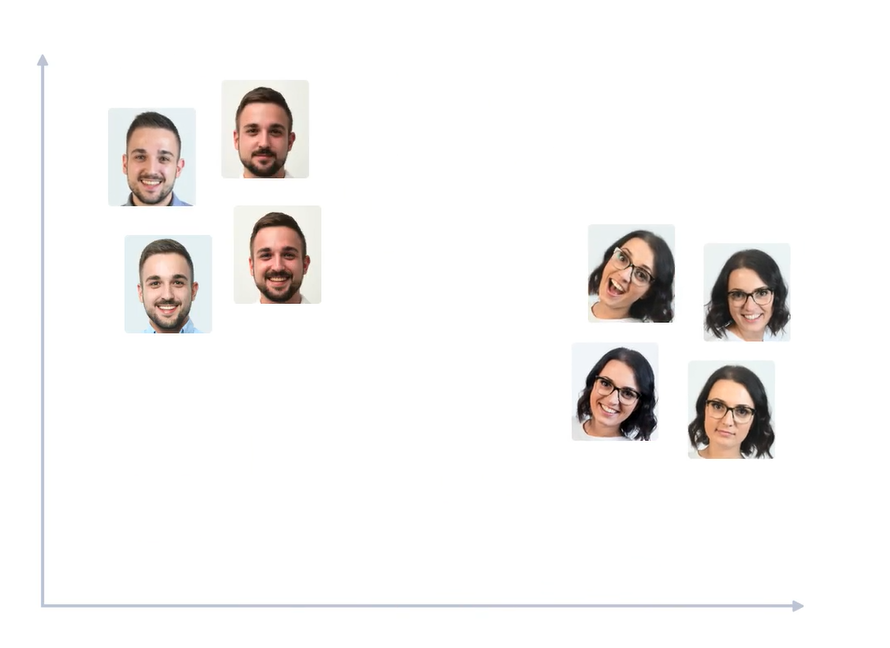
Mô tả chi tiết:

* Chọn ngẫu nhiên một hình ảnh neo (anchor)
* Chọn ngẫu nhiên một hình ảnh của cùng một người làm hình ảnh neo (positive) (ví dụ tích cực).
* Chọn ngẫu nhiên hình ảnh của một người khác với hình ảnh neo (negative) (ví dụ tiêu cực).
* Điều chỉnh các thông số mạng FaceNet để ví dụ tích cực gần với neo hơn ví dụ tiêu cực.

Chúng ta lặp lại các bước này cho đến khi không còn thay đổi gì nữa. Khi đó, tất cả các khuôn mặt của cùng một người gần nhau và xa người khác.

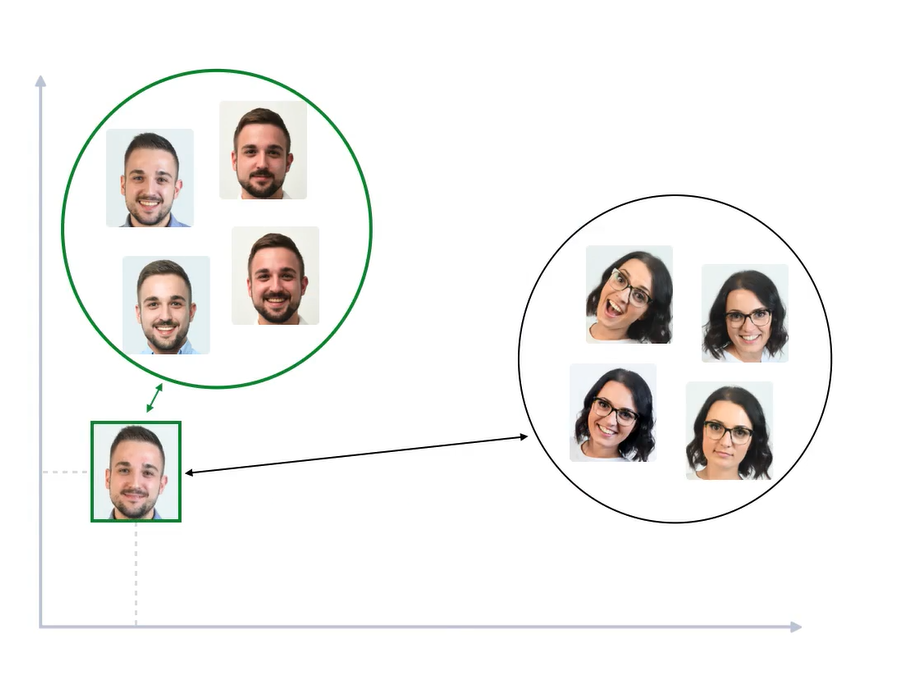






2.3.3 Nhận diện khuôn mặt

Cách để nhận ra một người là tính toán độ embedding của nó. Tính toán khoảng cách tới hình ảnh của những người đã được học. Nếu hình ảnh khuôn mặt đủ gần với người A thì ta nói hình ảnh này là của người A.

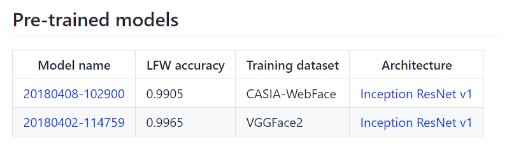


2.3.4 Kết quả đạt được:

Hầu hết chúng ta khi xây dựng một thuật toán nhận diện khuôn mặt sẽ không cần phải train lại mô hình facenet mà tận dụng lại các mô hình pretrain sẵn có. Bạn sẽ không cần phải tốn thời gian và công sức nếu không có đủ tài nguyên và dữ liệu. Đó cũng là lý do tôi cho rằng việc xây dựng mô hình nhận diện khuôn mặt ở thời điểm hiện tại rất dễ dàng.

Những mô hình pretrain được huấn luyện trên các dữ liệu lên tới hàng triệu ảnh. Do đó có khả năng mã hóa rất tốt các bức ảnh trên không gian 128 chiều. Việc còn lại của chúng ta là sử dụng lại mô hình, tính toán embedding véc tơ và huấn luyện embedding véc tơ bằng một classifier đơn giản để phân loại classes.

Trong đồ án này chúng tôi sử dụng pretrain model từ facenet repo – davidsandberg.



Kiến trúc mà tác giả sử dụng là Inception ResNetv1 trên 2 bộ dữ liệu là:

* CASIA-WebFace: Bộ dữ liệu bao gồm gần 500k ảnh được thu thập từ khoảng 10k người.
* VGGFace2: Bộ dữ liệu gồm khoảng 3 triệu ảnh được thu thập từ gần 9k người.

2.4 Phân tích thiết kế hệ thống.

2.4.1 Luồng hoạt động của hệ thống

sơ đồ khối

Ta có 2 giai đoạn chính như trên sơ đồ:

Giai đoạn 1: Huấn luyện dữ liệu cho hệ thống (Train Face Mask Detector)

Giai đoạn 2: Triển khai nhận diện và phân tích (Apply Face Mask Detector)

# 3. Kết quả

Sinh viên trình bày các kết quả đạt được, các chức năng đã triển khai/thử nghiệm, kết quả cụ thể khi thực hiện các chức năng. Nhóm SV chú ý mô tả các chi tiết về:

* dữ liệu đã sử dụng: nguồn gốc dữ liệu và cách thức thu thập; các tính chất của dữ liệu ví dụ như độ phân giải, dải giá trị,…; cách phân chia và kích thước tập huấn luyện/xác nhận/kiểm thử;
* các công cụ và framework đã dùng;
* các điều kiện tiến hành thực nghiệm: giá trị các tham số của các hàm API và giải thích, điều kiện của môi trường như ánh sáng, thời tiết,…;
* các độ đo (metrics) cụ thể dùng để đánh giá hiệu suất của giải pháp như độ chính xác (ví dụ đo bằng RMSE hoặc MAPE), tốc độ thực thi (ví dụ đo bằng FPS hoặc thời gian chạy), độ ổn định, tính bảo mật, khả năng mở rộng,...;
* quy trình và điều kiện kiểm thử hệ thống, và các kết quả đánh giá hệ thống theo các tiêu chí đã nêu.

# 4. Kết luận

Sinh viên đánh giá sản phẩm đồ án so với yêu cầu đặt ra và nêu kết luận về các kết quả đạt được. Nêu hướng phát triển nếu có thêm thời gian và kinh phí để triển khai đồ án.

# 5. Danh mục tài liệu tham khảo

Nhóm SV liệt kê các TLTK đã trích dẫn (cite) trong báo cáo tại đây.