**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

TRƯỜNG ĐẠI HỌC NGUYỄN TẤT THÀNH

KHOA CÔNG NGHỆ THÔNG TIN



KHÓA LUẬN TỐT NGHIỆP

TÊN ĐỀ TÀI

PHÁT TRIỂN HỆ THỐNG NHẬN DẠNG KHUÔN MẶT SỬ DỤNG OPENCV VÀ TENSORFLOW

**Giảng viên hướng dẫn:TH.S PHẠM ĐÌNH TÀI**

**Sinh viên thực hiện: NGÔ THỦY TIÊN**

**MSSV:2100003414**

**Khoá: 2021-2025**

**Ngành/ chuyên ngành:KHOA HỌC DỮ LIỆU**

**Tp HCM, tháng 12 năm 2024**

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

TRƯỜNG ĐẠI HỌC NGUYỄN TẤT THÀNH

KHOA CÔNG NGHỆ THÔNG TIN



KHÓA LUẬN TỐT NGHIỆP

TÊN ĐỀ TÀI

PHÁT TRIỂN HỆ THỐNG NHẬN DẠNG KHUÔN MẶT SỬ DỤNG OPENCV VÀ TENSORFLOW

**Giảng viên hướng dẫn:TH.S PHẠM ĐÌNH TÀI**

**Sinh viên thực hiện: NGÔ THỦY TIÊN**

**MSSV:2100003414**

**Khoá: 2021-2025**

**Ngành/ chuyên ngành:KHOA HỌC DỮ LIỆU**

**Tp HCM, tháng 12 năm 2024**

LỜI CẢM ƠN

Trước tiên, em xin gửi lời cảm ơn chân thành và sâu sắc nhất đến Ban Giám hiệu Trường Đại học Nguyễn Tất Thành, cùng các thầy cô trong khoa công nghệ thông tin đã tạo điều kiện và môi trường học tập lý tưởng, đồng thời hỗ trợ tôi hoàn thành chương trình học một cách tốt đẹp. Những kiến thức quý báu mà tôi được tiếp thu trong suốt thời gian học tập tại trường là nền tảng quan trọng để tôi thực hiện khóa luận tốt nghiệp này.

Đặc biệt, tôi xin bày tỏ lòng biết ơn sâu sắc đến Ths.Phạm Đình Tài, người đã tận tâm chỉ bảo, hướng dẫn và đồng hành cùng tôi trong suốt quá trình nghiên cứu và hoàn thành khóa luận. Những lời khuyên chân thành, những góp ý tận tình cùng sự động viên của thầy không chỉ giúp tôi vượt qua những khó khăn trong quá trình thực hiện đề tài mà còn giúp tôi trau dồi thêm nhiều kiến thức và kỹ năng quý giá.

Bên cạnh đó, tôi cũng xin dành lời cảm ơn đến các thầy cô, các cán bộ, nhân viên trong khoa khoa công nghệ thông tin đã hỗ trợ, cung cấp tài liệu và giúp đỡ tôi trong suốt thời gian thực hiện nghiên cứu. Sự nhiệt tình và tận tâm của các thầy cô đã giúp tôi rất nhiều trong việc hoàn thành tốt đề tài của mình.

Ngoài ra, tôi muốn gửi lời cảm ơn sâu sắc đến gia đình, những người luôn là nguồn động viên, khích lệ lớn lao về tinh thần và vật chất. Sự quan tâm, yêu thương và hỗ trợ vô điều kiện của bố mẹ, anh chị em trong gia đình đã là động lực để tôi vượt qua mọi khó khăn trong quá trình học tập và nghiên cứu.

Tôi nhận thức rằng, mặc dù đã rất cố gắng nhưng khóa luận của tôi khó tránh khỏi những thiếu sót. Vì vậy, tôi rất mong nhận được những ý kiến đóng góp từ các thầy cô và bạn bè để có thể hoàn thiện tốt hơn trong tương lai.

*TPHCM, Ngày …… tháng …… năm 2024*

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

1. Hình thức

1. Nội dung

1. Kết luận

*TPHCM, Ngày …… tháng …… năm 2024*

NHẬN XÉT CỦA GIẢNG VIÊN PHẢN BIỆN

1. Hình thức

1. Nội dung

1. Kết luận

*TPHCM, Ngày …… tháng …… năm 2024*

**MỤC LỤC**

[LỜI CẢM ƠN 1](#_Toc185548971)

[MỞ ĐẦU 12](#_Toc185548974)

[CHƯƠNG 1: GIỚI THIỆU CHUNG 13](#_Toc185548975)

[1.1 Giới thiệu về sự phát triển của công nghệ nhận diện khuôn mặt 13](#_Toc185548976)

[1.2 Lý do chọn đề tài 15](#_Toc185548977)

[1.3 Mục tiêu nghiên cứu 16](#_Toc185548978)

[1.4 Phạm vi nghiên cứu và giới hạn 16](#_Toc185548979)

[CHƯƠNG 2: CƠ SỞ LÝ THUYẾT 17](#_Toc185548980)

[2.1 Tổng quan về nhận dạng khuôn mặt 17](#_Toc185548981)

[2.2 Giới thiệu thư viện OpenCV 21](#_Toc185548982)

[2.3 Xử lý ảnh với open cv 22](#_Toc185548983)

[2.4 Giới thiệu về TensorFlow trong học sâu 47](#_Toc185548984)

[2.4.1 Giới thiệu 47](#_Toc185548985)

[2.4.2 TensorFlow - Nền tảng toán học 48](#_Toc185548986)

[Véc tơ 48](#_Toc185548987)

[2.4.3 Mạng nơ-ron tích chập 49](#_Toc185548988)

[2.4.4 Convolutional Neural Networks (CNNs) trong nhận diện khuôn mặt 50](#_Toc185548989)

[CNNs trong Nhận Diện Khuôn Mặt 51](#_Toc185548990)

[2.5 Giới thiệu về YOLO 51](#_Toc185548991)

[2.5.1 Tổng quan 51](#_Toc185548992)

[2.5.2 YOLOv5 52](#_Toc185548993)

[2.5.3 YOLO11 52](#_Toc185548994)

[Các tính năng chính 53](#_Toc185548995)

[2.5.1 So sánh YOLO với các mô hình phát hiện đối tượng khác 54](#_Toc185548996)

[CHƯƠNG 3: XÂY DỰNG MÔ HÌNH 56](#_Toc185548997)

[3.1 Tạo dữ liệu 56](#_Toc185548998)

[3.1.1 Tạo dữ liệu yolo 56](#_Toc185548999)

[3.1.2 Tạo dữ liệu CNN 57](#_Toc185549000)

[3.2 Xử lý hình ảnh 59](#_Toc185549001)

[3.2.1 YOLO 59](#_Toc185549002)

[3.2.2 CNNs 59](#_Toc185549003)

[4.3 Huấn luyện mô hình 60](#_Toc185549004)

[4.3.1 Yolov5 60](#_Toc185549005)

[3.3.2 Yolo11 61](#_Toc185549006)

[3.3.3.CNN 61](#_Toc185549007)

[3.4 Kết quả 63](#_Toc185549008)

[3.4.1. YOLOV5 63](#_Toc185549009)

[4.4.2 YOLO11 65](#_Toc185549010)

[3.4.3 CNN 67](#_Toc185549011)

[CHƯƠNG 4: KẾT QUẢ THỰC NGHIỆM 69](#_Toc185549012)

[4.1 Yolo 69](#_Toc185549013)

[4.1.1 Yolo v5 69](#_Toc185549014)

[4.1.2 Yolo 11 69](#_Toc185549015)

[4.2 CNNs 70](#_Toc185549016)

[4.3 Hệ thống chấm công 70](#_Toc185549017)

[CHƯƠNG 5: TỔNG KẾT 73](#_Toc185549018)

[DANH MỤC TÀI LIỆU THAM KHẢO 77](#_Toc185549019)

**DANH MỤC BẢNG**

[Bảng 1.1.Bảng so sánh giữa các phương pháp truyền thống và hiện đại trong nhận diện khuôn mặt 18](#_Toc185548960)

**DANH MỤC HÌNH**

[Hình 2.1. Gary Bradsky và Vadim Pisarevsky 21](#_Toc185548855)

[Hình 2.2. Guido van Rossum 22](#_Toc185548856)

[Hình 2.3. Bản dịch 23](#_Toc185548857)

[Hình 2.4. Sự xoay vòng 24](#_Toc185548858)

[Hình 2.5. Biến đổi afin 24](#_Toc185548859)

[Hình 2.6. Biến đổi quang điểm 25](#_Toc185548860)

[Hình 2.7. Ngưỡng đơn giản 26](#_Toc185548861)

[Hình 2.8. Ngưỡng đơn giản 27](#_Toc185548862)

[Hình 2.9. Sự nhị phân hóa của Otsu 27](#_Toc185548863)

[Hình 2.10. Tích chập 2D (Lọc hình ảnh) 28](#_Toc185548864)

[Hình 2.11. Làm mờ trung bình 29](#_Toc185548865)

[Hình 2.12. Làm mờ Gauss 29](#_Toc185548867)

[Hình 2.13. Làm mờ trung vị 30](#_Toc185548869)

[Hình 2.14. Lọc song phương 30](#_Toc185548871)

[Hình 2.15. Xói mòn 31](#_Toc185548872)

[Hình 2.16. Sự giãn nở 31](#_Toc185548873)

[Hình 2.17. Mở đầu 31](#_Toc185548874)

[Hình 2.18. Kết thúc 32](#_Toc185548875)

[Hình 2.19. Độ dốc hình thái 32](#_Toc185548876)

[Hình 2.20. Mũ chóp cao 32](#_Toc185548877)

[Hình 2.21. Mũ đen 33](#_Toc185548878)

[Hình 2.24. cường độ Gradient của hình ảnh 34](#_Toc185548879)

[Hình 2.25. Ngưỡng trễ 35](#_Toc185548880)

[Hình 2.26. Phát hiện cạnh Canny 35](#_Toc185548881)

[Hình 2.27. Kim tự tháp Gaussian 36](#_Toc185548882)

[Hình 2.28. Hình kim tự tháp 3 cấp 36](#_Toc185548883)

[Hình 2.29. Ba mức của một mức Laplacian 37](#_Toc185548884)

[Hình 2.30. Trộn hình ảnh bằng cách sử dụng Pyramids 37](#_Toc185548885)

[Hình 2.31. cv.TM\_CCOEFF 38](#_Toc185548886)

[Hình 2.32. cv.TM\_CCOEFF\_NORMED 38](#_Toc185548887)

[Hình 2.33. cv.TM\_CCORR 38](#_Toc185548888)

[Hình 2.34. cv.TM\_CCORR\_NORMED 39](#_Toc185548889)

[Hình 2.35. cv.TM\_SQDIFF 39](#_Toc185548890)

[Hình 2.36. cv.TM\_SQDIFF\_NORMED 39](#_Toc185548891)

[Hình 2.37. Mẫu phù hợp với nhiều đối tượng 40](#_Toc185548892)

[Hình 2.38. Biến đổi đường Hough 40](#_Toc185548893)

[Hình 2.39. Biến đổi Hough hoạt động 41](#_Toc185548894)

[Hình 2.40. Bộ tích lũy 41](#_Toc185548895)

[Hình 2.41. Biến đổi Hough trong OpenCV 42](#_Toc185548896)

[Hình 2.42. So sánh phép biến đổi Hough và phép biến đổi Hough xác suất 42](#_Toc185548897)

[Hình 2.43.  Kết quả sử dụng hàm cv.HoughLinesP() 43](#_Toc185548898)

[Hình 2.44. Biến đổi vòng tròn Hough 43](#_Toc185548899)

[Hình 2.45. Phân đoạn hình ảnh với thuật toán Watershed 44](#_Toc185548900)

[Hình 2.46. Kết quả nền 44](#_Toc185548901)

[Hình 2.47. Hình ảnh ngưỡng 45](#_Toc185548902)

[Hình 2.48. Bản đồ màu JET 45](#_Toc185548903)

[Hình 2.49. Kết quả phân đoạn 46](#_Toc185548904)

[Hình 2.50. Trích xuất tiền cảnh tương tác bằng thuật toán GrabCut 46](#_Toc185548905)

[Hình 2.51.Minh họa hoạt động thuật toán GrabCut 47](#_Toc185548906)

[Hình 2.52. TensorFlow 47](#_Toc185548907)

[Hình 2.53. Véc tơ 48](#_Toc185548908)

[Hình 2.54. Mô hình vector 48](#_Toc185548909)

[Hình 2.55. Ma trận 49](#_Toc185548910)

[Hình 2.56. Sơ đồ biểu diễn việc tạo ra các trường cục bộ tương ứng 50](#_Toc185548911)

[Hình 2.57. Hiệu suất yolov5u 52](#_Toc185548912)

[Hình 2.58. Số liệu hiệu suất yolov5u 52](#_Toc185548913)

[Hình 2.59. Hiệu suất yolo 11 53](#_Toc185548914)

[Hình 2.60. Số liệu hiệu suất yolo11 53](#_Toc185548915)

[Hình 3.1 Hàm thu thập dữ liệu yolo 56](#_Toc185548916)

[Hình 3.2 Kết quả thu thập 56](#_Toc185548917)

[Hình 3.3 Cấu trúc tập data yolo 57](#_Toc185548918)

[Hình 3.4 Hàm thu thập dữ liệu CNN 57](#_Toc185548919)

[Hình 3.5 Kết quả thu thập 57](#_Toc185548920)

[Hình 3.6 Cấu trúc file data 58](#_Toc185548921)

[Hình 3.7 Hàm chuyển đổi thành file csv 58](#_Toc185548922)

[Hình 3.8 Kết quả khi chuyển thành file csv 58](#_Toc185548923)

[Hình 3.9 Đánh nhãn dữ liệu 59](#_Toc185548924)

[Hình 3.10 Cắt dữ liệu 59](#_Toc185548925)

[Hình 3.11 Encoding label 59](#_Toc185548926)

[Hình 3.12 Chia tập train và test 59](#_Toc185548927)

[Hình 3.13 Cường hóa dữ liệu 60](#_Toc185548928)

[Hình 3.14 Tải yolov5 60](#_Toc185548929)

[Hình 3.15 Train yolov5 60](#_Toc185548930)

[Hình 3.16 File data.yaml 60](#_Toc185548931)

[Hình 3.17 Tải ultralytics 61](#_Toc185548932)

[Hình 3.18 Gọi yolo11 61](#_Toc185548933)

[Hình 3.19 Train 61](#_Toc185548934)

[Hình 3.20 File data.yaml 61](#_Toc185548935)

[Hình 3.21 Thư viện 61](#_Toc185548936)

[Hình 3.22 Mô hình 62](#_Toc185548937)

[Hình 3.23 Loss & learning\_rate và acc 62](#_Toc185548938)

[Hình 3.24 Checkpoint & EarlyStopping và tensorboard 62](#_Toc185548939)

[Hình 3.25 Train 63](#_Toc185548940)

[Hình 3.26 Kết quả 63](#_Toc185548941)

[Hình 3.27 Confusion Matrix 63](#_Toc185548942)

[Hình 3.28 F1-Confidence Curve 64](#_Toc185548943)

[Hình 3.29 results 64](#_Toc185548944)

[Hình 3.30 Kết quả 65](#_Toc185548945)

[Hình 3.31 Confusion Matrix 65](#_Toc185548946)

[Hình 3.32 F1-Confidence Curve 66](#_Toc185548947)

[Hình 3.33 results 66](#_Toc185548948)

[Hình 3.34 accuracy 67](#_Toc185548949)

[Hình 3.35 Model accuracy 67](#_Toc185548950)

[Hình 3.36 Model loss 67](#_Toc185548951)

[Hình 3.37 Hiệu suất của 3 mô hình 68](#_Toc185548952)

[Hình 4.1 Nhận diện thời gian thực yolov5 69](#_Toc185548953)

[Hình 4.2 Nhận diện thời gian thực yolo11 69](#_Toc185548954)

[Hình 4.3 Nhận diện thời gian thực CNN 70](#_Toc185548955)

[Hình 4.4 Giao diện 70](#_Toc185548956)

[Hình 4.5 Nhận diện check-int 71](#_Toc185548957)

[Hình 4.6 Nhận diện check-out 71](#_Toc185548958)

[Hình 4.7 File csv lưu trữ thời gian chấm công 72](#_Toc185548959)

MỞ ĐẦU

Trong thời đại công nghệ 4.0, sự bùng nổ của trí tuệ nhân tạo và học máy đã và đang mang đến những đột phá quan trọng trong nhiều lĩnh vực của đời sống và công nghiệp. Một trong những ứng dụng nổi bật của công nghệ này là hệ thống nhận dạng khuôn mặt. Không chỉ giúp tăng cường an ninh, nhận dạng khuôn mặt còn mang lại nhiều tiện ích trong quản lý nhân sự, đặc biệt là trong việc chấm công.

Truyền thống, việc chấm công cho nhân viên chủ yếu dựa vào các phương pháp như ký tên, quẹt thẻ hoặc sử dụng vân tay. Mặc dù có hiệu quả nhất định, nhưng các phương pháp này đều có những hạn chế như tốn thời gian, dễ bị gian lận và không đảm bảo độ chính xác tuyệt đối. Trong khi đó, nhận dạng khuôn mặt với ưu điểm nhanh chóng, tiện lợi và độ chính xác cao, đã trở thành một giải pháp tối ưu, khắc phục được hầu hết các nhược điểm của phương pháp truyền thống.

Hệ thống nhận dạng khuôn mặt sử dụng OpenCV và TensorFlow áp dụng vào chấm công là một bước tiến quan trọng trong việc cải thiện quy trình quản lý nhân sự. OpenCV, một thư viện mã nguồn mở mạnh mẽ dành cho xử lý ảnh và thị giác máy tính, kết hợp với TensorFlow, một nền tảng mã nguồn mở cho học máy, đã tạo nên một hệ thống nhận dạng khuôn mặt hiệu quả và đáng tin cậy.

Trong phần mở đầu này, chúng ta sẽ cùng khám phá những lợi ích vượt trội của việc áp dụng công nghệ nhận dạng khuôn mặt vào chấm công. Chúng ta sẽ tìm hiểu về cách thức hoạt động của hệ thống và tại sao việc phát triển hệ thống này lại cần thiết và cấp bách trong bối cảnh hiện nay. Việc triển khai hệ thống nhận dạng khuôn mặt không chỉ giúp nâng cao hiệu quả công việc, mà còn góp phần thúc đẩy sự phát triển của công nghệ trong đời sống hàng ngày. Qua đó, chúng ta sẽ thấy rõ vai trò quan trọng của trí tuệ nhân tạo và học máy trong việc nâng cao chất lượng cuộc sống và công việc.

CHƯƠNG 1: GIỚI THIỆU CHUNG

* 1. Giới thiệu về sự phát triển của công nghệ nhận diện khuôn mặt

Công nghệ nhận diện khuôn mặt, một trong những thành tựu vượt bậc của trí tuệ nhân tạo, đã trải qua một quá trình phát triển mạnh mẽ và ngày càng trở nên phổ biến trong nhiều lĩnh vực ứng dụng, đặc biệt trong khoa học dữ liệu. Nhận diện khuôn mặt là quá trình xác định và xác minh danh tính của một cá nhân dựa trên các đặc điểm sinh trắc học của khuôn mặt. Các nghiên cứu và ứng dụng trong lĩnh vực này không chỉ giúp cải thiện các hệ thống bảo mật mà còn đóng góp quan trọng vào sự phát triển của các hệ thống thông minh, phục vụ cho nhiều mục đích khác nhau từ giám sát, an ninh, đến phân tích hành vi người dùng.

Nhận diện khuôn mặt có thể được coi là một nhánh con của lĩnh vực nhận diện sinh trắc học, nơi các hệ thống sử dụng những đặc điểm sinh lý của con người để nhận dạng. Trong những năm 1990, công nghệ nhận diện khuôn mặt bắt đầu được nghiên cứu, nhưng chỉ trong hai thập kỷ qua, với sự bùng nổ của các kỹ thuật học máy và học sâu (deep learning), công nghệ này mới thực sự đạt được bước đột phá đáng kể về độ chính xác và tốc độ.

Ban đầu, các phương pháp nhận diện khuôn mặt chủ yếu sử dụng kỹ thuật phân tích hình học (geometrical analysis), tập trung vào việc đo đạc và phân tích các đặc điểm cố định của khuôn mặt, chẳng hạn như khoảng cách giữa các mắt, mũi, và miệng. Tuy nhiên, những phương pháp này thường gặp phải các hạn chế lớn trong việc xử lý các biến thể phức tạp như góc nhìn, điều kiện ánh sáng kém, và sự thay đổi trong hình dạng khuôn mặt. Mặc dù những phương pháp này có thể đạt được độ chính xác khá cao trong các điều kiện lý tưởng, nhưng chúng không thể đáp ứng được yêu cầu trong các tình huống thực tế, nơi có sự thay đổi liên tục về môi trường.

Đến những năm 2010, công nghệ nhận diện khuôn mặt đã có một bước tiến vượt bậc nhờ sự ra đời của các mô hình học sâu (deep learning), đặc biệt là các mạng nơ-ron tích chập (CNNs). Với khả năng tự học từ một lượng dữ liệu khổng lồ và phát hiện các đặc trưng phức tạp của khuôn mặt, các mô hình học sâu đã giúp cải thiện đáng kể hiệu suất của các hệ thống nhận diện khuôn mặt. Các mô hình như FaceNet, VGG-Face và OpenFace đã chứng minh khả năng nhận diện khuôn mặt với độ chính xác gần như tuyệt đối trong các điều kiện khác nhau.

Ngày nay, công nghệ nhận diện khuôn mặt đã được tích hợp vào hàng loạt ứng dụng và dịch vụ, từ các hệ thống an ninh công cộng, các phương tiện giao thông thông minh, cho đến các ứng dụng trong lĩnh vực y tế và marketing. Sự phát triển này một phần nhờ vào sự phát triển mạnh mẽ của các công cụ phần mềm như OpenCV, TensorFlow, và PyTorch, cũng như các cải tiến trong phần cứng máy tính, bao gồm GPU và các mạch tích hợp chuyên dụng cho AI (ASIC).

Trong lĩnh vực khoa học dữ liệu, công nghệ nhận diện khuôn mặt không chỉ giới hạn trong các ứng dụng bảo mật và giám sát mà còn có một vai trò quan trọng trong các nghiên cứu và phân tích dữ liệu liên quan đến hành vi và nhận diện. Việc sử dụng các thuật toán học máy để phân tích và xử lý dữ liệu khuôn mặt mở ra nhiều cơ hội mới trong việc phát triển các ứng dụng thông minh phục vụ cho nghiên cứu và dự đoán hành vi người dùng.

Một trong những ứng dụng quan trọng của công nghệ nhận diện khuôn mặt trong khoa học dữ liệu là trong phân tích cảm xúc và hành vi. Việc nhận diện khuôn mặt kết hợp với các thuật toán học sâu có thể giúp phát hiện các biểu cảm cảm xúc của con người, từ đó xây dựng các mô hình phân tích hành vi, ứng dụng trong việc nghiên cứu tâm lý học, tâm thần học, và hành vi người tiêu dùng. Các hệ thống phân tích cảm xúc từ khuôn mặt có thể sử dụng các tính năng như biểu cảm mắt, miệng, và các cơ mặt khác để xác định trạng thái cảm xúc của một người, từ đó đưa ra các dự đoán về tâm trạng hoặc sự thay đổi trong hành vi.

Dữ liệu khuôn mặt cũng đóng vai trò quan trọng trong các nghiên cứu về nhận dạng dân số. Ví dụ, các hệ thống nhận diện khuôn mặt có thể được sử dụng để phân tích các đặc điểm khuôn mặt trong các cuộc khảo sát dân số, giúp đưa ra các dự đoán về xu hướng nhân khẩu học và thị trường tiêu dùng.

Một trong những thách thức lớn đối với việc ứng dụng nhận diện khuôn mặt trong khoa học dữ liệu là vấn đề bảo mật và quyền riêng tư. Dữ liệu khuôn mặt là một dạng dữ liệu nhạy cảm, và việc thu thập, lưu trữ và sử dụng nó đụng phải nhiều vấn đề về pháp lý và đạo đức. Vì vậy, các nghiên cứu khoa học dữ liệu liên quan đến nhận diện khuôn mặt không chỉ phải tuân thủ các tiêu chuẩn bảo mật và quyền riêng tư, mà còn cần phải cân nhắc kỹ lưỡng đến các yếu tố xã hội và đạo đức trong việc thu thập và phân tích dữ liệu sinh trắc học.

Bên cạnh đó, sự phát triển của các kỹ thuật phân tích hình ảnh đa chiều và học máy tiên tiến, kết hợp với các nguồn dữ liệu lớn, sẽ mở ra cơ hội để sử dụng công nghệ nhận diện khuôn mặt trong các nghiên cứu về hành vi người tiêu dùng, quảng cáo, y học, và nhiều lĩnh vực khác. Công nghệ này không chỉ giúp tự động hóa các quy trình và nâng cao độ chính xác trong việc phân tích dữ liệu, mà còn giúp tạo ra những hệ thống có thể tự học và thích ứng với các thay đổi trong dữ liệu theo thời gian.

Tuy nhiên, công nghệ nhận diện khuôn mặt cũng sẽ tiếp tục đối mặt với những thách thức liên quan đến bảo mật và quyền riêng tư, yêu cầu các nhà nghiên cứu và phát triển phải tiếp tục cải tiến và ứng dụng các phương pháp bảo vệ dữ liệu mạnh mẽ. Do đó, công nghệ nhận diện khuôn mặt trong khoa học dữ liệu sẽ không chỉ là một công cụ mạnh mẽ, mà còn phải được sử dụng một cách có trách nhiệm, phù hợp với các chuẩn mực đạo đức và pháp lý hiện hành.

1.2 Lý do chọn đề tài

Trong bối cảnh công nghệ thông tin và trí tuệ nhân tạo ngày càng phát triển mạnh mẽ, việc ứng dụng công nghệ nhận diện khuôn mặt vào các hệ thống quản lý như chấm công là một xu hướng tất yếu. Đây là lý do chính khiến chúng tôi lựa chọn đề tài nghiên cứu phát triển hệ thống chấm công thông minh sử dụng công nghệ nhận diện khuôn mặt.

Chấm công thông minh với công nghệ nhận diện khuôn mặt sẽ giúp doanh nghiệp và tổ chức giảm thiểu rủi ro gian lận thời gian làm việc, nâng cao tính minh bạch và tiết kiệm chi phí vận hành. Đặc biệt, hệ thống này có thể dễ dàng tích hợp với các hệ thống quản lý nhân sự khác, giúp tạo ra một môi trường làm việc thông minh, hiệu quả và hiện đại.

1.3 Mục tiêu nghiên cứu

Xây dựng hệ thống nhận diện khuôn mặt tự động cho quá trình chấm công. Mục tiêu chính của nghiên cứu này là xây dựng và phát triển một hệ thống nhận diện khuôn mặt thông minh cho phép tự động hóa quá trình chấm công trong các tổ chức, doanh nghiệp. Hệ thống sẽ sử dụng công nghệ nhận diện khuôn mặt để xác định danh tính của nhân viên khi họ vào hoặc ra khỏi khu vực làm việc.

Một mục tiêu quan trọng của nghiên cứu là cải thiện độ chính xác của hệ thống nhận diện khuôn mặt. Để đảm bảo rằng hệ thống có thể nhận diện chính xác khuôn mặt của nhân viên trong mọi tình huống, chúng tôi sẽ sử dụng các thuật toán học sâu tiên tiến như mạng nơ-ron tích chập (CNN) và YOLO để huấn luyện mô hình nhận diện khuôn mặt. Hệ thống sẽ được tối ưu hóa để hoạt động hiệu quả trong các điều kiện ánh sáng khác nhau và trong môi trường thực tế, nơi có thể có sự thay đổi về vị trí, góc độ và trạng thái khuôn mặt của người sử dụng.Tăng cường bảo mật và tính năng xác thực.

1.4 Phạm vi nghiên cứu và giới hạn

Phạm vi nghiên cứu, nghiên cứu sẽ áp dụng các công nghệ nhận diện khuôn mặt hiện đại như OpenCV, TensorFlow và YOLO. OpenCV sẽ được sử dụng để phát hiện và xử lý ảnh khuôn mặt, trong khi TensorFlow sẽ giúp huấn luyện mô hình học máy để cải thiện độ chính xác nhận diện. YOLO, với khả năng nhận diện nhanh chóng trong thời gian thực, em sẽ huấn luyện để so sánh với mô hình CNNs của em

Giới hạn nghiên cứu, mặc dù hệ thống sẽ được thiết kế để hoạt động trong các điều kiện ánh sáng và môi trường nhất định, nhưng sẽ có một số giới hạn.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1 Tổng quan về nhận dạng khuôn mặt

Các kỹ thuật và mô hình nhận dạng khuôn mặt truyền thống và hiện đại: nhận diện khuôn mặt là quá trình tự động xác định danh tính một cá nhân dựa trên các đặc điểm sinh lý của khuôn mặt. Ban đầu, nhận diện khuôn mặt được thực hiện bằng các phương pháp đơn giản như so sánh hình ảnh khuôn mặt với các mẫu đã lưu trữ trong cơ sở dữ liệu.

Dưới đây là bảng so sánh giữa các phương pháp truyền thống và hiện đại trong nhận diện khuôn mặt:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Phương pháp | Ưu điểm | | Nhược điểm | | Ứng dụng |
| Phương pháp đặc trưng hình học | - Dễ hiểu, đơn giản để triển khai.  - Đảm bảo tính toán nhanh chóng.  - Không yêu cầu phần cứng mạnh. | | - Khó khăn trong việc xử lý các thay đổi về góc nhìn, ánh sáng.  - Không thể nhận diện tốt khi khuôn mặt có thay đổi theo thời gian.  - Độ chính xác thấp trong các điều kiện phức tạp. | | Thường được áp dụng trong các hệ thống nhận diện khuôn mặt đơn giản, nơi điều kiện môi trường ổn định. |
| Phương pháp dựa trên mẫu (Template Matching) | Đơn giản, dễ triển khai và nhanh chóng trong các hệ thống nhỏ.  - Không cần đào tạo mô hình phức tạp. | | - Không hiệu quả khi có sự thay đổi góc nhìn hay ánh sáng.  - Cần một cơ sở dữ liệu mẫu lớn và tính toán mất nhiều thời gian khi mở rộng. | | Dùng trong các ứng dụng nhận diện khuôn mặt trong các điều kiện ổn định (ví dụ, máy chấm công) nhưng gặp khó khăn trong môi trường phức tạp. |
| Mạng Nơ-ron Tích Chập (CNNs) | - Tự động học và phát hiện đặc trưng phức tạp của khuôn mặt.  - Độ chính xác cao ngay cả trong điều kiện ánh sáng yếu và góc nhìn thay đổi.  - Có thể học từ dữ liệu không có gắn nhãn. | | - Yêu cầu lượng dữ liệu huấn luyện lớn.  - Cần phần cứng tính toán mạnh mẽ (GPU/TPU) để huấn luyện mô hình.  - Phức tạp trong việc triển khai và tối ưu hóa. | | Áp dụng rộng rãi trong các ứng dụng nhận diện khuôn mặt trong môi trường thực tế, như hệ thống giám sát an ninh, nhận diện trong các ứng dụng di động, và các hệ thống nhận diện người dùng. |
| Học máy không giám sát và bán giám sát | - Giảm sự phụ thuộc vào dữ liệu gắn nhãn.  - Tiết kiệm thời gian và công sức trong việc thu thập và xử lý dữ liệu.  - Tăng hiệu quả trong các ứng dụng yêu cầu tốc độ và linh hoạt. | | - Độ chính xác thấp hơn so với các mô hình có giám sát.  - Có thể gặp khó khăn trong việc phát hiện các đặc trưng phức tạp khi không đủ dữ liệu huấn luyện. | | Phù hợp với các ứng dụng cần xử lý nhanh và hiệu quả với dữ liệu ít hoặc không có nhãn, như nhận diện khuôn mặt trong các ứng dụng thông minh, chấm công tự động, và các hệ thống an ninh. |
| Eigenfaces (Phân tích thành phần chính - PCA) | - Tiết kiệm bộ nhớ và thời gian tính toán so với các phương pháp phức tạp hơn.  - Tốt cho việc nhận diện khuôn mặt trong một không gian đặc trưng thấp hơn. | | - Không có khả năng xử lý những thay đổi lớn về góc nhìn, ánh sáng hoặc các biểu cảm khuôn mặt.  - Kết quả không chính xác khi có sự thay đổi lớn về hình dạng khuôn mặt. | | Sử dụng trong các hệ thống nhận diện khuôn mặt căn bản và không đòi hỏi yêu cầu cao về môi trường. |
| YOLO (You Only Look Once) | | - Phát hiện và nhận diện khuôn mặt nhanh chóng và chính xác trong thời gian thực.  - Có khả năng nhận diện nhiều khuôn mặt cùng lúc trong một khung hình.  - Phù hợp với các ứng dụng thời gian thực yêu cầu độ chính xác cao. | - Yêu cầu phần cứng mạnh để đạt hiệu suất tối ưu.  - Cần lượng dữ liệu huấn luyện lớn và đa dạng.  - Phức tạp trong việc triển khai và tối ưu hóa. | Được sử dụng rộng rãi trong các hệ thống giám sát an ninh, nhận diện khuôn mặt trong video, và các ứng dụng thời gian thực khác. | |

Bảng 1.1.Bảng so sánh giữa các phương pháp truyền thống và hiện đại trong nhận diện khuôn mặt

* Nhận xét chung:

**Phương pháp đặc trưng hình học**

Phương pháp đặc trưng hình học là một trong những phương pháp đầu tiên được sử dụng trong nhận diện khuôn mặt. Nó dựa trên các đặc trưng hình học cơ bản của khuôn mặt, như khoảng cách giữa các điểm đặc trưng (mắt, mũi, miệng) và tỷ lệ giữa chúng.. Các đặc trưng hình học có thể được tính toán nhanh chóng mà không yêu cầu phần cứng mạnh, điều này giúp giảm chi phí và tài nguyên cần thiết cho hệ thống.

**Phương pháp dựa trên mẫu (Template Matching)**

Template Matching là một phương pháp đơn giản dựa trên việc so sánh trực tiếp hình ảnh khuôn mặt cần nhận diện với các mẫu đã lưu trữ trong cơ sở dữ liệu. Phương pháp này không yêu cầu đào tạo mô hình phức tạp, do đó dễ dàng triển khai trong các hệ thống nhỏ.

**Mạng Nơ-ron Tích Chập (CNNs)**

Mạng Nơ-ron Tích Chập (Convolutional Neural Networks - CNNs) là một bước tiến lớn trong lĩnh vực nhận diện khuôn mặt. CNNs có khả năng tự động học và phát hiện các đặc trưng phức tạp từ hình ảnh, điều mà các phương pháp truyền thống khó thực hiện. Nhờ vào cấu trúc nhiều lớp và khả năng học từ dữ liệu lớn, CNNs có thể xử lý tốt các điều kiện ánh sáng yếu, thay đổi góc nhìn và các biểu cảm khác nhau trên khuôn mặt.

**Học máy không giám sát và bán giám sát**

Các phương pháp học máy không giám sát và bán giám sát đang ngày càng trở nên phổ biến trong lĩnh vực nhận diện khuôn mặt. Chúng giúp giảm sự phụ thuộc vào dữ liệu gắn nhãn, tiết kiệm thời gian và công sức trong việc thu thập và xử lý dữ liệu.

**Eigenfaces (Phân tích thành phần chính - PCA)**

Eigenfaces là một phương pháp dựa trên kỹ thuật phân tích thành phần chính (Principal Component Analysis - PCA) để giảm chiều dữ liệu và trích xuất các đặc trưng quan trọng của khuôn mặt.

**YOLO (You Only Look Once)**

YOLO (You Only Look Once) là một trong những phương pháp tiên tiến nhất hiện nay, nổi bật với khả năng phát hiện và nhận diện khuôn mặt nhanh chóng và chính xác trong thời gian thực. YOLO sử dụng một mạng nơ-ron sâu duy nhất để xử lý toàn bộ hình ảnh và nhận diện nhiều khuôn mặt cùng lúc trong một khung hình, giúp giảm thiểu thời gian xử lý và tăng hiệu suất.

2.2 Giới thiệu thư viện OpenCV

OpenCV được **Gary Bradsky** khởi xướng tại Intel vào năm 1999 và bản phát hành đầu tiên ra mắt vào năm 2000.

Hình 2.1. Gary Bradsky và Vadim Pisarevsky

**Vadim Pisarevsky** đã tham gia cùng Gary Bradsky để quản lý nhóm phần mềm OpenCV của Intel tại Nga.OpenCV hiện hỗ trợ vô số thuật toán liên quan đến thị giác máy tính và học máy và đang mở rộng từng ngày.

OpenCV hỗ trợ nhiều ngôn ngữ lập trình khác nhau như C++, Python, Java, v.v. và có sẵn trên nhiều nền tảng khác nhau bao gồm Windows, Linux, OS X, Android và iOS. Các giao diện cho hoạt động GPU tốc độ cao dựa trên CUDA và OpenCL cũng đang được phát triển tích cực.

Python là một ngôn ngữ lập trình mục đích chung do **Guido van Rossum** khởi xướng và nhanh chóng trở nên rất phổ biến, chủ yếu là vì tính đơn giản và khả năng đọc mã của nó. Nó cho phép lập trình viên thể hiện ý tưởng trong ít dòng mã hơn mà không làm giảm khả năng đọc.



Hình 2.2. Guido van Rossum

2.3 Xử lý ảnh với open cv

* **Thay đổi không gian màu**

Trong OpenCV, có hơn 150 phương pháp chuyển đổi không gian màu. Tuy nhiên, chúng ta sẽ tập trung vào hai phương pháp được sử dụng phổ biến nhất: chuyển đổi từ BGR sang màu xám và từ BGR sang HSV.

Để chuyển đổi không gian màu, chúng ta sử dụng hàm cv.cvtColor(input\_image, flag) trong đó flag xác định loại chuyển đổi. Cụ thể:

* cv.COLOR\_BGR2GRAY:BGR sang màu xám
* cv.COLOR\_BGR2HSV :BGR sang HSV

**Lưu ý**

* Đối với không gian màu HSV, phạm vi của giá trị Hue là [0,179], phạm vi của Saturation là [0,255], và phạm vi của Value là [0,255].

**Làm thế nào để tìm giá trị HSV để theo dõi?**

Sử dụng cùng một hàm cv.cvtColor(),bạn có thể lấy [H-10, 100, 100] và [H+10, 255, 255] làm giới hạn dưới và giới hạn trên tương ứng.

**Biến đổi hình học của hình ảnh**

* Hàm cv.warpAffine  sử dụng ma trận chuyển đổi 2x3
* Hàm cv.warpPerspective sử dụng ma trận chuyển đổi 3x3

**Tỷ lệ**

Scaling chỉ là thay đổi kích thước của hình ảnh. ,hàm cv.resize() cho mục đích này.. Các phương pháp nội suy khác nhau được sử dụng. Các phương pháp nội suy được ưa thích là:

* cv.INTER\_AREA để thu nhỏ
* cv.INTER\_CUBIC (chậm)
* cv.INTER\_LINEAR để phóng to
* cv.INTER\_LINEAR được sử dụng cho tất cả các mục đích thay đổi kích thước

Lưu ý

Đối số thứ ba của hàm cv.warpAffine() là kích thước của hình ảnh đầu ra, có dạng (width, height) .width = số cột và height = số hàng.

**Bản dịch**

Sự tịnh tiến là sự dịch chuyển vị trí của một vật thể. Nếu bạn biết sự dịch chuyển theo hướng (x,y) và để nó như vậy(tx,ty), bạn có thể tạo ma trận chuyển đổi M như sau:

M=

Bạn có thể đưa make vào một mảng Numpy có kiểu np.float32 và truyền nó vào hàm cv.warpAffine()



Hình 2.3. Bản dịch

**Sự xoay vòng**

Xoay hình ảnh theo một gócsốđược thực hiện bằng ma trận biến đổi có dạng

M=

Nhưng OpenCV cung cấp phép quay có tỷ lệ với tâm quay có thể điều chỉnh để bạn có thể quay ở bất kỳ vị trí nào bạn thích. Ma trận biến đổi đã sửa đổi được đưa ra bởi

Ở đâu:

Để tìm ma trận chuyển đổi này, OpenCV cung cấp một hàm, cv.getRotationMatrix2D



Hình 2.4. Sự xoay vòng

**Biến đổi afin**

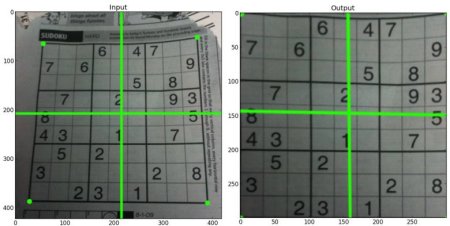
Trong phép biến đổi affine, tất cả các đường thẳng song song trong ảnh gốc vẫn sẽ song song trong ảnh đầu ra..Sau đó, cv.getAffineTransform sẽ tạo một ma trận 2x3 sẽ được truyền đến cv.warpAffine .



Hình 2.5. Biến đổi afin

**Biến đổi quan điểm**

Đối với phép biến đổi phối cảnh, bạn cần một ma trận biến đổi 3x3. Các đường thẳng sẽ vẫn thẳng ngay cả sau khi biến đổi. ma trận biến đổi bằng hàm cv.getPerspectiveTransform  và áp dụng cv.warpPerspective với ma trận biến đổi 3x3 này.



Hình 2.6. Biến đổi quang điểm

* **Ngưỡng hình ảnh**

**Ngưỡng đơn giản**

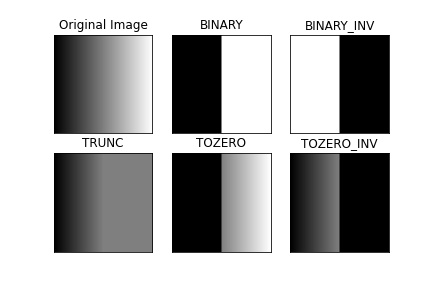
Quá trình này rất đơn giản: cùng một giá trị ngưỡng được áp dụng cho mỗi pixel. Nếu giá trị pixel nhỏ hơn hoặc bằng ngưỡng, nó được đặt thành 0; ngược lại, nó được đặt thành giá trị tối đa. Để thực hiện ngưỡng hóa, ta sử dụng hàm cv.threshold.

* Đối số đầu tiên của hàm là ảnh nguồn, phải là ảnh thang độ xám.
* Đối số thứ hai là giá trị ngưỡng dùng để phân loại các giá trị pixel.
* Đối số thứ ba là giá trị tối đa gán cho các giá trị pixel vượt quá ngưỡng.

Ngưỡng cơ bản, kiểu cv.THRESH\_BINARY. Các loại ngưỡng đơn giản khác bao gồm:

* cv.THRESH\_BINARY
* cv.THRESH\_BINARY\_INV
* cv.THRESH\_TRUNC
* cv.THRESH\_TOZERO
* cv.THRESH\_TOZERO\_INV

Phương pháp này trả về hai giá trị: giá trị ngưỡng đã sử dụng và hình ảnh đã được ngưỡng hóa.



Hình 2.7. Ngưỡng đơn giản

**Ngưỡng thích ứng**

Thuật toán ngưỡng thích ứng xác định giá trị ngưỡng cho mỗi pixel dựa trên một vùng nhỏ xung quanh nó.

Hàm cv.adaptiveThreshold còn sử dụng ba tham số đầu vào bổ sung:

**Phương pháp thích ứng:**

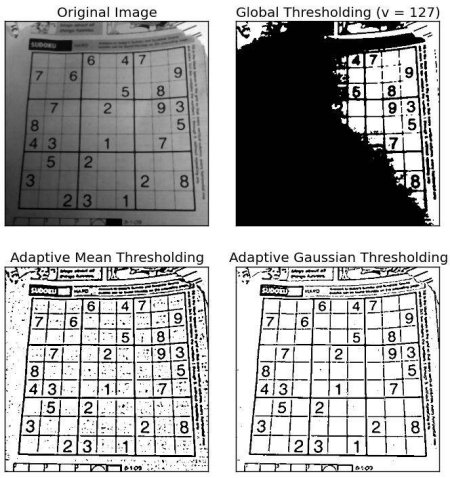
Xác định cách tính giá trị ngưỡng:

* cv.ADAPTIVE\_THRESH\_MEAN\_C: Giá trị ngưỡng là trung bình của các pixel trong vùng lân cận trừ đi một hằng số C.
* cv.ADAPTIVE\_THRESH\_GAUSSIAN\_C: Giá trị ngưỡng là tổng trọng số Gaussian của các pixel lân cận trừ đi một hằng số C.

**blockSize:** Xác định kích thước của vùng lân cận xung quanh mỗi pixel.

**C:** Là hằng số được trừ đi từ giá trị trung bình hoặc tổng trọng số của các pixel trong vùng lân cận.

Sử dụng các tham số này, ngưỡng thích ứng có thể xử lý tốt hơn các hình ảnh có sự biến đổi độ sáng mạnh mẽ.

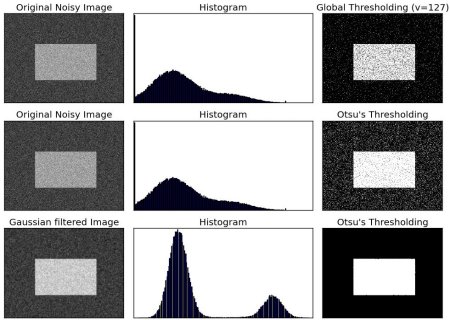


Hình 2.8. Ngưỡng đơn giản

**Sự nhị phân hóa của Otsu**

Phương pháp Otsu hoạt động theo nguyên tắc này, tự động xác định giá trị ngưỡng toàn cục tối ưu từ biểu đồ histogram của hình ảnh.

Để áp dụng phương pháp Otsu, chúng ta sử dụng hàm cv.threshold() với cờ cv.THRESH\_OTSU được thêm vào.



Hình 2.9. Sự nhị phân hóa của Otsu

**Binarization của Otsu hoạt động như thế nào?**

Thuật toán của Otsu cố gắng tìm giá trị ngưỡng (t) giúp giảm thiểu **phương sai trong lớp có trọng số** được đưa ra bởi mối quan hệ:

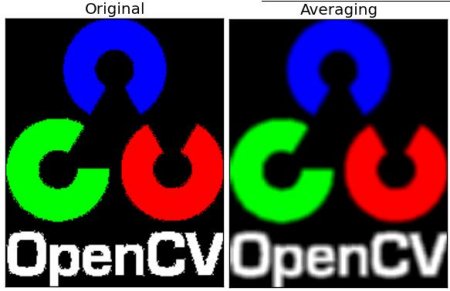
Ở đâu

Trên thực tế, nó tìm ra giá trị t nằm giữa hai đỉnh sao cho phương sai của cả hai lớp là tối thiểu

* **Làm mịn hình ảnh**

**Tích chập 2D (Lọc hình ảnh)**

Hàm cv.filter2D() để tích chập một hạt nhân với một hình ảnh. Ví dụ về 1 hạt nhận 5x5 ở bên dưới



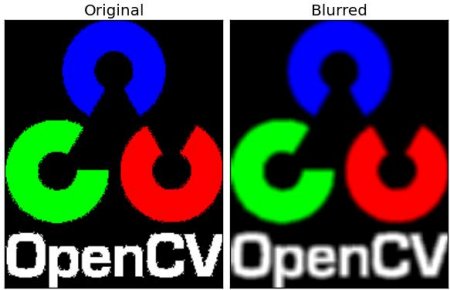
Hình 2.10. Tích chập 2D (Lọc hình ảnh)

**Làm mờ hình ảnh (Làm mịn hình ảnh)**

Làm mờ hình ảnh được thực hiện bằng cách tích chập hình ảnh với một hạt nhân bộ lọc thông thấp.

**1. Trung bình**

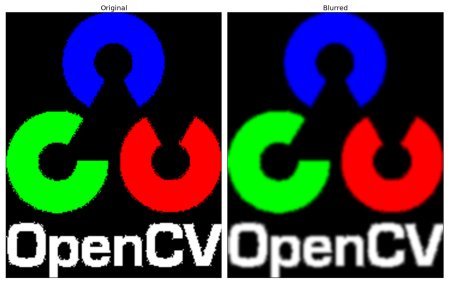
Thực hiện bằng cách tích chập hình ảnh với một bộ lọc hộp chuẩn hóa. Quá trình này đơn giản lấy giá trị trung bình của tất cả các điểm ảnh trong vùng hạt nhân và thay thế giá trị của phần tử trung tâm bằng giá trị trung bình đó. thực hiện điều này bằng cách sử dụng hàm cv.blur() hoặc cv.boxFilter()



Hình 2.11. Làm mờ trung bình

**2. Làm mờ Gauss**

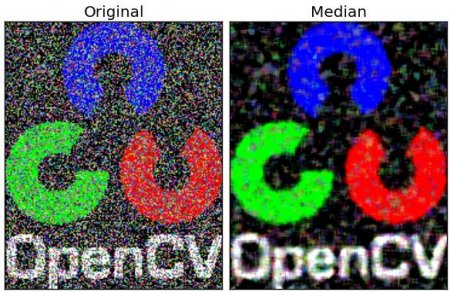
Phương pháp này sử dụng một hạt nhân Gaussian, được thực hiện bằng hàm cv.GaussianBlur().



Hình 2.12. Làm mờ Gauss

**3. Làm mờ trung vị**

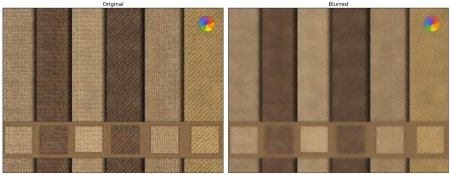
Hàm cv.medianBlur() thay thế giá trị của pixel trung tâm bằng giá trị trung vị của tất cả các pixel trong vùng hạt nhân.



Hình 2.13. Làm mờ trung vị

1. **Lọc song phương**

Hàm cv.bilateralFilter() hiệu quả trong việc loại bỏ nhiễu mà vẫn giữ các cạnh sắc nét, mặc dù nó chậm hơn các bộ lọc khác.



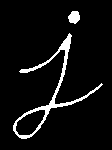
Hình 2.14. Lọc song phương

* **Biến đổi hình thái**

Biến đổi hình thái là một số phép toán đơn giản dựa trên hình dạng ảnh. Phép toán này thường được thực hiện trên ảnh nhị phân.

**1. Xói mòn**

Quá trình này hoạt động bằng cách trượt hạt nhân qua hình ảnh, tương tự như phép tích chập 2D



Hình 2.15. Xói mòn

**2. Sự giãn nở**

Quá trình giãn nở hoàn toàn ngược lại với xói mòn. Ở đây, một pixel sẽ có giá trị '1' nếu ít nhất một pixel dưới hạt nhân là '1'. Điều này làm tăng vùng trắng trong hình ảnh hoặc làm lớn hơn kích thước của đối tượng tiền cảnh.



Hình 2.16. Sự giãn nở

**3. Mở đầu**

Opening chỉ là một tên gọi khác của **quá trình xói mòn theo sau là sự giãn nở** .Sử dụng hàm cv.morphologyEx()



Hình 2.17. Mở đầu

**4. Kết thúc**

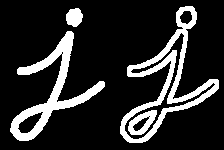
Đóng là ngược lại với Mở, **Giãn theo sau là Xói mòn** . Nó hữu ích trong việc đóng các lỗ nhỏ bên trong các vật thể ở tiền cảnh hoặc các điểm đen nhỏ trên vật thể.



Hình 2.18. Kết thúc

**5. Độ dốc hình thái**

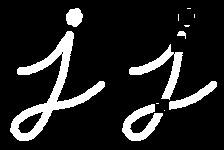
Đây là sự khác biệt giữa sự giãn nở và xói mòn của hình ảnh.



Hình 2.19. Độ dốc hình thái

**6. Mũ chóp cao**

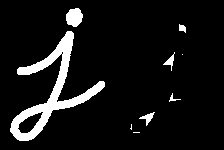
Đây là sự khác biệt giữa hình ảnh đầu vào và hình ảnh mở.



Hình 2.20. Mũ chóp cao

**7. Mũ đen**

Đó là sự khác biệt giữa phần đóng của hình ảnh đầu vào và hình ảnh đầu vào.



Hình 2.21. Mũ đen

* **Độ dốc hình ảnh**

OpenCV cung cấp ba loại bộ lọc gradient hoặc bộ lọc thông cao, Sobel, Scharr và Laplacian.

**1. Đạo hàm Sobel và Scharr**

Toán tử Sobel là sự kết hợp giữa phép làm mịn Gauss và phép tính vi phân, giúp nó chống nhiễu hiệu quả hơn. Bạn có thể chọn hướng của đạo hàm cần tính, theo chiều dọc hoặc chiều ngang, thông qua các đối số yorder và xorder.

**2. Đạo hàm Laplacian**

Nó tính toán Laplacian của hình ảnh được đưa ra bởi mối quan hệ,Δsrc= trong đó mỗi đạo hàm được tìm thấy bằng cách sử dụng đạo hàm Sobel. Nếu k size = 1, thì hạt nhân sau được sử dụng để lọc:

* **Phát hiện cạnh Canny**

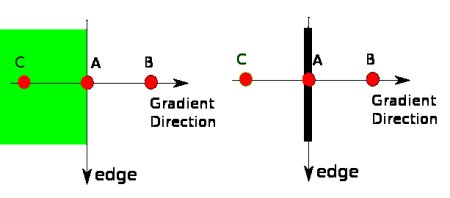
Canny Edge Detection là một thuật toán phát hiện cạnh nổi tiếng được phát triển bởi John F. Canny vào năm 1986

**Giảm nhiễu**: Do phát hiện cạnh rất nhạy cảm với nhiễu trong hình ảnh, bước đầu tiên là loại bỏ nhiễu bằng bộ lọc Gaussian 5x5.

**Tính toán cường độ Gradient**: Sau khi làm mịn, hình ảnh được lọc bằng bộ lọc Sobel theo cả hai hướng (ngang và dọc) để tính toán đạo hàm bậc nhất theo hướng ngang (Gx) và hướng dọc (Gy).

Gradient luôn vuông góc với các cạnh và được làm tròn thành một trong bốn hướng: dọc, ngang và hai hướng chéo.

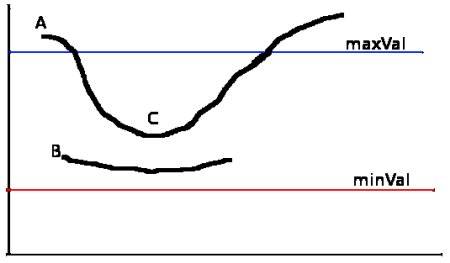
Sau khi tính độ lớn và hướng gradient, một lần quét toàn bộ ảnh được thực hiện để loại bỏ các pixel không phải là giá trị cực đại cục bộ theo hướng của gradient. Điều này giúp chỉ giữ lại các điểm ảnh thực sự tạo thành các cạnh.



Hình 2.24. cường độ Gradient của hình ảnh

1. **Ngưỡng trễ**

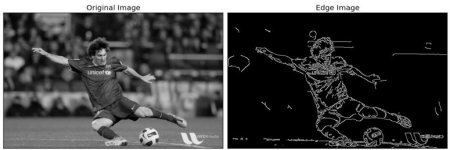
Hai giá trị ngưỡng, minVal và maxVal, được sử dụng để xác định điều này. Các cạnh có độ dốc lớn hơn maxVal được coi là "cạnh chắc chắn" và sẽ được giữ lại, trong khi những cạnh có độ dốc nhỏ hơn minVal sẽ bị loại bỏ. Những cạnh nằm giữa hai ngưỡng này sẽ được phân loại dựa trên sự kết nối của chúng với các pixel "cạnh chắc chắn". Nếu chúng kết nối với các pixel này, chúng sẽ được xem là một phần của cạnh, ngược lại, chúng sẽ bị loại bỏ.



Hình 2.25. Ngưỡng trễ

**Phát hiện cạnh Canny trong OpenCV**

Sử dụng hàm  cv.Canny() .. Nếu là True, nó sẽ sử dụng phương trình được đề cập . Theo mặc định, giá trị này là False.



Hình 2.26. Phát hiện cạnh Canny

* **Kim tự tháp hình ảnh**

Bộ các hình ảnh Kim tự tháp hình ảnh, vì khi các hình ảnh có độ phân giải khác nhau được xếp chồng lên nhau, với hình ảnh có độ phân giải cao nhất ở dưới cùng và thấp nhất ở trên cùng, trông giống như một kim tự tháp.

Có hai loại kim tự tháp hình ảnh chính:

**Kim tự tháp Gaussian:**

Ở mỗi cấp độ cao hơn trong kim tự tháp, hình ảnh được tạo ra bằng cách giảm độ phân giải của hình ảnh ở cấp thấp hơn bằng cách loại bỏ các hàng và cột liên tiếp. Mỗi pixel ở cấp độ cao hơn được tính bằng cách lấy trọng số từ 5 pixel ở cấp độ thấp hơn theo cách Gaussian.

**Kim tự tháp Laplacian:**

Đây là biến thể của kim tự tháp Gaussian. Mỗi cấp trong kim tự tháp Laplacian được tạo ra từ sự chênh lệch giữa hình ảnh của cấp hiện tại và cấp kế tiếp trong kim tự tháp Gaussian.

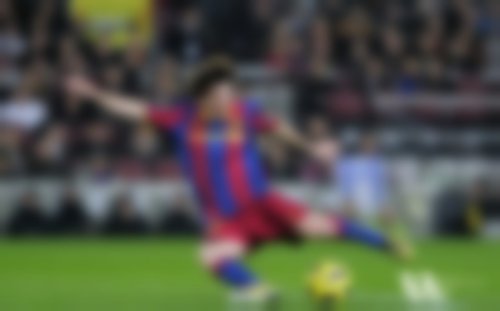
Hàm cv.pyrDown() và cv.pyrUp() trong OpenCV. Hàm pyrDown giúp giảm độ phân giải của hình ảnh, trong khi pyrUp giúp tăng độ phân giải.



Hình 2.27. Kim tự tháp Gaussian

Có thể duyệt xuống kim tự tháp hình ảnh bằng hàm cv.pyrUp() .

Hãy nhớ rằng, higher\_reso2 không bằng higher\_reso, vì khi bạn giảm độ phân giải, bạn sẽ mất thông tin.



Hình 2.28. Hình kim tự tháp 3 cấp

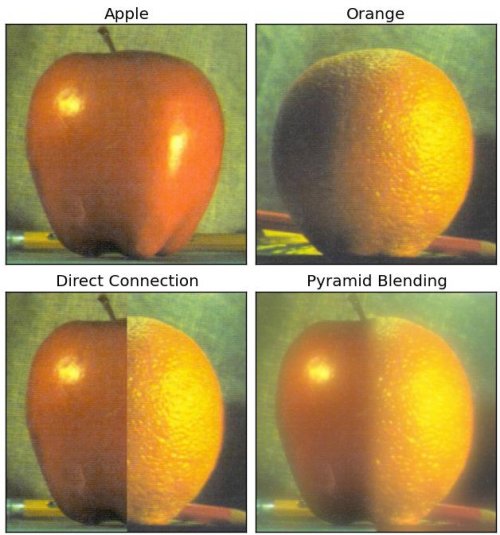
Kim tự tháp Laplacian được hình thành từ Kim tự tháp Gaussian. Không có chức năng độc quyền nào cho chức năng đó. Ảnh kim tự tháp Laplacian chỉ giống như ảnh cạnh.



Hình 2.29. Ba mức của một mức Laplacian

**Trộn hình ảnh bằng cách sử dụng Pyramids**

Một ứng dụng của Pyramids là Image Blending. Việc pha trộn hình ảnh với Pyramids giúp bạn pha trộn liền mạch mà không để lại nhiều dữ liệu trong hình ảnh.



Hình 2.30. Trộn hình ảnh bằng cách sử dụng Pyramids

* **Phù hợp mẫu**

Template Matching là một kỹ thuật trong xử lý ảnh được sử dụng để tìm kiếm và xác định vị trí của một hình ảnh mẫu trong một hình ảnh lớn hơn. Hàm cv.matchTemplate() thực hiện việc này.

**So khớp mẫu trong OpenCV**

* cv.TM\_CCOEFF



Hình 2.31. cv.TM\_CCOEFF

* cv.TM\_CCOEFF\_NORMED



Hình 2.32. cv.TM\_CCOEFF\_NORMED

* cv.TM\_CCORR



Hình 2.33. cv.TM\_CCORR

* cv.TM\_CCORR\_NORMED



Hình 2.34. cv.TM\_CCORR\_NORMED

* cv.TM\_SQDIFF



Hình 2.35. cv.TM\_SQDIFF

* cv.TM\_SQDIFF\_NORMED



Hình 2.36. cv.TM\_SQDIFF\_NORMED

**Mẫu phù hợp với nhiều đối tượng**

Để tìm kiếm một đối tượng có nhiều lần xuất hiện, cv.minMaxLoc() sẽ cung cấp cho bạn tất cả các vị trí.

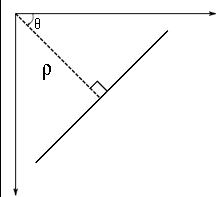


Hình 2.37. Mẫu phù hợp với nhiều đối tượng

* **Biến đổi đường Hough**

Biến đổi Hough là một phương pháp mạnh mẽ trong nhận diện hình học, đặc biệt là phát hiện các đường thẳng trong ảnh. Nó có thể phát hiện hình dạng ngay cả khi nó bị vỡ hoặc bị bóp méo một chút.

Một đường thẳng có thể được biểu diễn như y = mx+c hoặc ở dạng tham số, như Ở đây p là khoảng cách vuông góc từ gốc đến đường thẳng,



Hình 2.38. Biến đổi đường Hough

**Mảng tích lũy Hough (Accumulator Array):**

* Tạo một mảng 2D
* **θ:** góc sẽ có giá trị từ 0° đến 180
* **ρ:** khoảng cách có thể thay đổi từ 0 đến độ dài đường chéo của hình ảnh

Mảng tích lũy sẽ có số hàng tương ứng với các giá trị ρ, và số cột tương ứng với các giá trị θ.

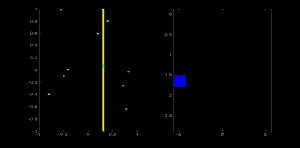
**Quá trình bỏ phiếu (Voting Process):**

* Mỗi điểm trên đường thẳng trong ảnh sẽ bỏ phiếu vào mảng tích lũy.
* Đối với mỗi điểm (x, y) trên đường thẳng, ta tính toán các giá trị **ρ** và **θ** từ phương trình đường thẳng và tăng giá trị tại vị trí (ρ, θ) trong mảng tích lũy.
* Quá trình này sẽ được lặp lại cho tất cả các điểm trên đường thẳng.

**Tìm kiếm trong mảng tích lũy:**

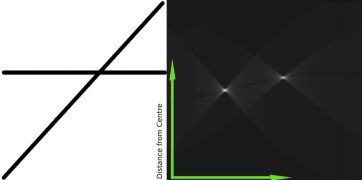
* Sau khi quá trình bỏ phiếu hoàn tất, mảng tích lũy sẽ chứa các giá trị phản ánh sự xuất hiện của các đường thẳng trong ảnh.
* Để phát hiện các đường thẳng, ta tìm kiếm các giá trị cực đại trong mảng tích lũy.

Khi mảng tích lũy có giá trị cực đại tại vị trí (ρ, θ), điều này có nghĩa là một đường thẳng trong ảnh được biểu diễn bởi tham số này. Bạn có thể vẽ đường thẳng này lên ảnh bằng cách sử dụng các giá trị (ρ, θ) để xác định phương trình đường thẳng.



Hình 2.39. Biến đổi Hough hoạt động

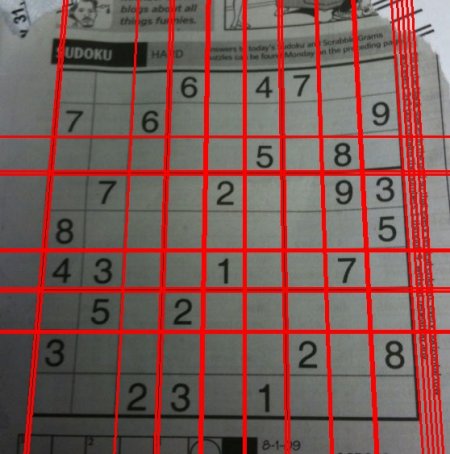
Đây là cách biến đổi hough hoạt động đối với các đường.



Hình 2.40. Bộ tích lũy

**Biến đổi Hough trong OpenCV**

Hàm [cv.HoughLines()](https://docs.opencv.org/4.x/dd/d1a/group__imgproc__feature.html#ga46b4e588934f6c8dfd509cc6e0e4545a) . Nó chỉ trả về một mảng các giá trị :math:(rho, theta)`.ρ được đo bằng pixel và số được đo bằng radian. Tham số đầu tiên, hình ảnh đầu vào phải là hình ảnh nhị phân,. Tham số thứ hai và thứ ba là ρ và số độ chính xác tương ứng. Đối số thứ tư là ngưỡng



Hình 2.41. Biến đổi Hough trong OpenCV

**Biến đổi Hough xác suất**

Phép biến đổi Hough xác suất là một phép tối ưu hóa của phép biến đổi Hough mà chúng ta đã thấy. Nó không tính đến tất cả các điểm. Thay vào đó, nó chỉ tính đến một tập hợp con ngẫu nhiên các điểm đủ để phát hiện đường thẳng. Chúng ta chỉ cần giảm ngưỡng.

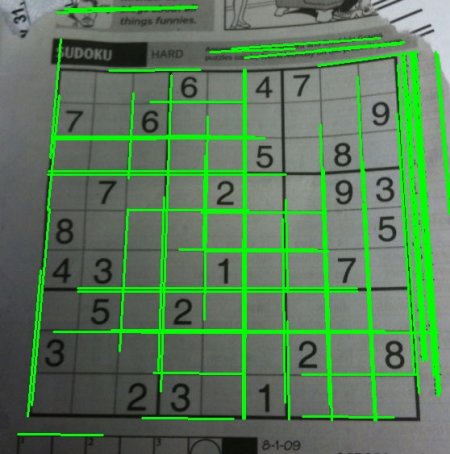


Hình 2.42. So sánh phép biến đổi Hough và phép biến đổi Hough xác suất

Hàm được sử dụng là [cv.HoughLinesP()](https://docs.opencv.org/4.x/dd/d1a/group__imgproc__feature.html" \l "ga8618180a5948286384e3b7ca02f6feeb" \o "Tìm các đoạn thẳng trong ảnh nhị phân bằng cách sử dụng phép biến đổi Hough xác suất.) . Nó có hai đối số mới.

* **minLineLength** - Độ dài tối thiểu của dòng
* **maxLineGap** - Khoảng cách tối đa được phép giữa các đoạn thẳng để coi chúng như một đoạn thẳng duy nhất.

Điều tuyệt vời nhất là nó trả về trực tiếp hai điểm cuối của các đường thẳng



Hình 2.43.  Kết quả sử dụng hàm [cv.HoughLinesP()](https://docs.opencv.org/4.x/dd/d1a/group__imgproc__feature.html#ga8618180a5948286384e3b7ca02f6feeb)

* **Biến đổi vòng tròn Hough**

Một vòng tròn được biểu diễn về mặt toán học như.Ở đây là tâm của vòng tròn, và r là bán kính của hình tròn. OpenCV sử dụng phương pháp phức tạp hơn, phương pháp Gradient Hough sử dụng thông tin gradient của các cạnh.

Hàm được sử dụng là [cv.HoughCircles()](https://docs.opencv.org/4.x/dd/d1a/group__imgproc__feature.html" \l "ga47849c3be0d0406ad3ca45db65a25d2d" \o "Tìm hình tròn trong ảnh thang độ xám bằng cách sử dụng phép biến đổi Hough.)



Hình 2.44. Biến đổi vòng tròn Hough

* **Phân đoạn hình ảnh với thuật toán Watershed**

Thuật toán phân vùng lưu vực (watershed) trong xử lý ảnh là một phương pháp mạnh mẽ để phân tách các đối tượng trong hình ảnh. Nó được hình dung như một quá trình "lấp đầy thung lũng" trong một bề mặt địa hình, nơi các thung lũng được lấp đầy bằng nước, và khi nước từ các thung lũng khác nhau gặp nhau

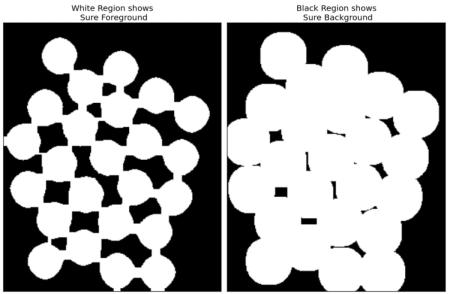
**Mã số**

Dưới đây chúng ta sẽ xem một ví dụ về cách sử dụng Biến đổi khoảng cách cùng với phân vùng để phân đoạn các đối tượng tiếp xúc với nhau.



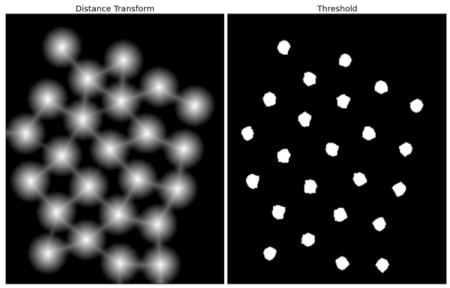
Hình 2.45. Phân đoạn hình ảnh với thuật toán Watershed

Để xử lý nhiễu và phân đoạn các vật thể trong hình ảnh, chúng ta có thể sử dụng một số phép toán hình thái học cơ bản như **mở**, **đóng**, **xói mòn**, và **giãn nở**. Đây là những công cụ mạnh mẽ trong xử lý ảnh để làm sạch hình ảnh, loại bỏ nhiễu và phân tách các vật thể.



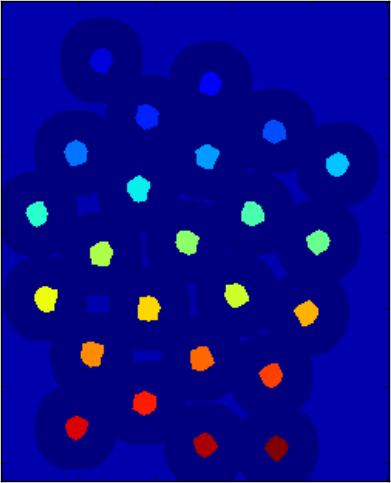
Hình 2.46. Kết quả nền

Thuật toán Watershed sẽ giúp chúng ta phân tách các đối tượng.Trong quy trình này, **vùng không xác định** (unknown regions) thường nằm quanh các ranh giới của các đối tượng, nơi tiền cảnh và nền gặp nhau, hoặc các đối tượng chạm vào nhau. Thuật toán Watershed sẽ giúp phân đoạn những vùng này bằng cách "lấp đầy" các vùng thung lũng (vùng không xác định) và xác định rõ ràng các đối tượng



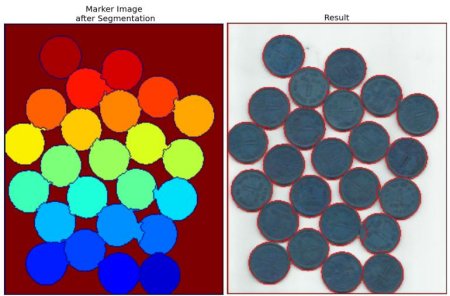
Hình 2.47. Hình ảnh ngưỡng

Sử dụng hàm **cv.connectedComponents()** để gán nhãn cho các vùng trong ảnh.phải đánh dấu các vùng nền (background) bằng một số nguyên khác, không phải là 0, và chỉ giữ giá trị 0 cho các vùng không xác định.



Hình 2.48. Bản đồ màu JET

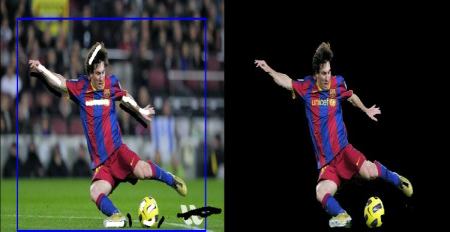
Vùng ranh giới sẽ được đánh dấu bằng -1.



Hình 2.49. Kết quả phân đoạn

* **Trích xuất tiền cảnh tương tác bằng thuật toán GrabCut**

Thuật toán **GrabCut** là một phương pháp phân đoạn hình ảnh được sử dụng rộng rãi, đặc biệt là trong các ứng dụng yêu cầu phân tách tiền cảnh (foreground) và hậu cảnh (background) trong ảnh, với sự tương tác tối thiểu từ người dùng.



Hình 2.50. Trích xuất tiền cảnh tương tác bằng thuật toán GrabCut

**Điều gì xảy ra ở phía sau:**

Khởi tạo:

* + Người dùng chỉ định một hình chữ nhật bao quanh đối tượng.
  + Dán nhãn cứng cho các vùng chắc chắn là tiền cảnh và hậu cảnh.

Mô hình hóa phân phối màu sắc với GMM:

* GMM học phân phối màu sắc của tiền cảnh và hậu cảnh.

Xây dựng đồ thị:

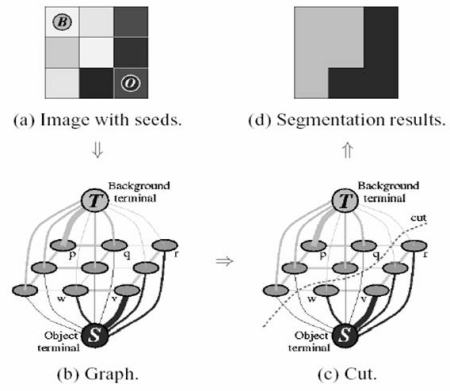
* Đồ thị được tạo ra, với các nút là các pixel và các trọng số được xác định theo độ tương đồng màu sắc.

Áp dụng mincut:

* MinCut phân chia đồ thị thành hai phần: tiền cảnh và hậu cảnh.

Lặp lại:

* Quá trình được lặp lại cho đến khi phân đoạn hội tụ.

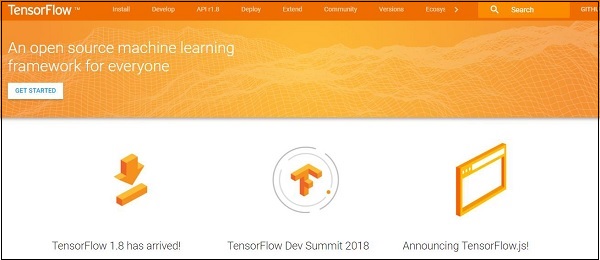


Hình 2.51.Minh họa hoạt động thuật toán GrabCut

2.4 Giới thiệu về TensorFlow trong học sâu

2.4.1 Giới thiệu

TensorFlow là một khuôn khổ học máy mã nguồn mở dành cho tất cả các nhà phát triển. Nó được sử dụng để triển khai các ứng dụng học máy và học sâu. Để phát triển và nghiên cứu các ý tưởng hấp dẫn về trí tuệ nhân tạo.TensorFlow được thiết kế bằng ngôn ngữ lập trình Python, do đó nó được coi là một khuôn khổ dễ hiểu.



Hình 2.52. TensorFlow

**Các tính năng quan trọng sau của TensorFlow**

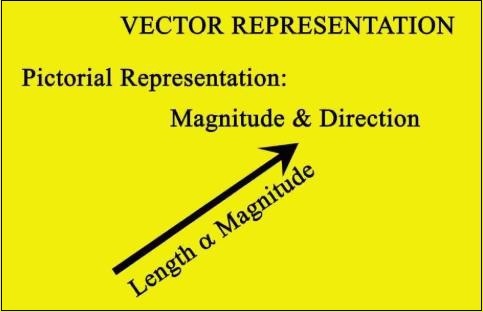
* Nó bao gồm tính năng xác định, tối ưu hóa và tính toán các biểu thức toán học một cách dễ dàng với sự trợ giúp của mảng đa chiều gọi là tenxơ.
* Nó bao gồm hỗ trợ lập trình cho mạng nơ-ron sâu và các kỹ thuật học máy.
* Nó bao gồm tính năng tính toán có khả năng mở rộng cao với nhiều tập dữ liệu khác nhau.
* TensorFlow sử dụng tính toán GPU, tự động hóa quản lý. Nó cũng bao gồm một tính năng độc đáo là tối ưu hóa cùng một bộ nhớ và dữ liệu được sử dụng.

2.4.2 TensorFlow - Nền tảng toán học

Véc tơ

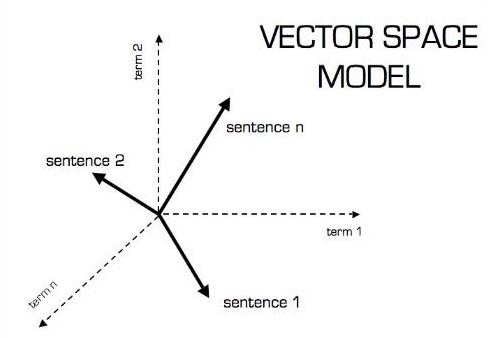
Một mảng số, có thể liên tục hoặc rời rạc, được định nghĩa là một vectơ. Các thuật toán học máy xử lý các vectơ có độ dài cố định để tạo ra kết quả tốt hơn.

Thuật toán học máy xử lý dữ liệu đa chiều nên vectơ đóng vai trò quan trọng.



Hình 2.53. Véc tơ

Biểu diễn hình ảnh của mô hình vector được hiển thị như bên dưới



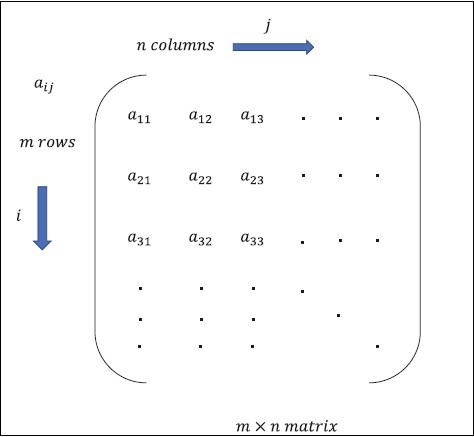
Hình 2.54. Mô hình vector

**Vô hướng**

Scalar có thể được định nghĩa là vectơ một chiều. Scalar là những vectơ chỉ bao gồm độ lớn và không có hướng. Với scalar, chúng ta chỉ quan tâm đến độ lớn.

**Ma trận**

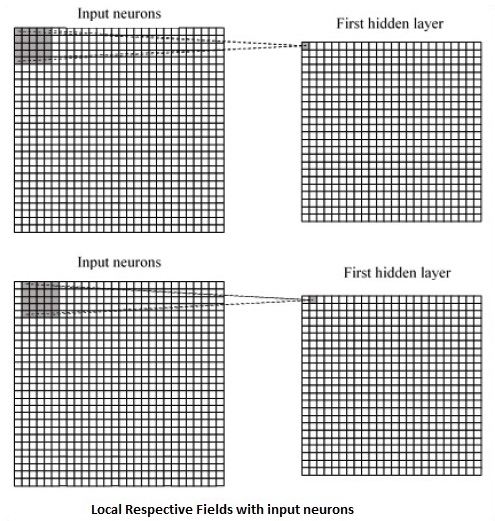
Ma trận có thể được định nghĩa là mảng đa chiều, được sắp xếp theo định dạng hàng và cột. Kích thước của ma trận được xác định theo chiều dài hàng và chiều dài cột.



Hình 2.55. Ma trận

2.4.3 Mạng nơ-ron tích chập

Mạng nơ-ron tích chập được thiết kế để xử lý dữ liệu thông qua nhiều lớp mảng. Loại mạng nơ-ron này được sử dụng trong các ứng dụng như nhận dạng hình ảnh hoặc nhận dạng khuôn mặt. Sự khác biệt chính giữa CNN và bất kỳ mạng nơ-ron thông thường nào khác là CNN lấy đầu vào là một mảng hai chiều và hoạt động trực tiếp trên hình ảnh thay vì tập trung vào việc trích xuất tính năng mà các mạng nơ-ron khác tập trung vào.



Hình 2.56. Sơ đồ biểu diễn việc tạo ra các trường cục bộ tương ứng

Ở đây, các neuron riêng lẻ thực hiện chuyển dịch theo thời gian. Quá trình này được gọi là "tích chập".

Việc ánh xạ các kết nối từ lớp đầu vào đến bản đồ đặc điểm ẩn được định nghĩa là “trọng số chung” và độ lệch bao gồm được gọi là “độ lệch chung”.

2.4.4 Convolutional Neural Networks (CNNs) trong nhận diện khuôn mặt

Convolutional Neural Networks (CNNs) là một loại mạng nơ-ron nhân tạo đặc biệt hiệu quả trong việc xử lý hình ảnh và nhận dạng đối tượng. CNNs đã trở thành tiêu chuẩn trong lĩnh vực thị giác máy tính, đặc biệt là các tác vụ như phân loại hình ảnh, nhận diện khuôn mặt, phát hiện đối tượng và phân đoạn hình ảnh.

CNNs bao gồm nhiều lớp khác nhau, bao gồm:

* Convolutional Layers: Sử dụng các bộ lọc (filters) để quét qua ảnh đầu vào và tạo ra các bản đồ đặc trưng (feature maps). Các bộ lọc này giúp phát hiện các đặc trưng quan trọng như cạnh, góc, và các mẫu hình học trong ảnh.
* Pooling Layers: Giảm kích thước của các bản đồ đặc trưng, giúp giảm số lượng tham số và tính toán trong mô hình. Max Pooling là phương pháp phổ biến nhất, chọn giá trị lớn nhất trong mỗi vùng của bản đồ đặc trưng.
* Fully Connected Layers: Kết nối mọi nút ở lớp trước với mọi nút ở lớp sau, giúp tổng hợp các đặc trưng đã được trích xuất từ các lớp trước để thực hiện dự đoán cuối cùng.
* Activation Functions: Các hàm kích hoạt như ReLU (Rectified Linear Unit) giúp mô hình học các đặc trưng phi tuyến tính từ dữ liệu.

CNNs trong Nhận Diện Khuôn Mặt

CNNs được áp dụng rộng rãi trong nhận diện khuôn mặt nhờ khả năng học và trích xuất các đặc trưng phức tạp từ ảnh. Một số mô hình CNN nổi tiếng trong nhận diện khuôn mặt bao gồm:

* LeNet-5: Một trong những mô hình CNN đầu tiên, được thiết kế cho nhận diện chữ viết tay nhưng cũng có thể áp dụng cho nhận diện khuôn mặt đơn giản.
* AlexNet: AlexNet sử dụng các lớp Convolutional và Pooling xen kẽ với các lớp Fully Connected để nhận diện các đối tượng trong ảnh.
* VGGNet: Sử dụng nhiều lớp Convolutional với các bộ lọc nhỏ (3x3) và đạt hiệu suất cao trong các tác vụ nhận diện khuôn mặt.
* FaceNet: Sử dụng CNNs để học các biểu diễn khuôn mặt (face embeddings) sao cho khoảng cách Euclidean giữa các khuôn mặt tương tự là nhỏ và giữa các khuôn mặt khác biệt là lớn. FaceNet đạt độ chính xác cao trong nhận diện khuôn mặt.
* ResNet: Giới thiệu khái niệm các kết nối dư (skip connections) giúp giảm hiện tượng mất mát thông tin và cải thiện độ sâu của mô hình..

2.5 Giới thiệu về YOLO

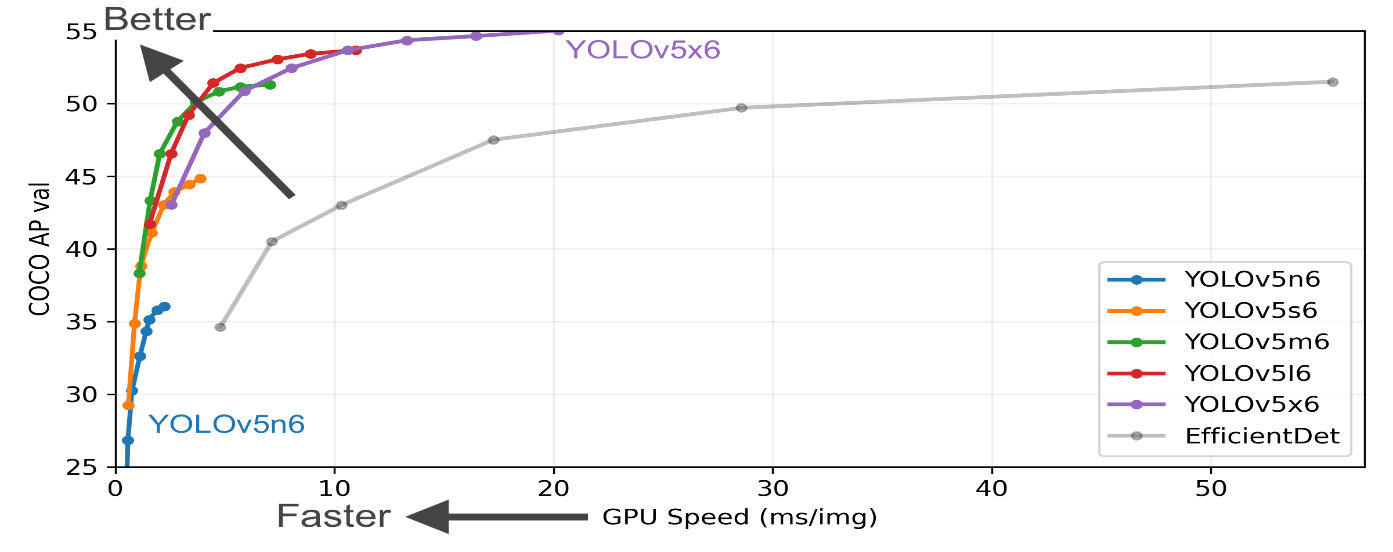
2.5.1 Tổng quan

Tổng quan về YOLO: YOLO (You Only Look Once) là một mô hình học sâu nổi bật trong việc phát hiện đối tượng trong hình ảnh và video. Khác với các phương pháp phát hiện đối tượng truyền thống, YOLO chỉ cần một lần duy nhất để nhìn vào toàn bộ hình ảnh và xác định tất cả các đối tượng trong đó.

Cấu trúc và cơ chế hoạt động của YOLO: chia hình ảnh đầu vào thành lưới, mỗi ô trong lưới sẽ dự đoán các bounding box và xác suất của các đối tượng thuộc lớp nào đó. Việc dự đoán được thực hiện trong một bước duy nhất, giúp giảm thiểu độ trễ trong thời gian thực.

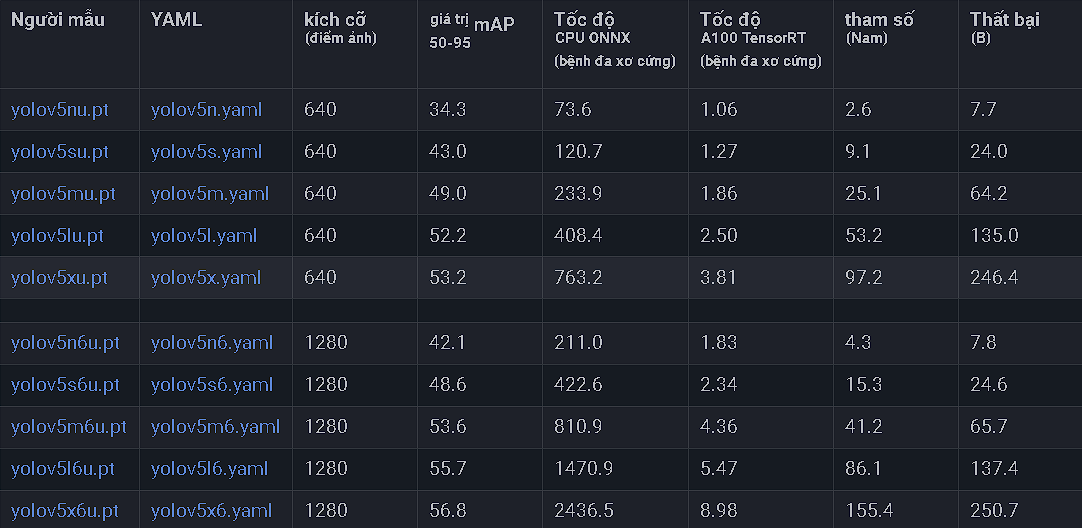
2.5.2 YOLOv5

YOLOv5u đại diện cho sự tiến bộ trong phương pháp phát hiện đối tượng . Xuất phát từ kiến trúc nền tảng của mô hình YOLOv5 được phát triển bởi Ultralytics , YOLOv5u tích hợp đầu tách không có neo, không có đối tượng, một tính năng đã được giới thiệu trước đó trong các mô hình YOLOv8 .



Hình 2.57. Hiệu suất yolov5u

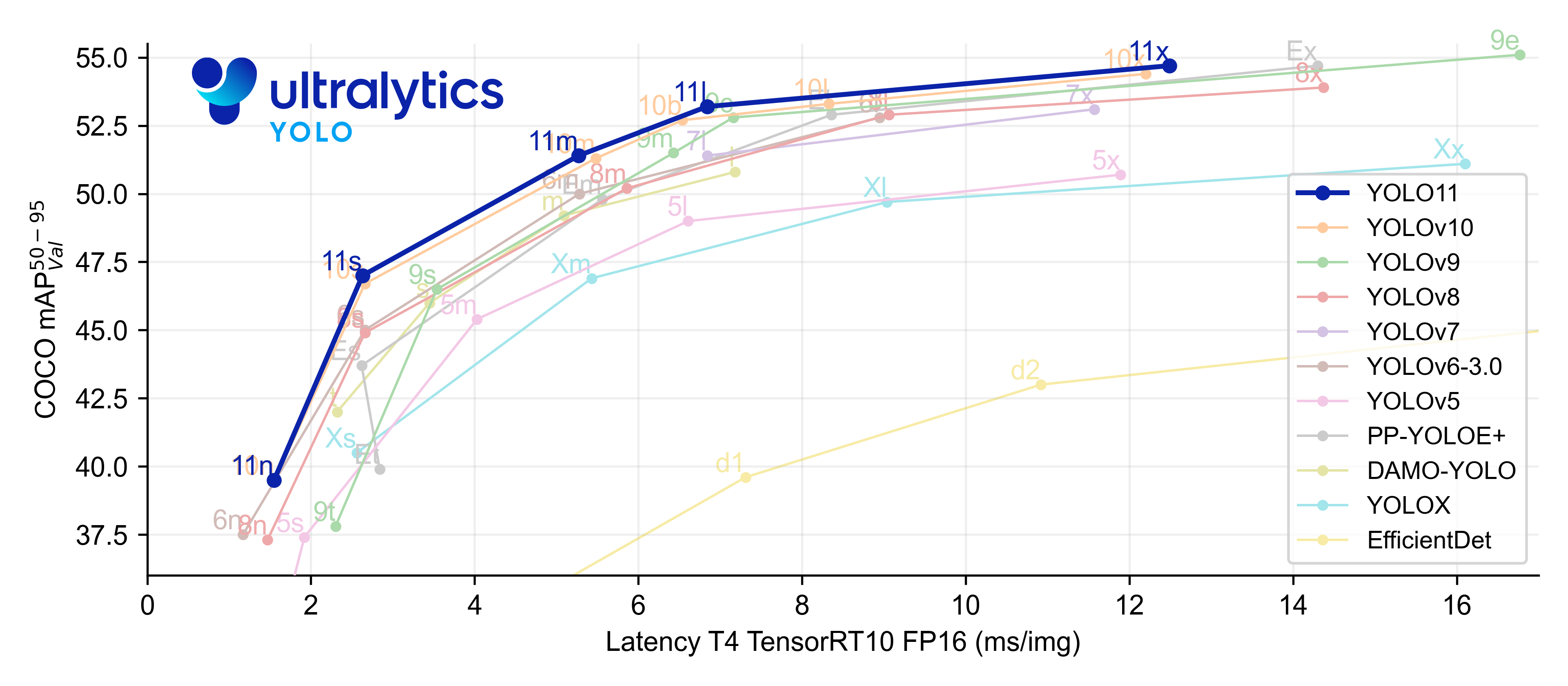
Các tính năng như tách không neo, sự cân bằng giữa tốc độ và độ chính xác, cùng với nhiều mô hình đào tạo trước làm cho YOLOv5u trở thành một lựa chọn lý tưởng



Hình 2.58. Số liệu hiệu suất yolov5u

2.5.3 YOLO11

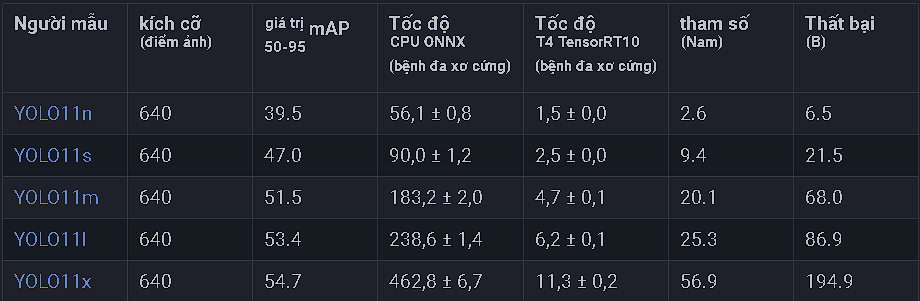
YOLO11 là phiên bản mới nhất của Ultralytics YOLO loạt các máy dò vật thể thời gian thực, định nghĩa lại những gì có thể với độ chính xác , tốc độ và hiệu quả tiên tiến. Xây dựng dựa trên những tiến bộ ấn tượng của trước đây YOLO phiên bản, YOLO11 mang đến những cải tiến đáng kể về kiến trúc và phương pháp đào tạo, khiến nó trở thành lựa chọn linh hoạt cho nhiều tác vụ thị giác máy tính .



Hình 2.59. Hiệu suất yolo 11

Các tính năng chính

YOLO11 là một cải tiến lớn so với các phiên bản trước, với những tính năng nâng cao giúp cải thiện độ chính xác và tốc độ phát hiện đối tượng. Các tính năng như trích xuất tính năng nâng cao, tối ưu hóa hiệu suất và tốc độ, và đạt được độ chính xác cao với ít tham số hơn



Hình 2.60. Số liệu hiệu suất yolo11

2.5.1 So sánh YOLO với các mô hình phát hiện đối tượng khác

**Faster R-CNN**

Cấu trúc: Faster R-CNN kết hợp hai giai đoạn chính: Region Proposal Network (RPN) để tạo ra các đề xuất vùng và một mạng CNN để phân loại và tinh chỉnh các đề xuất này.

Độ chính xác: Độ chính xác cao trong việc phát hiện đối tượng nhờ vào cấu trúc hai giai đoạn chi tiết.

Tốc độ: Tốc độ chậm hơn so với YOLO vì phải thực hiện hai giai đoạn riêng biệt.

**SSD (Single Shot MultiBox Detector)**

Cấu trúc: SSD thực hiện cả phát hiện và phân loại đối tượng trong một bước duy nhất bằng cách sử dụng nhiều lớp convolutional để dự đoán các bounding boxes với các tỷ lệ và kích thước khác nhau.

Độ chính xác: Độ chính xác cao

Tốc độ: Tốc độ nhanh, gần tương đương với YOLO, vì không có giai đoạn đề xuất vùng.

**RetinaNet**

Cấu trúc: RetinaNet sử dụng cấu trúc Feature Pyramid Network (FPN) và một mạng backbone CNN để phát hiện đối tượng. Nó nổi bật với cơ chế focal loss để xử lý vấn đề mất cân bằng lớp.

Độ chính xác: Độ chính xác rất cao

Tốc độ: Tốc độ trung bình, chậm hơn YOLO

**EfficientDet**

Cấu trúc: EfficientDet sử dụng EfficientNet làm backbone kết hợp với BiFPN (Bidirectional Feature Pyramid Network) để cải thiện việc chia sẻ và sử dụng các đặc trưng.

Độ chính xác: Độ chính xác cao nhờ vào kiến trúc tối ưu hóa và sử dụng hiệu quả tài nguyên.

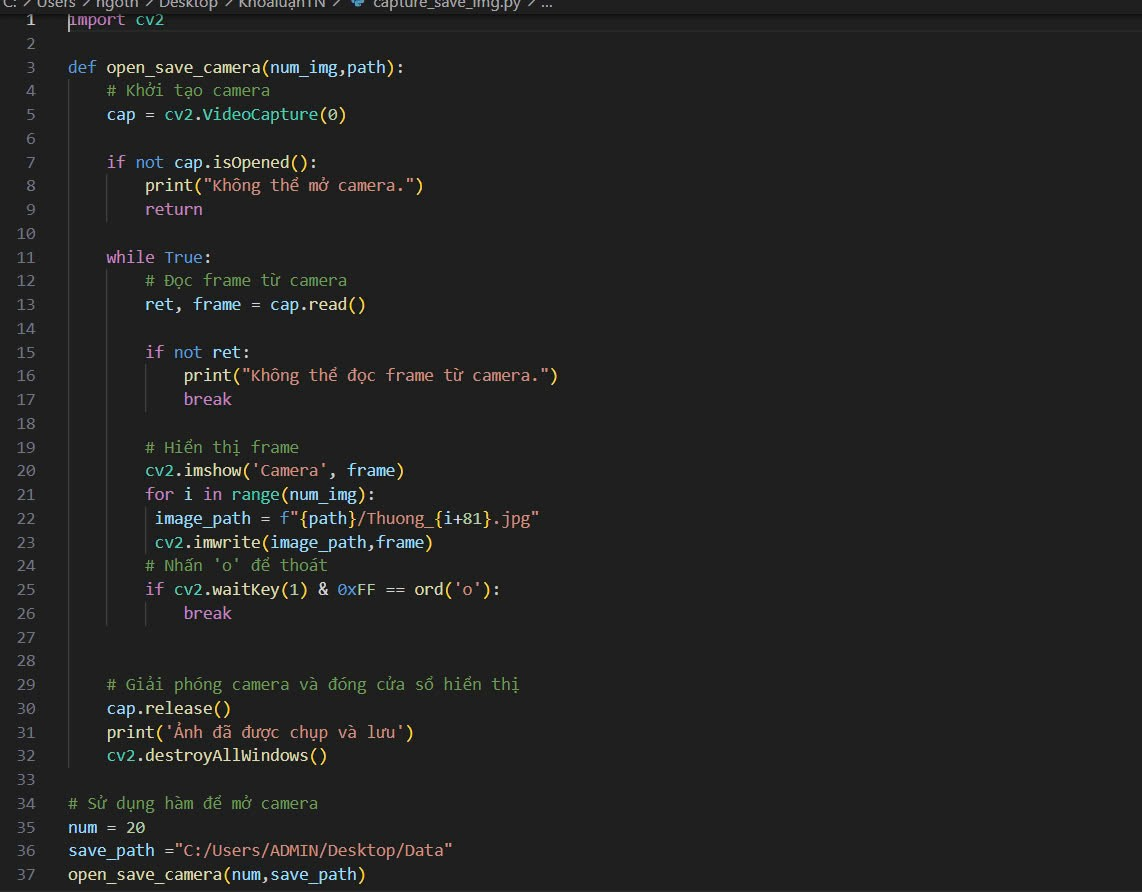
Tốc độ: Tốc độ, gần tương đương với YOLO.

**Kết luận:**YOLO nổi bật với tốc độ nhanh, phù hợp cho các ứng dụng thời gian thực, trong khi các mô hình khác như Faster R-CNN và RetinaNet có độ chính xác cao hơn nhưng tốc độ chậm hơn, phù hợp cho các ứng dụng cần độ chính xác cao hơn và không yêu cầu thời gian thực.

CHƯƠNG 3: XÂY DỰNG MÔ HÌNH

3.1 Tạo dữ liệu

3.1.1 Tạo dữ liệu yolo



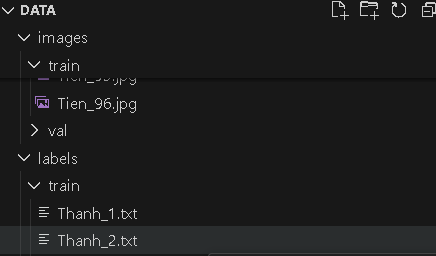
Hình 3.1 Hàm thu thập dữ liệu yolo

Tập data để train yolo có 300 ảnh gồm 3 nhãn đó là Tiên,Thành,Thương có size 640x640

Số lượng ảnh của mỗi nhãn là 100 ảnh

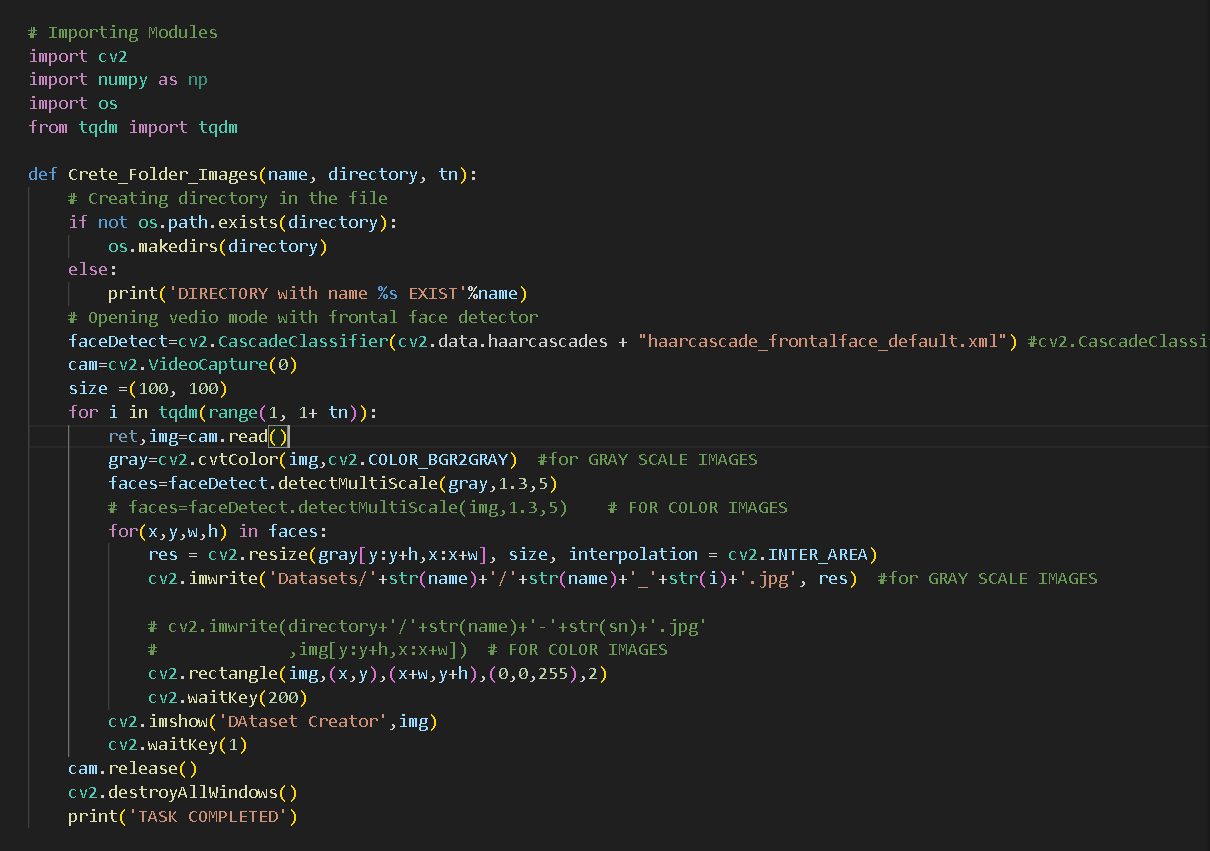
Hình 3.2 Kết quả thu thập



Hình 3.3 Cấu trúc tập data yolo

3.1.2 Tạo dữ liệu CNN

Ở hàm bên dưới em dùng thuật toán CascadeClasifier của open cv để xác định khuôn mặt để thu thập dữ liệu cho mô hình CNN của mình



Hình 3.4 Hàm thu thập dữ liệu CNN

Tập data để train mô hình CNN có 312 ảnh gồm 4 nhãn đó là Khánh,Tiên,Thành,Thương có size 100x100

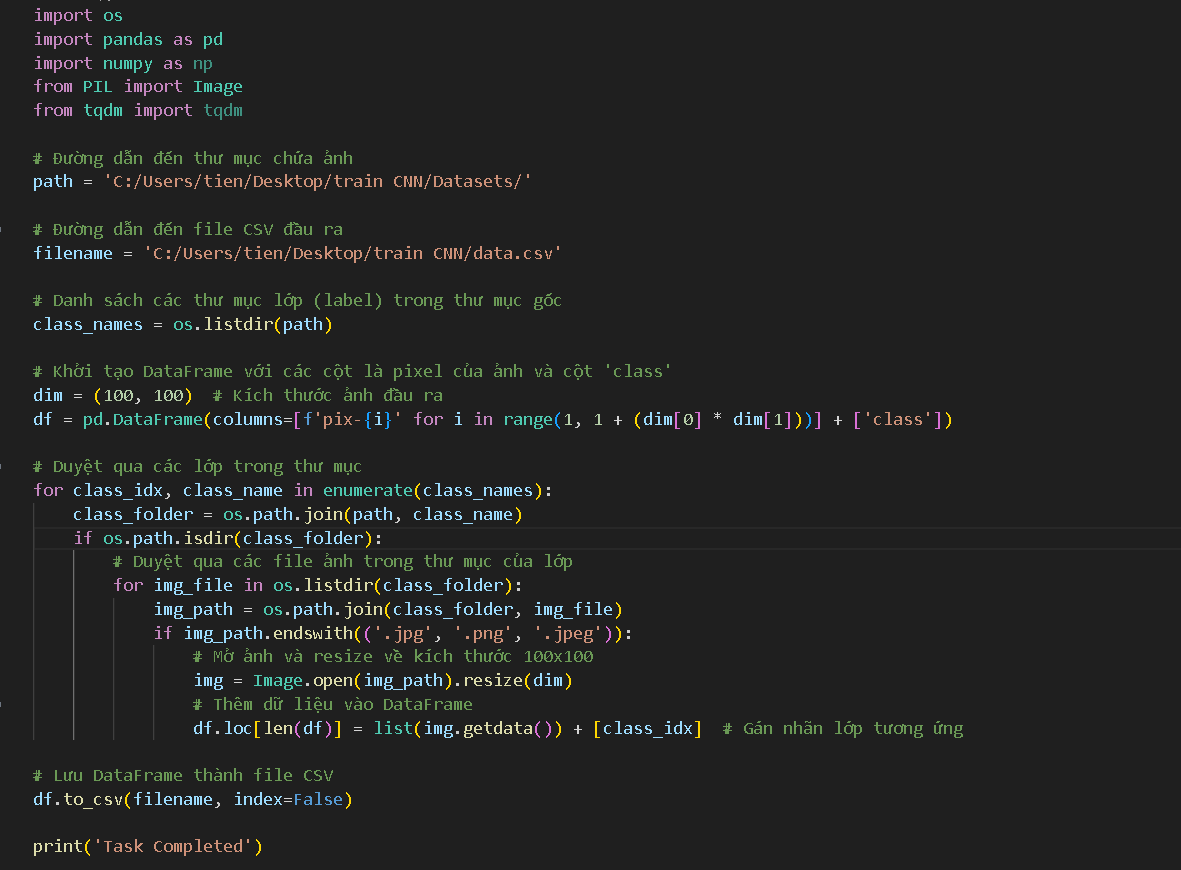
Số lượng ảnh của mỗi nhãn lần lượt là Khánh 132 ảnh,Tiên74 ảnh ,Thành 40 ảnh và Thương 66 ảnh



Hình 3.5 Kết quả thu thập

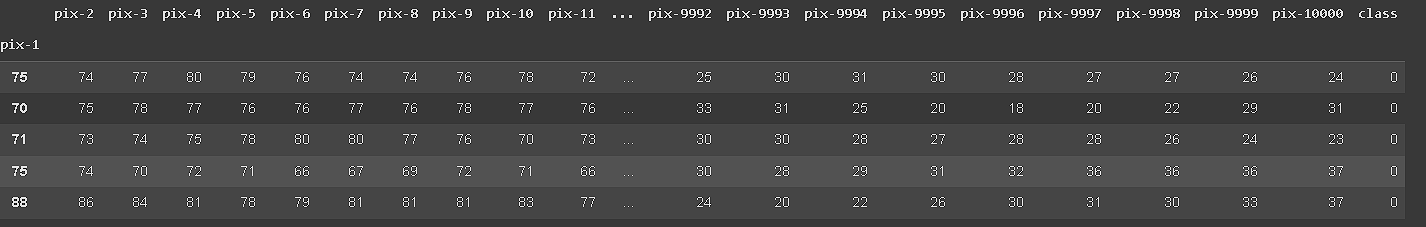


Hình 3.6 Cấu trúc file data



Hình 3.7 Hàm chuyển đổi thành file csv

Sau khi chuyển thành file csv



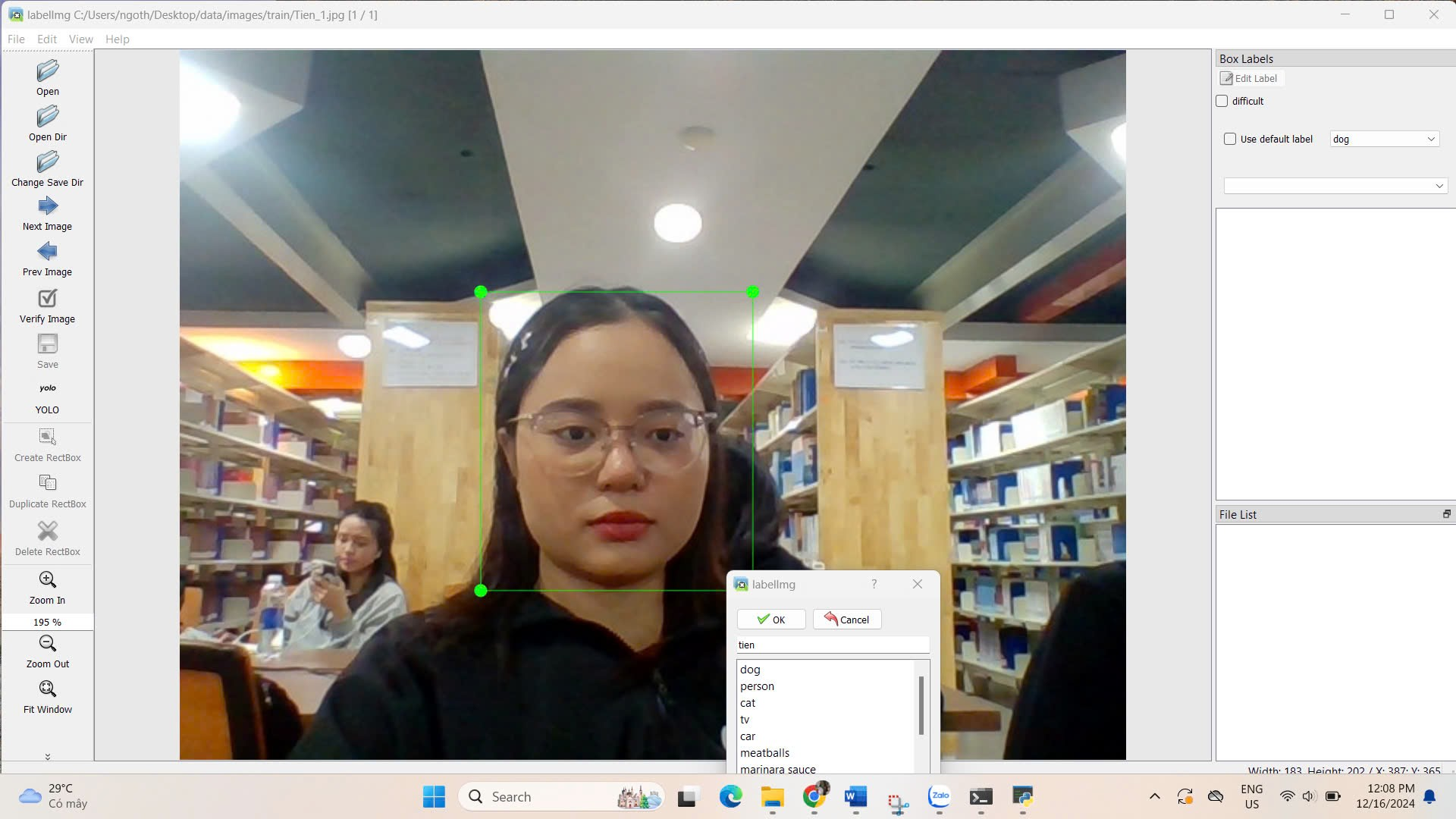
Hình 3.8 Kết quả khi chuyển thành file csv

3.2 Xử lý hình ảnh

3.2.1 YOLO

Đánh nhãn

Em sử dụng labelImg để gán nhãn cho dữ liệu



Hình 3.9 Đánh nhãn dữ liệu

3.2.2 CNNs

Ở hàm thu thập ảnh ở trên em cũng có xử lý ảnh ở trong hàm như resize ảnh về 100x100 chuyển màu ảnh về màu xám để phù hợp với mô hình

Và ở trong colab trước khi huấn luyện em thực hiện các bước sau

Cắt dữ liệu



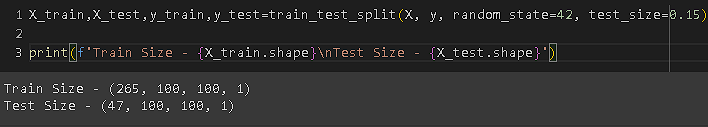
Hình 3.10 Cắt dữ liệu

Encoding label



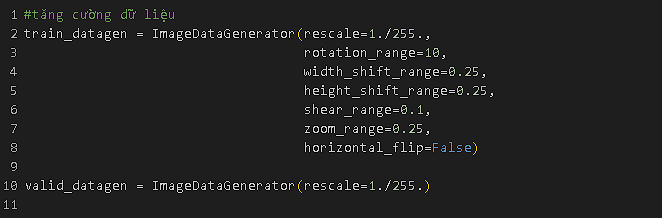
Hình 3.11 Encoding label

Chia tập train và tập test



Hình 3.12 Chia tập train và test

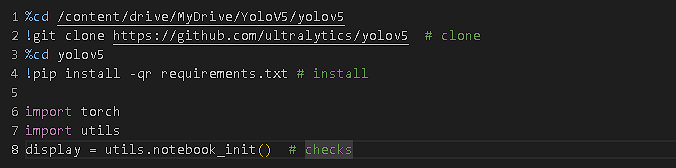
Cường hóa dữ liệu



Hình 3.13 Cường hóa dữ liệu

4.3 Huấn luyện mô hình

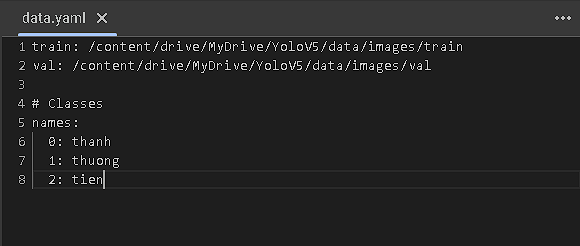
4.3.1 Yolov5



Hình 3.14 Tải yolov5



Hình 3.15 Train yolov5

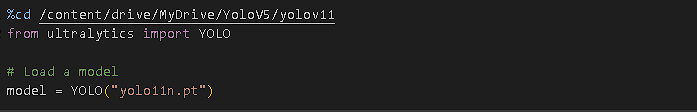


Hình 3.16 File data.yaml

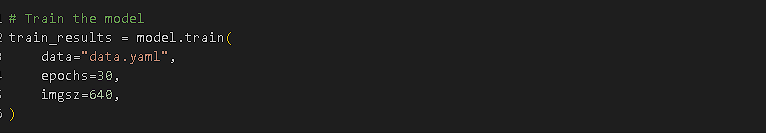
3.3.2 Yolo11



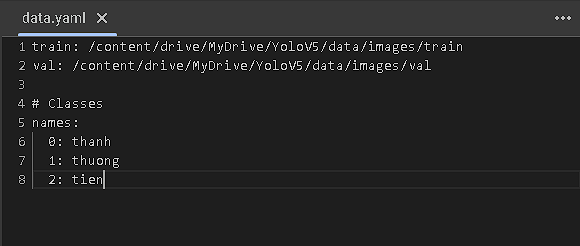
Hình 3.17 Tải ultralytics



Hình 3.18 Gọi yolo11

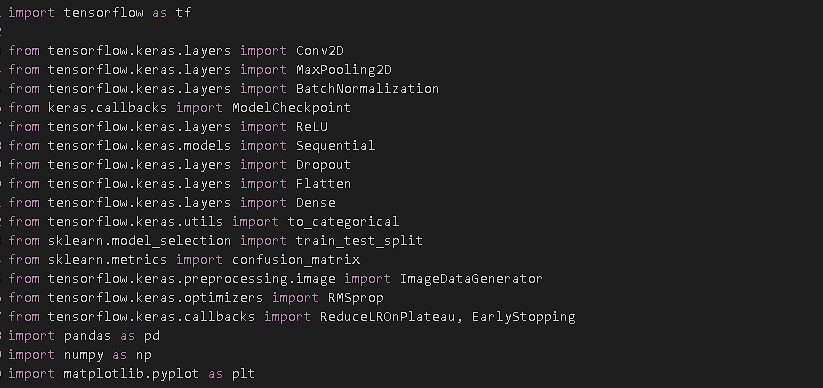


Hình 3.19 Train

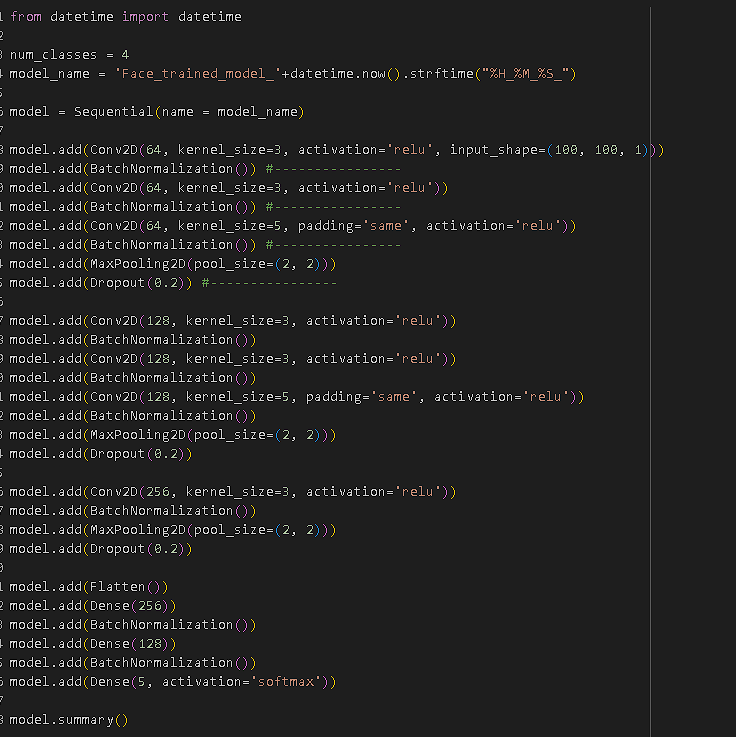


Hình 3.20 File data.yaml

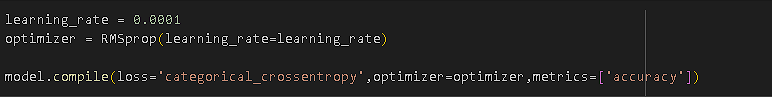
3.3.3.CNN



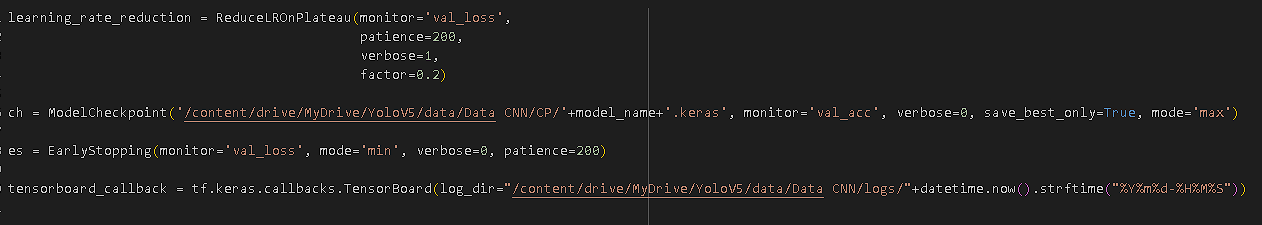
Hình 3.21 Thư viện



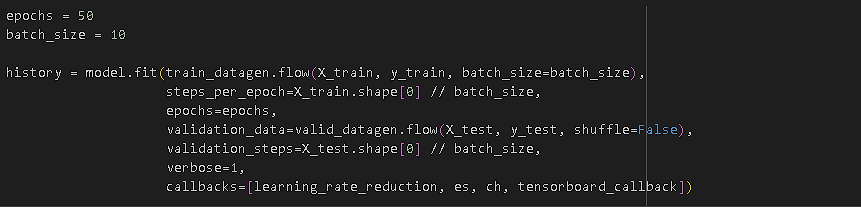
Hình 3.22 Mô hình



Hình 3.23 Loss & learning\_rate và acc



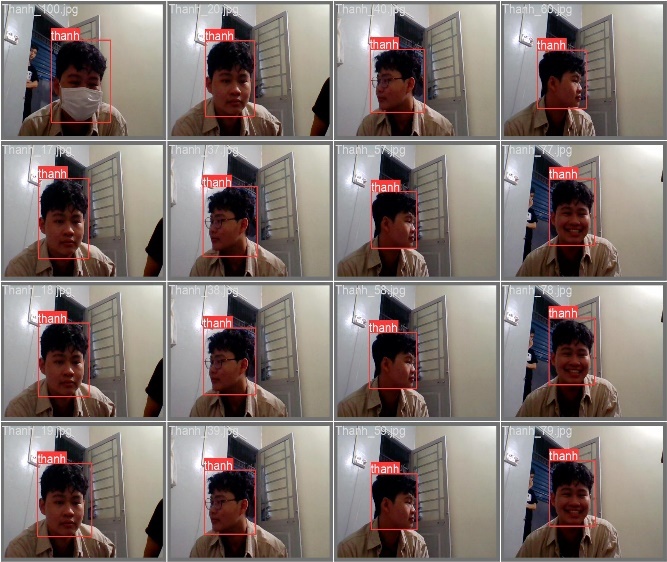
Hình 3.24 Checkpoint & EarlyStopping và tensorboard



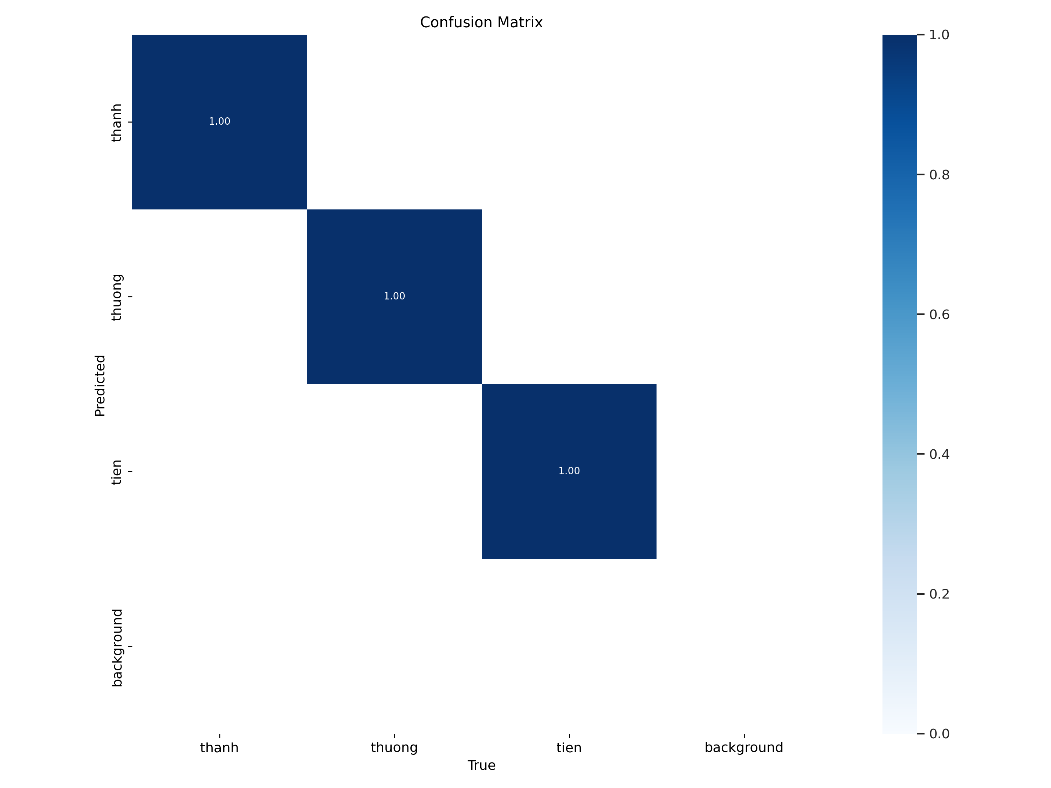
Hình 3.25 Train

3.4 Kết quả

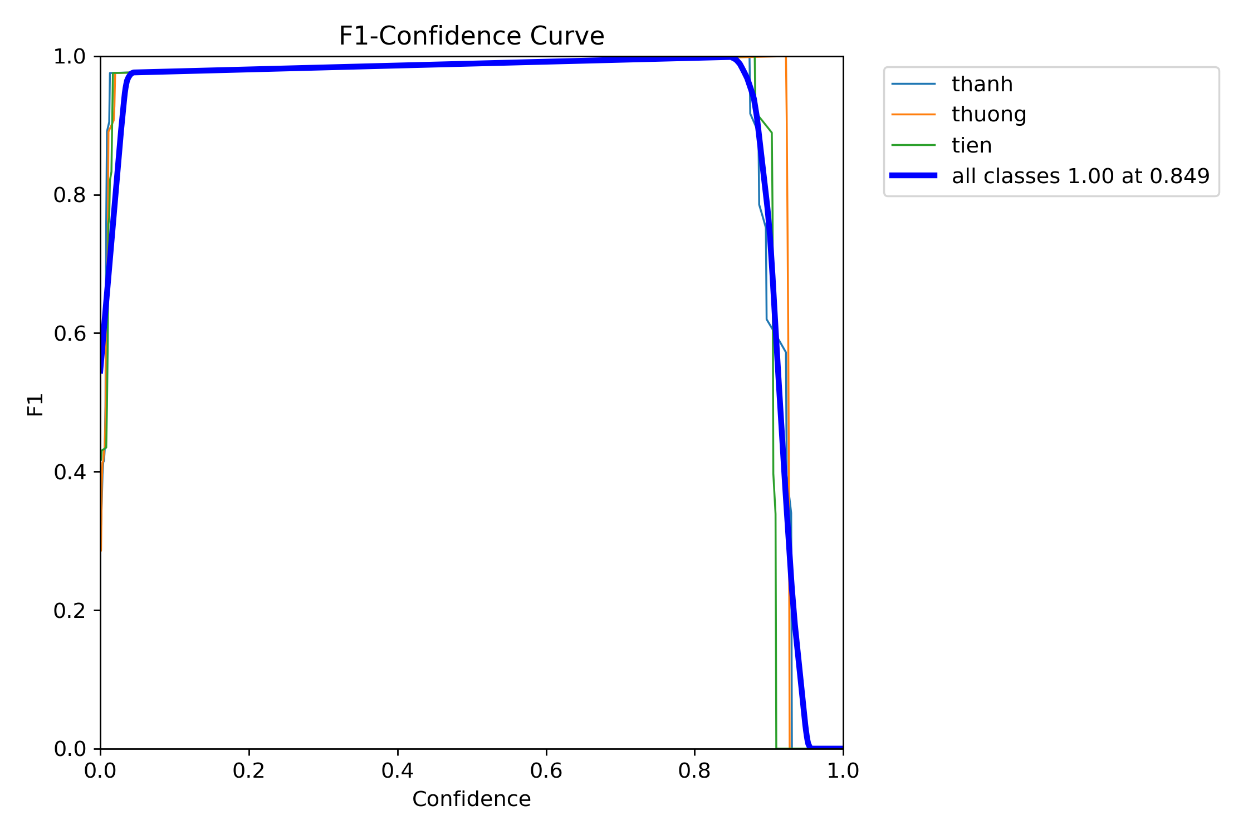
3.4.1. YOLOV5



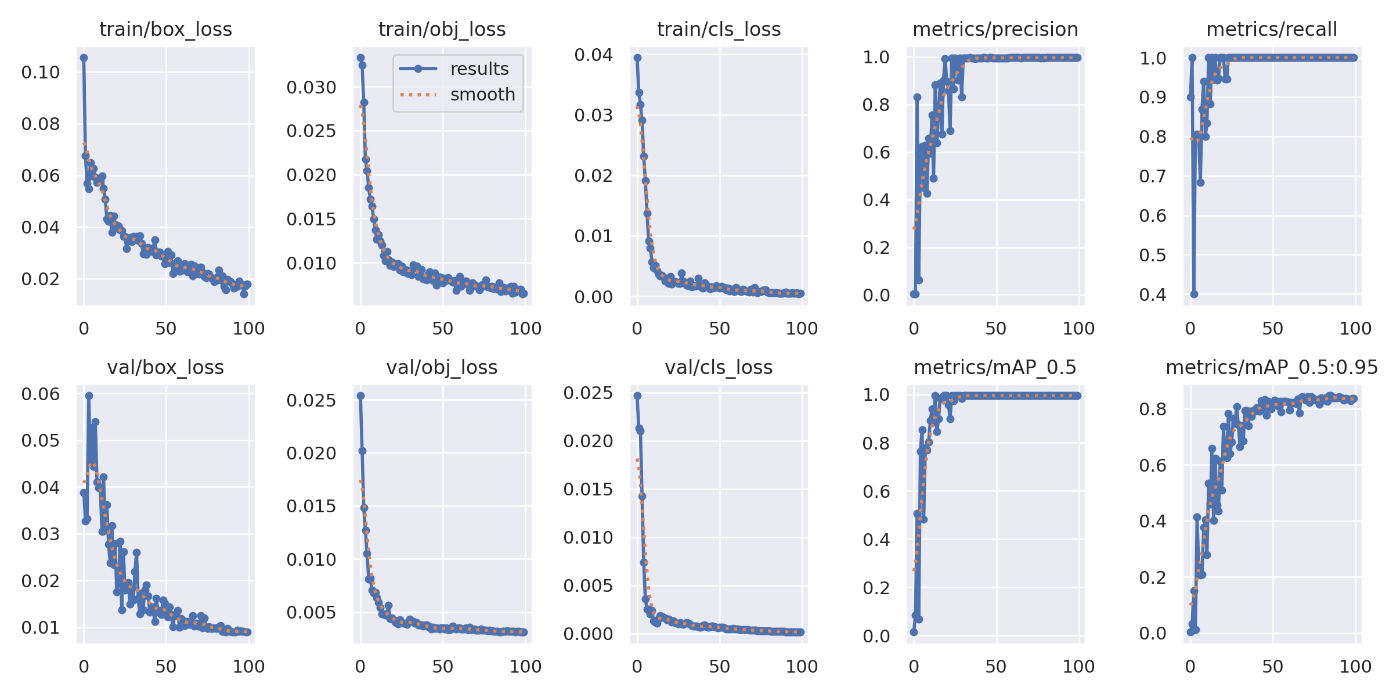
Hình 3.26 Kết quả



Hình 3.27 Confusion Matrix



Hình 3.28 F1-Confidence Curve

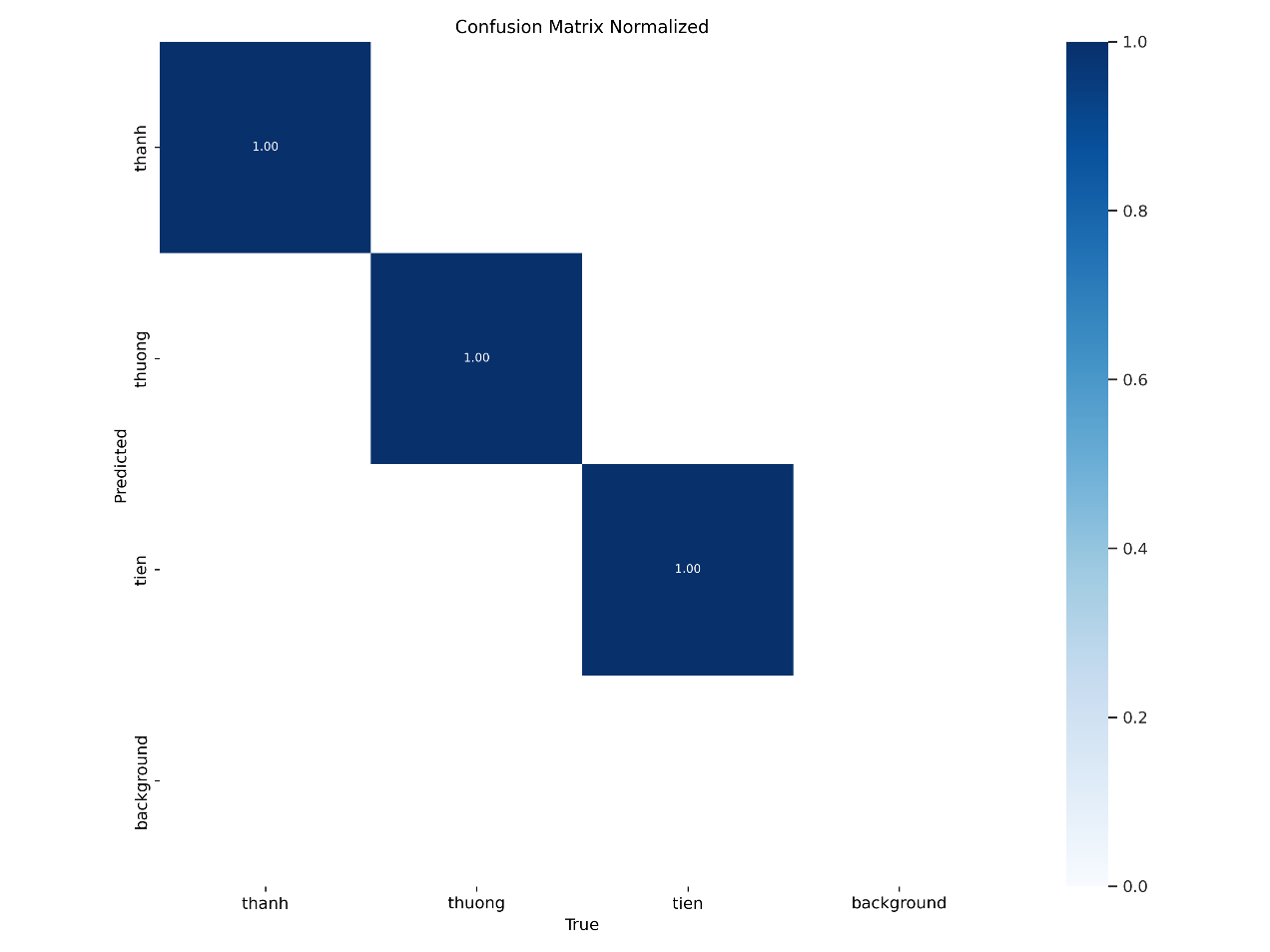


Hình 3.29 results

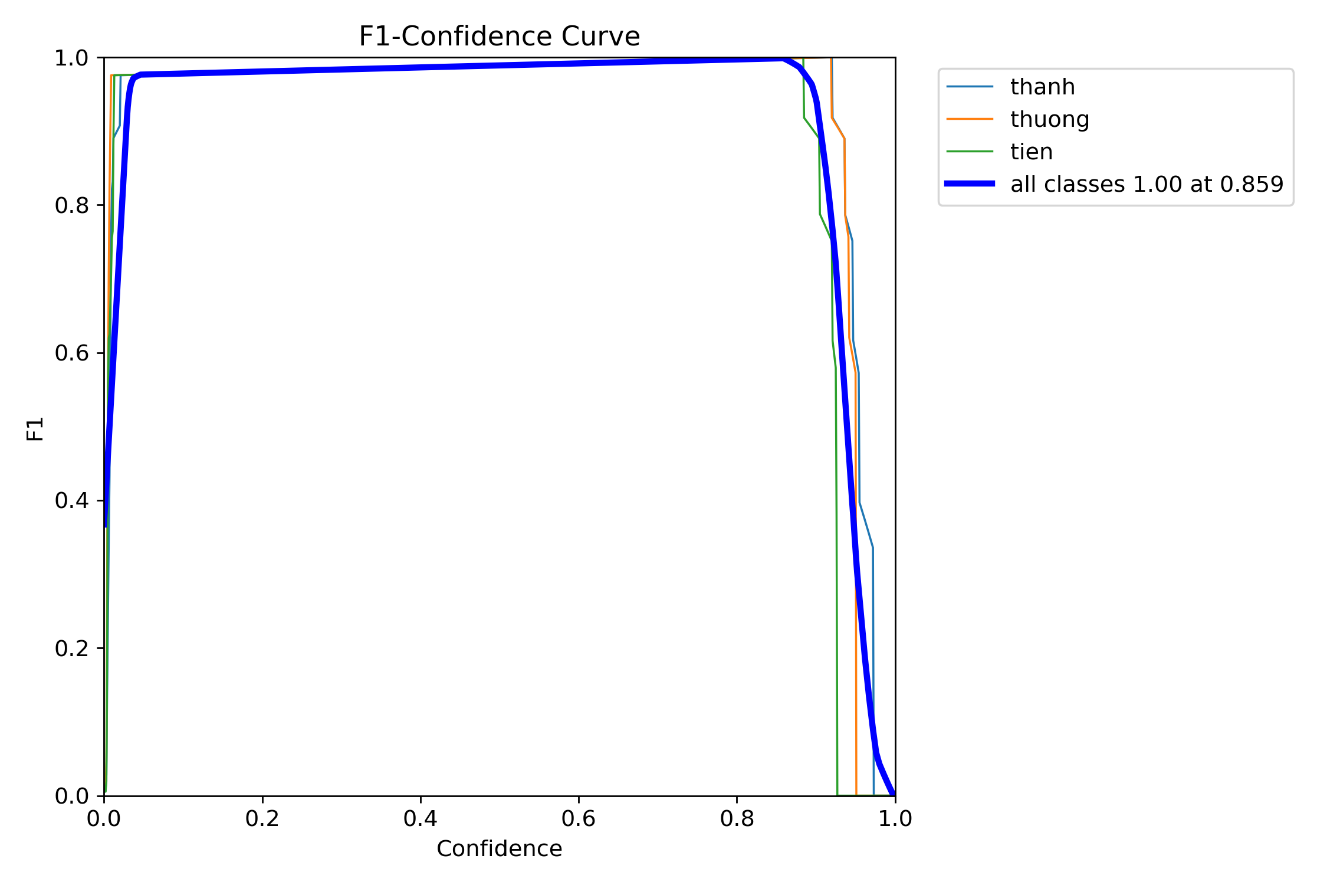
4.4.2 YOLO11

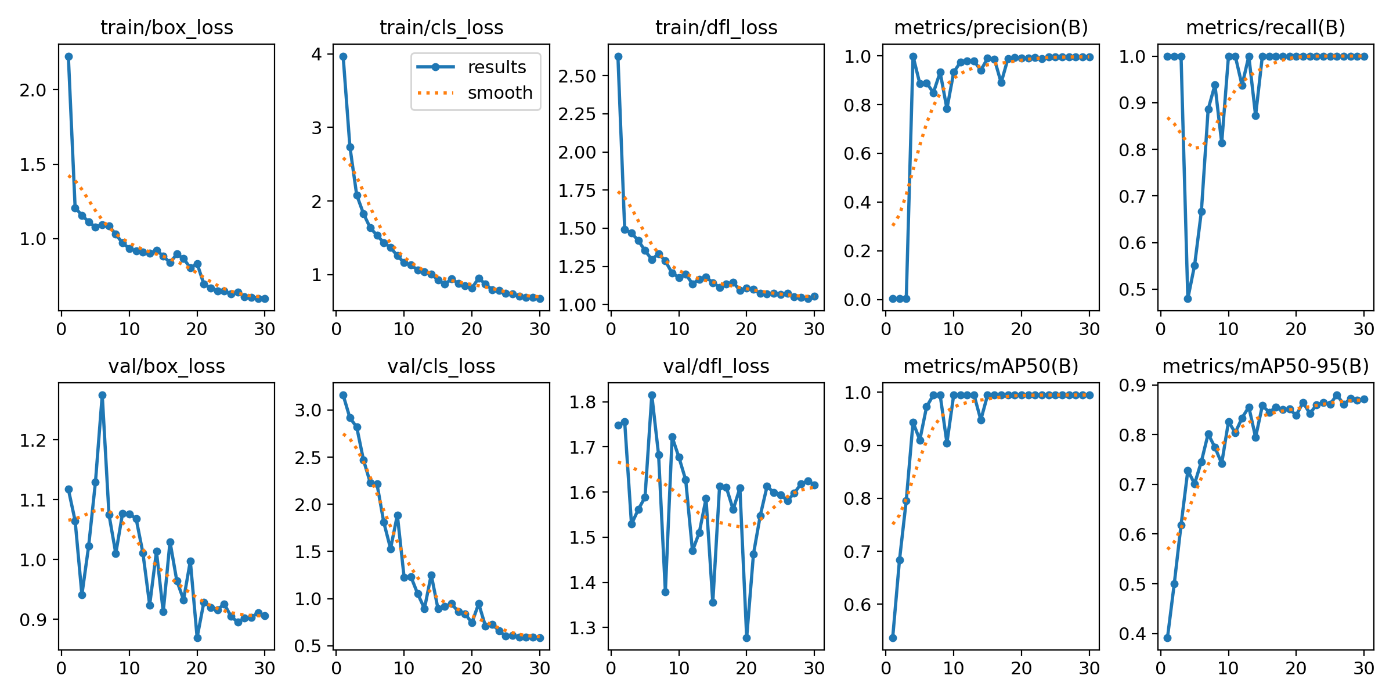
Hình 3.30 Kết quả



Hình 3.31 Confusion Matrix

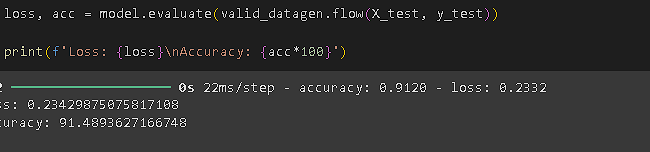


Hình 3.32 F1-Confidence Curve

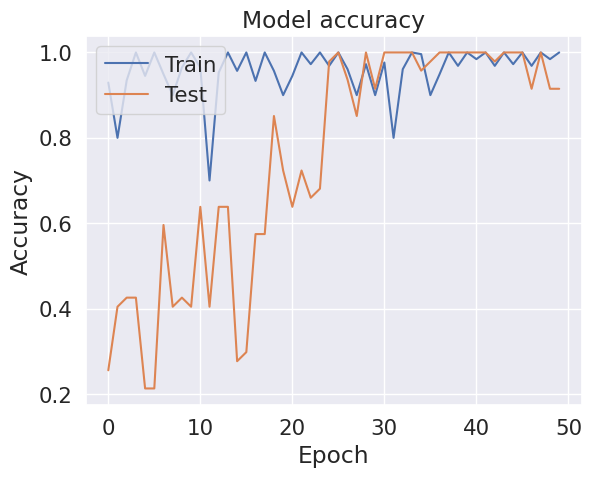


Hình 3.33 results

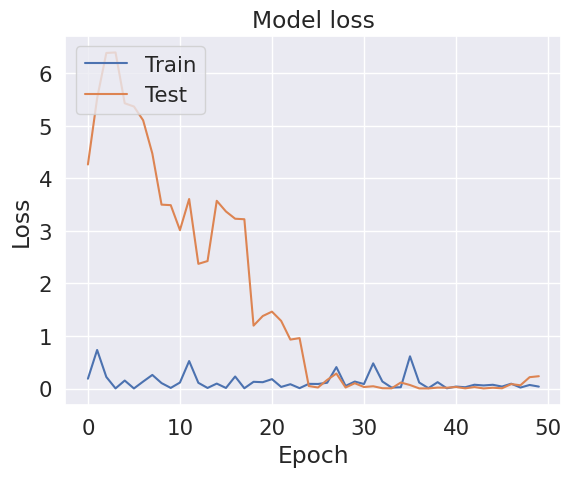
3.4.3 CNN



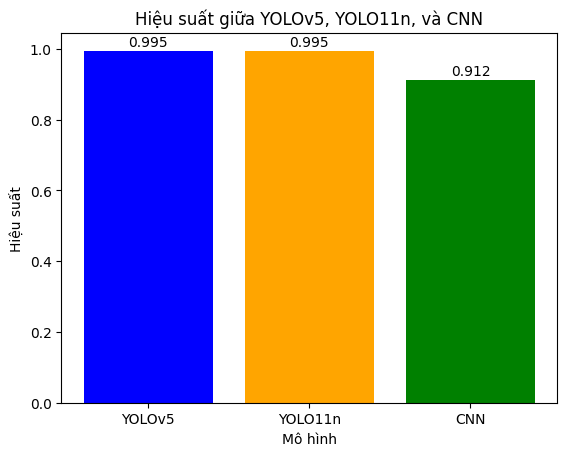
Hình 3.34 accuracy



Hình 3.35 Model accuracy



Hình 3.36 Model loss

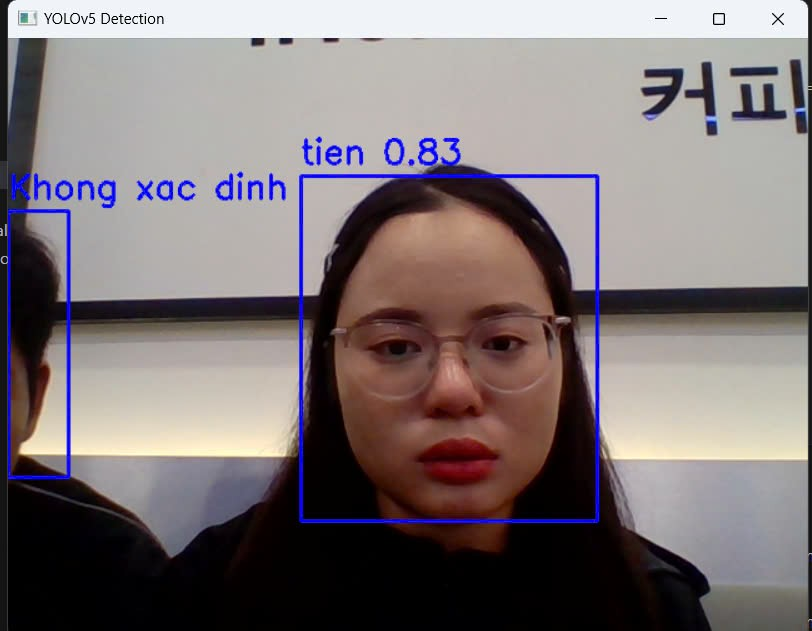


Hình 3.37 Hiệu suất của 3 mô hình

CHƯƠNG 4: KẾT QUẢ THỰC NGHIỆM

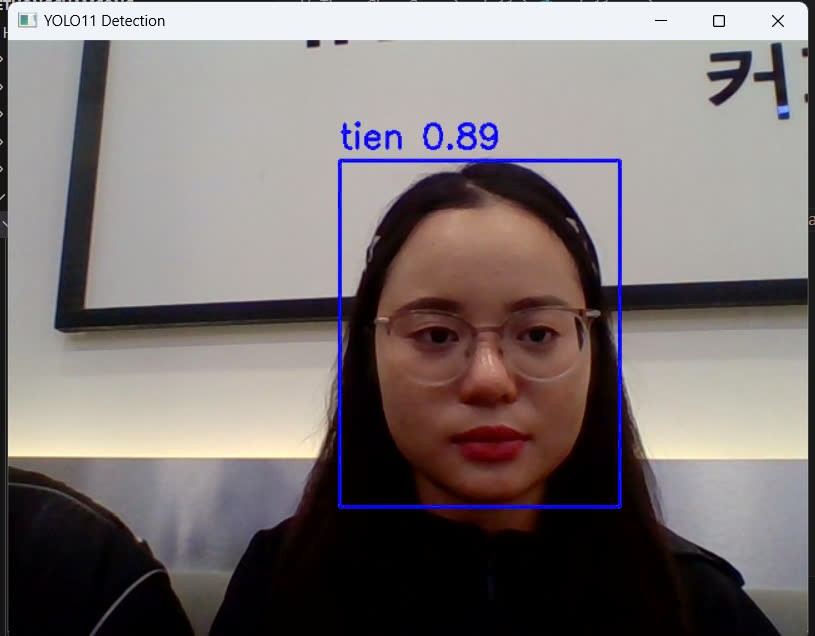
4.1 Yolo

4.1.1 Yolo v5



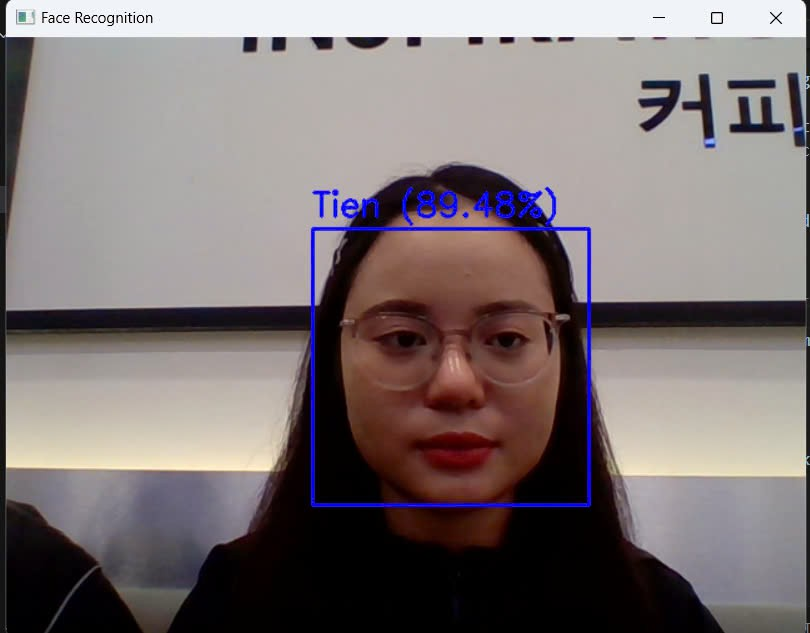
Hình 4.1 Nhận diện thời gian thực yolov5

4.1.2 Yolo 11



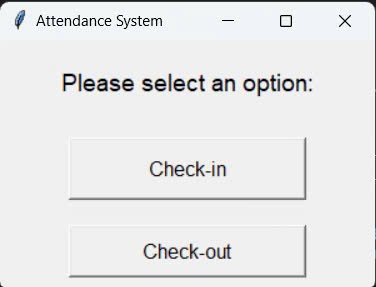
Hình 4.2 Nhận diện thời gian thực yolo11

4.2 CNNs

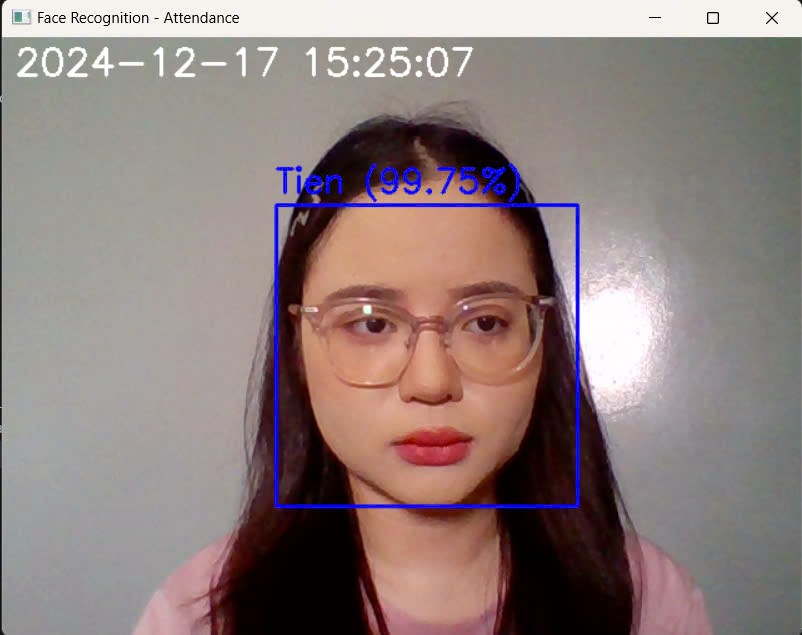


Hình 4.3 Nhận diện thời gian thực CNN

4.3 Hệ thống chấm công



Hình 4.4 Giao diện

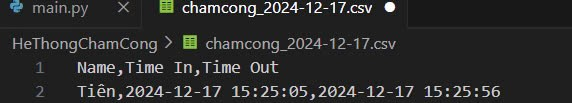


Hình 4.5 Nhận diện check-int



Hình 4.6 Nhận diện check-out

Khi chấm công thành công thì hệ thống sẽ phát ra tiếng kêu để xác nhận đã xác nhận đã chấm công thành công và lưu trữ tên và thời gian chấm công vào một file csv



Hình 4.7 File csv lưu trữ thời gian chấm công

CHƯƠNG 5: TỔNG KẾT

Trong bối cảnh công nghệ phát triển mạnh mẽ, hệ thống chấm công tự động đã trở thành một yêu cầu thiết yếu đối với nhiều doanh nghiệp, tổ chức. Hệ thống này không chỉ giúp nâng cao tính chính xác mà còn giảm thiểu các sai sót liên quan đến quá trình chấm công thủ công. Đề tài này nghiên cứu và phát triển một hệ thống chấm công sử dụng công nghệ nhận dạng khuôn mặt, giúp nhận diện và ghi nhận thời gian vào ra của nhân viên một cách tự động.

Hệ thống được xây dựng dựa trên mô hình Convolutional Neural Networks (CNN), một mô hình học sâu mạnh mẽ trong nhận diện ảnh. Ngoài ra, đề tài còn sử dụng và so sánh hai mô hình khác là YOLOv5 và YOLO11n với mục đích đánh giá hiệu suất của các mô hình này trong bài toán nhận diện khuôn mặt. Mô hình CNN được lựa chọn làm mô hình chính cho việc chấm công do tính hiệu quả trong việc nhận diện khuôn mặt trong môi trường thực tế.

Mục tiêu của đề tài không chỉ là phát triển hệ thống nhận diện khuôn mặt mà còn là xây dựng giao diện người dùng đơn giản sử dụng Tkinter, giúp người dùng dễ dàng tương tác và quản lý hệ thống.

Mục tiêu chính của đề tài là xây dựng một hệ thống nhận dạng khuôn mặt để chấm công tự động, sử dụng mô hình CNN để nhận diện khuôn mặt, từ đó ghi lại thời gian chấm công của nhân viên. Đề tài cũng nghiên cứu và so sánh độ chính xác giữa các mô hình nhận dạng khuôn mặt khác nhau như YOLOv5 và YOLO11n, nhằm chọn lựa mô hình phù hợp với yêu cầu ứng dụng thực tế. Hệ thống này sẽ hỗ trợ các doanh nghiệp trong việc theo dõi thời gian làm việc của nhân viên một cách tự động và hiệu quả.

Công nghệ sử dụng:

* OpenCV: Được sử dụng để xử lý ảnh và nhận diện khuôn mặt từ video trực tiếp. OpenCV cung cấp các thuật toán phát hiện khuôn mặt mạnh mẽ và có thể xử lý nhanh chóng các ảnh đầu vào.
* TensorFlow: Đây là framework chính để xây dựng và huấn luyện mô hình CNN cho việc nhận diện khuôn mặt. TensorFlow hỗ trợ các mô hình học sâu, giúp việc huấn luyện và triển khai mô hình trở nên dễ dàng và hiệu quả.
* YOLOv5 và YOLO11n: Cả hai mô hình này đều là các mô hình học sâu nổi bật trong việc nhận diện đối tượng. Chúng được sử dụng trong đề tài để so sánh với CNN về hiệu suất nhận diện khuôn mặt.
* Tkinter: Dùng để xây dựng giao diện người dùng đơn giản. Giao diện này giúp người dùng dễ dàng tương tác với hệ thống chấm công và theo dõi các thông tin liên quan đến việc chấm công.

Quy trình hoạt động của hệ thống:

* Nhận diện khuôn mặt: Hệ thống sẽ sử dụng mô hình CNN để nhận diện khuôn mặt của nhân viên khi họ đứng trước camera. Quá trình nhận diện sẽ được thực hiện ngay lập tức khi có người xuất hiện trong khung hình.
* Chấm công và lưu trữ thời gian: Sau khi nhận diện khuôn mặt thành công, hệ thống sẽ tự động ghi lại thời gian chấm công vào file CSV. Các thông tin về tên nhân viên và thời gian chấm công sẽ được lưu lại, giúp quản lý dễ dàng theo dõi.
* Giao diện người dùng: Giao diện được xây dựng đơn giản nhưng đầy đủ chức năng, người dùng có thể dễ dàng thấy được thông báo khi chấm công thành công và theo dõi quá trình nhận diện khuôn mặt. Giao diện này được xây dựng bằng Tkinter, với các cửa sổ thông báo và các nút bấm đơn giản, phù hợp cho việc sử dụng trong môi trường công sở.

Trong quá trình huấn luyện và thử nghiệm ba mô hình (CNN, YOLOv5 và YOLO11n), các kết quả độ chính xác (accuracy) của các mô hình được ghi nhận như sau:

* CNN: Đạt độ chính xác 91.2% trong việc nhận diện khuôn mặt.
* YOLOv5: Đạt độ chính xác rất cao là 99.5%.
* YOLO11n: Cũng đạt độ chính xác cao tương đương với YOLOv5, 99.5%.

Các kết quả cho thấy YOLOv5 và YOLO11n đều có độ chính xác vượt trội hơn so với CNN. Tuy nhiên, với các yêu cầu về tài nguyên máy tính và môi trường thực tế, mô hình CNN vẫn đủ mạnh và có thể áp dụng hiệu quả trong nhiều trường hợp chấm công, đặc biệt khi yêu cầu về tốc độ và tài nguyên không quá khắt khe.

Ưu điểm của hệ thống:

* Độ chính xác cao: Các mô hình YOLOv5 và YOLO11n cho kết quả nhận diện rất chính xác, gần như đạt 100% trong các thử nghiệm với các bộ dữ liệu thử nghiệm nhỏ. Hệ thống nhận diện khuôn mặt rất nhanh chóng và chính xác.
* Giao diện người dùng thân thiện: Giao diện đơn giản giúp người dùng dễ dàng sử dụng mà không cần nhiều kiến thức về công nghệ.
* Tiết kiệm thời gian và nhân lực: Hệ thống tự động hóa quá trình chấm công, giúp tiết kiệm thời gian cho cả nhân viên và người quản lý, đồng thời giảm thiểu sai sót trong việc ghi nhận thời gian.
* Lưu trữ dữ liệu hiệu quả: Dữ liệu về thời gian chấm công được lưu trữ tự động trong file CSV, giúp dễ dàng theo dõi và quản lý.

Hướng phát triển trong tương lai:

* Kết nối cơ sở dữ liệu: Mở rộng hệ thống để kết nối với các cơ sở dữ liệu như MySQL hoặc MongoDB nhằm lưu trữ dữ liệu chấm công lâu dài và dễ dàng truy xuất.
* Cải thiện độ chính xác: Nâng cấp mô hình CNN hoặc thử nghiệm với các mô hình học sâu phức tạp hơn như ResNet hay VGG để đạt được kết quả chính xác hơn.
* Ứng dụng trên các nền tảng khác: Phát triển hệ thống chấm công cho các nền tảng di động hoặc tích hợp vào các ứng dụng chấm công trực tuyến để mở rộng khả năng sử dụng.
* Tăng cường tính bảo mật: Cải thiện hệ thống để nhận diện và bảo mật hơn, đảm bảo không xảy ra việc gian lận trong quá trình chấm công như việc sử dụng ảnh giả mạo.

Kết luận: Đề tài đã thành công trong việc phát triển một hệ thống chấm công tự động sử dụng công nghệ nhận dạng khuôn mặt. Hệ thống này không chỉ giúp giảm thiểu sai sót trong quá trình chấm công mà còn nâng cao hiệu quả công việc trong các doanh nghiệp. Các mô hình YOLOv5 và YOLO11n cho kết quả nhận diện rất chính xác và có thể áp dụng cho các ứng dụng cần độ chính xác cao. Tuy nhiên, mô hình CNN cũng là một lựa chọn đáng chú ý cho các bài toán nhận diện khuôn mặt trong môi trường không yêu cầu tài nguyên tính toán quá cao. Hệ thống đã đạt được mục tiêu và có thể được phát triển thêm với khả năng kết nối cơ sở dữ liệu và mở rộng quy mô.

DANH MỤC TÀI LIỆU THAM KHẢO

A. Tài liệu tham khảo tiếng Việt

1. Nguyễn Đức Anh (2019), Trí tuệ nhân tạo cơ bản và ứng dụng, NXB Khoa học và Kỹ thuật, Hà Nội.

2. Nguyễn Hồng Bảo (2020), Xử lý ảnh kỹ thuật số và ứng dụng trong thực tế, NXB Thông tin và Truyền thông, Hà Nội.

3. Trần Văn Hùng (2018), Mạng nơ-ron và ứng dụng trong nhận dạng hình ảnh, NXB Đại học Quốc gia TP. HCM, TP. HCM.

4. Vũ Đức Minh (2021), Lập trình Python cơ bản và nâng cao với OpenCV, NXB Lao động Xã hội, Hà Nội.

B. Tài liệu tham khảo tiếng Anh

1. Bishop, C.M. (2006), Pattern Recognition and Machine Learning, Springer, New York.

2. Chollet, F. (2018), Deep Learning with Python, Manning Publications, New York.

3. Goodfellow, I., Bengio, Y., Courville, A. (2016), Deep Learning, MIT Press, Cambridge, MA.

4. Kingma, D.P., Ba, J. (2014), Adam: A Method for Stochastic Optimization, arXiv preprint arXiv:1412.6980.

5. Rosebrock, A. (2017), Deep Learning for Computer Vision with Python, PyImageSearch, New York.