**KHOA CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

**ĐẠI HỌC ĐÀ NẴNG**

**BÁO CÁO**

**Trí tuệ nhân tạo**

Đề tài: Viết ứng dụng nhận dạng ngón tay, sử dụng ngón tay để điều khiển trang

Sinh viên thực hiện: **Chu Văn Hoàng**

Lớp: **17IT3**

Đà Nẵng, 12/2019

Mục lục

[**1** **Tóm tắt** 3](#_Toc27697721)

[**2** **Giới thiệu** 3](#_Toc27697722)

[**3** **Những dự án, bài viết liên quan** 3](#_Toc27697723)

[**4** **Dataset và chức năng** 4](#_Toc27697724)

[**5** **Các phương thức, hàm** 4](#_Toc27697725)

[**6** **Thử nghiệm / Kết quả / Thảo luận** 5](#_Toc27697726)

[**7** **Kết luận / Công việc dự kiến trong tương lai** 7](#_Toc27697727)

[**8** **Phụ lục** 8](#_Toc27697728)

[**9** **Đóng góp** 8](#_Toc27697729)

[**10** **Tài liệu tham khảo** 9](#_Toc27697730)

# **1 Tóm tắt**

Trong thời kỳ phát triển mạnh mẽ của ngành công nghệ thông tin, lĩnh vực Trí tuệ nhân tạo đang là một trong những lĩnh vực được nhiều nhà đầu tư một cách mạnh mẽ. Trí tuệ nhân tạo được áp dụng vào vô số các dự án với những chức năng như: nhận diện chữ viết, nhận diện biển số,… Từ đó mình cũng muốn thử sức với lĩnh vực này trong việc tạo một ứng dụng có thể nhận dạng bàn tay và đếm số ngón tay để điều khiển trang đọc. Việc tìm hiểu có sử dụng các tài liệu, dự án từ một số nguồn trên mạng, và được chỉnh sửa để có thể đạt được hiệu suất cao hơn.

# **2 Giới thiệu**

Đối với ứng dụng này, để có thể nhận diện được ngón tay, mình đã sử dụng các thư viện: Keras, Tensorflow, Opencv, và ngôn ngữ lập trình Python. Ứng dụng sử dụng thư viện Tensorflow, Keras để huấn luyện (training, lấy dữ liệu huấn luyện cho ứng dụng thông qua thông thư viện các ảnh chụp số các ngón tay từ không có ngón tay nào đến 5 ngón tay theo nhiều kiểu dáng khác nhau. Đầu vào của ứng dụng là các khung hình được lấy trực tiếp từ camera. Sau quá trình lọc nhiễu, lọc màu sang định dạng màu đen trắng (Gray). Tiếp đến là lọc ngưỡng (threshold) để có thể loại bỏ phần nền (background). Sau khi có được hình ảnh là khu vực cần nhận dạng, tìm và nhận dạng các đường viền (contours) để phân tích hình dạng từ đó nhận dạng đối tượng được đưa vào. Sau khi có được các đường viền (contours), xác định các than lồi của đối tượng thông qua (cv2.convexHull()), tiếp đến tìm các điểm khuyết tật lồi (convexity defect) dựa vào cv2.convexityDefects() từ đó có thể xác nhận được số ngón tay (đối với ứng dụng mình đang làm). Sau khi đã xác được số ngón tay, sử dụng pyautogui để kích hoạt các phím (trong ứng dụng sử dụng phím dịch lên và dịch xuống – press up, press down).

# **3 Những dự án, bài viết liên quan**

Bài viết: Using Deep learning and CNNs to make a Hand Gesture recognition model

Sử dụng Machine learning để có thể thực hiện việc phân loại, nhận diện ngón tay. Phương thức sử dụng là Deep Learning với sự trợ giúp của Convolutional Neural Networks dựa trên Tensorlow và Keras.

Điểm mạnh của thuật toán CNNs đó là có thể đưa ra kết quả với độ tin cậy cao >95% dựa trên mô hình Deep Learning, và cho hiệu suất tốt.

# **4 Dataset và chức năng**

Đối với dữ liệu thử của ứng dụng này sử dụng các hình ảnh về bàn tay để làm dữ liệu huấn luyện cho ứng dụng. Các hình ảnh được sử dụng có độ phân giải 300x300 pixel.

Dữ liệu huấn luyện gồm có:

NONE (các hình ảnh về bàn tay nắm – không ngón tay): 1509 ảnh – 3.82MB

ONE (các hình ảnh về bàn tay với một ngón tay giơ lên với nhiều mẫu): 1510 – 3.97MB

TWO (các hình ảnh về bàn tay với một ngón tay giơ lên với nhiều mẫu): 1515 –4.66MB

THREE (các hình ảnh về bàn tay với một ngón tay giơ lên với nhiều mẫu): 1511 – 5.09MB

FOUR (các hình ảnh về bàn tay với một ngón tay giơ lên với nhiều mẫu): 1526 – 5.38MB

FIVE (các hình ảnh về bàn tay với một ngón tay giơ lên với nhiều mẫu): 1510 – 4.93MB

Các tấm ảnh sẽ được được cắt thành 28x28, với chế độ đen trắng (grayscale) với tất cả 7572 bức cho 6 lớp ngón tay.

Sử dụng các hàm Conv2D, MaxPooling2D, Dropout, Flattern, Dense của Tensorflow.keras để tạo model.

Tiến hành huấn luyện (train) dữ liệu cho ứng dụng bằng model.fit\_generator().

# **5 Các phương thức, hàm**

Hàm createHandHistogram(frame): dùng để tạo biểu đồ tay từ các khung hình từ camera tiếp nhận.

Hàm setupFrame(frame\_width, frame\_heigth): vẽ khung dùng cho việc nhận diện bàn tay

Hàm calculateFingers(contours, contourAndHull): đếm số khuyết tật lồi (convexity defect) để xác định số ngón tay.

Hàm dectHand(frame, handHist): chuyển khung nhận diện tay thành các dạng xóa nền, đen trắng và hiển thị ra một tab riêng để dễ xác định.

Hàm startDetecting(): chạy ứng dụng với vòng while. Bước đầu tiến hành khởi động camera, thiết lập độ lớn cho khung nhận diện bàn tay, chạy vòng lặp while, thiết lập các phím tắt và khởi chạy các hàm ở trên.

# **6 Thử nghiệm / Kết quả / Thảo luận**

Thử nghiệm:

Histogram: Đại diện đồ họa của phân bố tần suất dữ liệu cho các dữ liệu huấn luyện và dữ liệu kiểm thử.

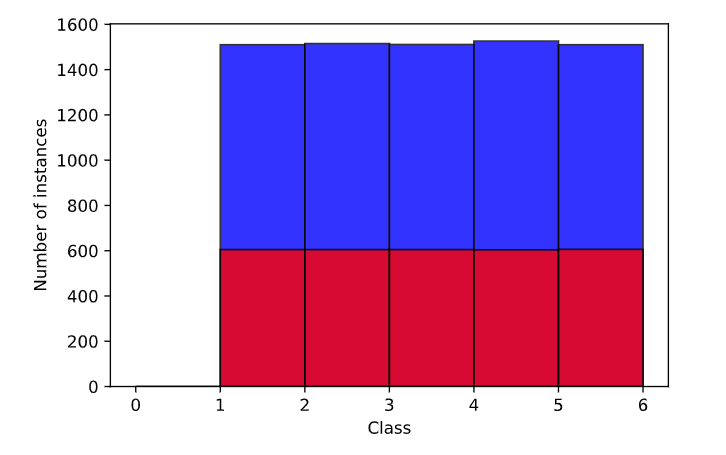


Figure 1Biểu đồ phân bố tần suất dữ liệu

Dữ liệu ảnh huấn luyện được về kích thước 16x16

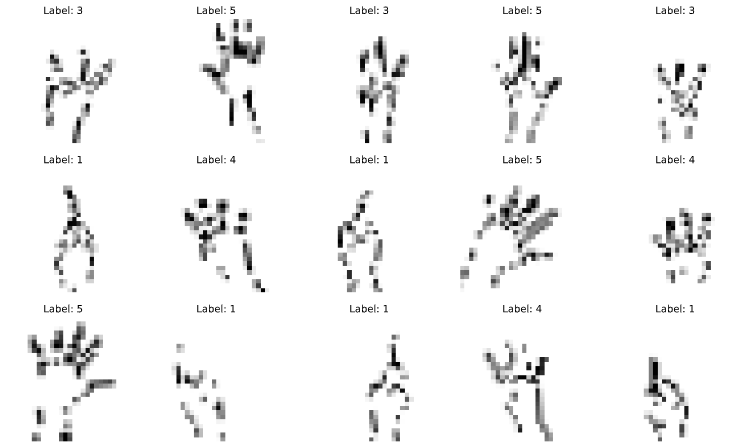


Figure 2 Dữ liệu ảnh huấn luyện

Biểu đồ biểu diễn độ chính xác cho phần huấn luyện dữ liệu:

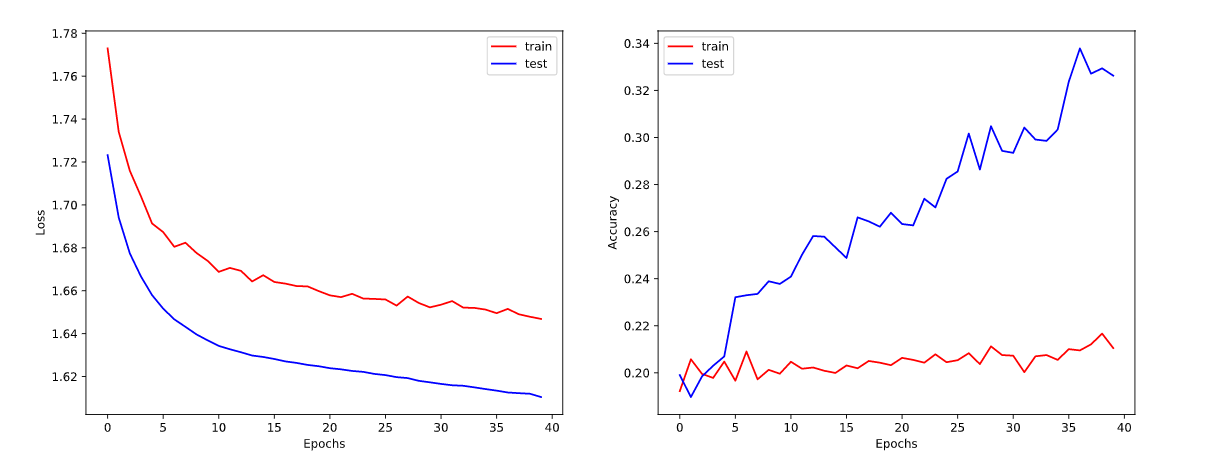


Figure 3 Biểu đồ độ chính xác

Kết quả:

Sử dụng 2 ngón tay để bật phím down (press down) để kéo trang lướt xuống

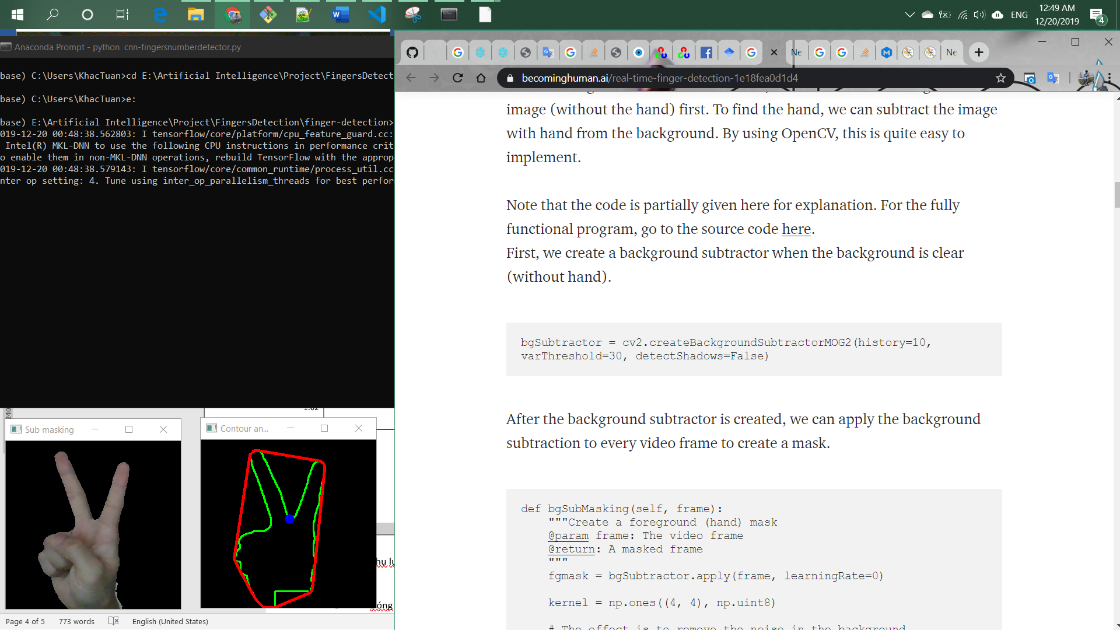


Figure 4 Lướt xuống với 2 ngón tay

Sử dụng 5 ngón tay để bật phím Space để khiến khủng long nhảy

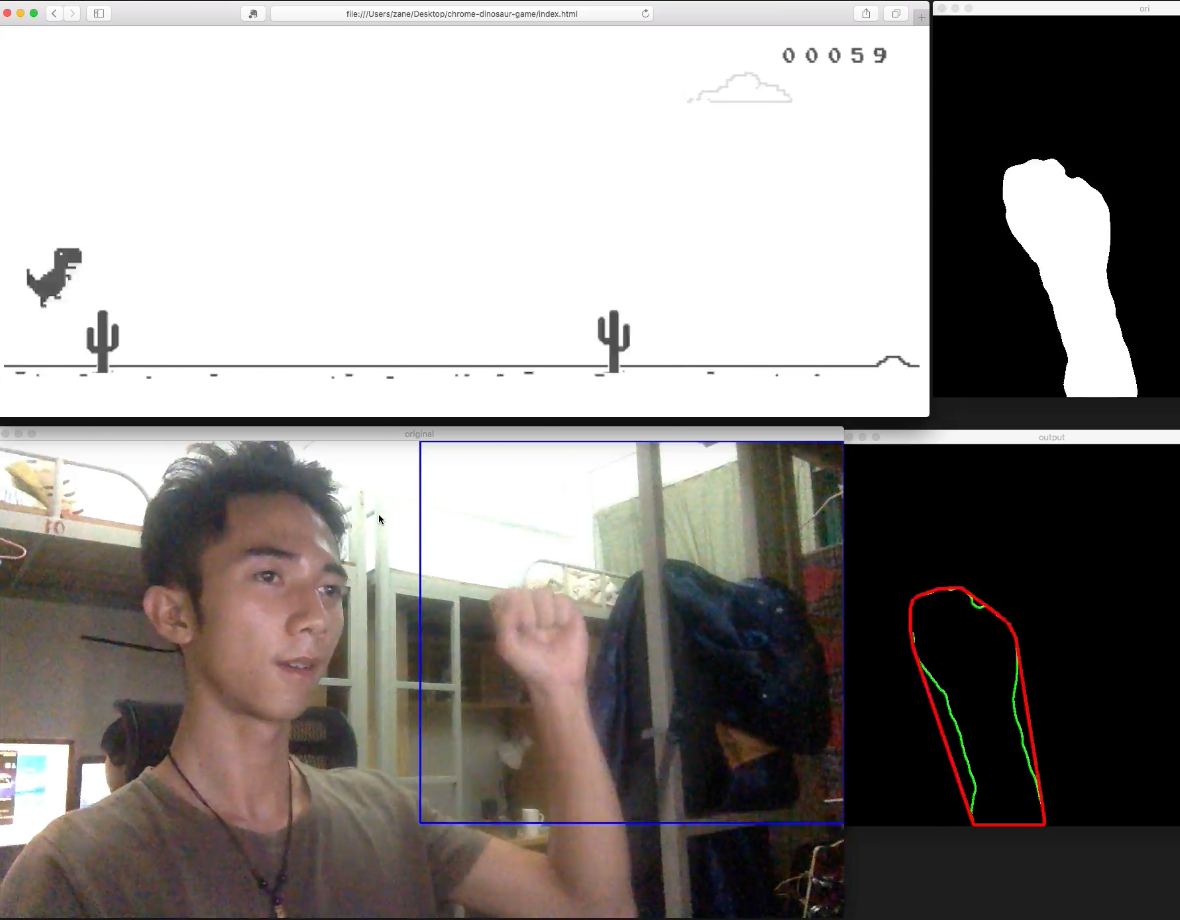


Figure 5 Lướt lên với 3 ngón tay

Thảo luận:

Hiện tại, ứng dụng vẫn còn chưa thực sự nhận diện được một cách chính xác. Khi camera bị lắc, không giữ yên dẫn đến hiện tượng khung hình nhận diên bị nhiễu dẫn đến kết quả đạt được không thực sự chính xác.

# **7 Kết luận / Công việc dự kiến trong tương lai**

Bước đầu đã có thể sử dụng multi threading để tạo ra các luồng giúp cho ứng dụng chạy mượt mà hơn, ứng dụng cũng đã có thể nhận diện được bàn tay và ngón tay trong một số điều kiện nhất định. Ứng dụng cũng đã có thể điều khiển được việc lướt lên, xuống một trang.

Dự kiến trong tương lai: Sẽ tìm hiểu sâu hơn về các thư viện Tensorflow, Keras, OpenCv để có thể hiểu chính xác các hàm đã sử dụng cho ứng dụng nhận diện ngón tay. Từ đó có thể phát triển ứng dụng hoàn chỉnh hơn, có thể nhận diện một cách chính xác trong nhiều điều kiện môi trường khác nhau.

# **8 Phụ lục**

Thuật toán được sử dụng để đếm số khuyết tật lồi (convexity defect) từ đó có thể nhận dạng được số ngón tay.

for i in range(defects.shape[0]):

s,e,f,d = defects[i,0]

start = tuple(contour[s][0])

end = tuple(contour[e][0])

far = tuple(contour[f][0])

a = math.sqrt((end[0] - start[0])\*\*2 + (end[1] - start[1])\*\*2)

b = math.sqrt((far[0] - start[0])\*\*2 + (far[1] - start[1])\*\*2)

c = math.sqrt((end[0] - far[0])\*\*2 + (end[1] - far[1])\*\*2)

angle = (math.acos((b\*\*2 + c\*\*2 - a\*\*2)/(2\*b\*c))\*180)/3.14

# nếu góc> = 90 vẽ một đường tròn tại điểm xa

if angle <= 90:

count\_defects += 1

cv2.circle(crop\_image,far,1,[0,0,255],-1)

cv2.line(crop\_image,start,end,[0,255,0],2)

# **9 Đóng góp**

Nhóm 1 thành viên:

* Công việc đã thực hiện:

+ Tìm hiểu về các dự án về nhận diện ngón tay (fingers detection) tương tự.

+ Tìm hiểu về một số hàm trong các thư viện được sử dụng trong ứng dụng.

+ Chỉnh sửa, thực hiện viết đa luồng (multi threading) cho ứng dụng

# **10 Tài liệu tham khảo**

- <https://github.com/jaredvasquez/CNN-HowManyFingers>

- <https://github.com/lzane/Fingers-Detection-using-OpenCV-and-Python>

- <https://github.com/ChinHuan/finger-detection>

- <https://becominghuman.ai/real-time-finger-detection-1e18fea0d1d4>

- <https://docs.opencv.org/trunk/index.html>

- <https://www.tensorflow.org/>

- <https://machinelearningcoban.com/2018/07/06/deeplearning/>

- <https://www.learnopencv.com/convex-hull-using-opencv-in-python-and-c/>

- <https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.hist.html>

- <https://scikit-learn.org/stable/>

- <https://docs.python.org/3/library/threading.html>

- <https://realpython.com/intro-to-python-threading/>

- <https://docs.python.org/3/library/concurrent.futures.html>

- <https://docs.python.org/3/library/time.html>