

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA KHOA HỌC MÁY TÍNH



ĐỒ ÁN CUỐI KỲ

CÁC KỸ THUẬT HỌC SÂU VÀ ỨNG DỤNG

*Đề tài :*

*Triển khai mô hình dự đoán tuổi và giới tính lên thiết bị di động*

Giảng viên: T.S Nguyễn Vinh Tiệp

Nhóm sinh viên thực hiện:

Trương Đức Vũ - 18520194

Trần Trung Anh - 18530473

Vũ Quý San - 18520143

Lớp: CS431.L21

-- TP.HCM, 06/2021 --

**Mục lục**

**1. Abstract 2**

**2. Giới thiệu 2**

**3. Hướng tiếp cận 3**

**3.1. Bộ dữ liệu 3**

**3.2. Mô hình 4**

**3.2.1. Mô hình dự đoán giới tính 4**

**3.2.2. Mô hình dự đoán tuổi 10**

**3.2.3. Mô hình phát hiện khuôn mặt 10**

**3.3. Tensorflow lite 11**

**3.4. Ứng dụng Android 11**

**4. Kết luận 13**

**5. Tài liệu tham khảo 14**

1. **Abstract**

Trong đồ án này, chúng tôi thực hiện hai mô hình là xác định tuổi và giới tính với mục đích là triển khai trên điện thoại di động android. Ở hai mô hình trên, chúng tôi chọn mạng CNN đơn giản nhưng đủ nhanh và đáp ứng được kết quả tốt. Sử dụng bộ dữ liệu [UTKFace](https://www.kaggle.com/jangedoo/utkface-new) để đào tạo mô hình. Cuối cùng, chúng tôi chuyển model thành dạng TF Lite để phù hợp với cấu hình thiết bị di động.

1. **Giới thiệu**

**Diagram

Description automatically generated**

*Hình 1: Giới thiệu Deep learning trên thiết bị di động*

Deep learning là yếu tố then chốt cho nhiều tiến bộ gần đây trong ứng dụng trí tuệ nhân tạo. Lợi ích lớn hơn sẽ đến khi các công nghệ trí tuệ nhân tạo trở nên phổ biến trong các ứng dụng di động, chẳng hạn như tự động hóa lái xe, robot giá cả phải chăng cho gia đình và hỗ trợ cá nhân thông minh hơn trên điện thoại di động. So với cảm biến di động truyền thống và mô hình điện toán đám mây. Ưu điểm của việc chạy các mô hình trong ứng dụng di động thay vì gửi chúng lên đám mây là giảm độ trễ và khả năng đảm bảo quyền riêng tư dữ liệu cho người dùng. Mặc dù có nhiều thư viện học sâu và các công cụ AI, việc nhúng thành công mô hình học sâu vào ứng dụng di động có thể là một thách thức.

Vì vậy trong đồ án này, chúng tôi sẽ xây dựng một mô hình deep learning với mạng neural network tùy chỉnh giúp giảm số lượng tham số cũng như giảm quy mô của mô hình. Ngoài ra, chúng tôi cũng dùng một số phương pháp để tối ưu mô hình. Cuối cùng, chúng tôi triển khai nó lên điện thoại di động.

1. **Hướng tiếp cận**
   1. **Bộ dữ liệu**

Như yêu cầu đặt ra của bài toán, ta cần một bộ dữ liệu đáp ứng được số lượng cũng như chất lượng cho ba bài toán nhỏ: Phát hiện khuôn mặt, Nhận diện tuổi tác và Nhận diện giới tính. Theo đó, chúng tôi chọn bộ dữ liệu UTKFace.



*Hình 2: Bộ dữ liệu UTKFace*

UTKFace là một dữ liệu lớn bao gồm hơn 20.000 hình ảnh khuôn mặt con người đã được gán nhãn với các thông tin về tuổi tác, giới tính và dân tộc. Bên cạnh đó, các hình ảnh trong bộ dữ liệu này đã được cắt, căn chỉnh vị trí của mỗi khuôn mặt con người, điều này hỗ trợ rất lớn cho bước Phát hiện khuôn mặt của ứng dụng.

Mỗi hình ảnh khuôn mặt con người trong bộ dữ liệu này được gán nhãn trực tiếp trên tên mỗi file, chi tiết như sau:

[age]-[gender]-[race]-[datetime].jpg

Trong đó:

* age: biểu diễn độ tuổi bằng một số nguyên trong khoảng từ 0 đến 116
* gender: biểu diễn giới tính bằng số 0 nếu là Nam, số 1 nếu là Nữ
* race: biểu diễn dân tộc của người trong hình, gồm White, Black, Asian, Indian và Others lần lượt bằng các số 0, 1, 2, 3, 4.
* datetime: biểu diễn thời gian mà bức hình được gán nhãn

Chẳng hạn với một mẫu dữ liệu trong bộ dữ liệu UTKFace,



*Hình 3: Một hình trong tập dữ liệu UTKFace.*

Tên file: 1\_0\_0\_20161219154724341.jpg

Ta trích xuất được các nhãn như sau:

* Độ tuổi [age]: 1
* Giới tính [gender]: 1 - Nam

Dân tộc [race]: 0 – White

* 1. **Mô hình**

Để thực hiện hai mô hình trên Android, một là dự đoán tuổi và mô hình kia là phân loại giới tính. Vì các mô hình này được triển khai trên điện thoại thông minh Android (có sức mạnh tính toán tương đối thấp), vì vậy chúng ta cần nhưng mô hình mà có ít số lượng tham số, do đó thì thời gian suy luận cũng thấp hơn. Bên cạnh đó, mô hình cũng phải tạo ra kết quả khả quan ít nhất là trên tập dữ liệu UTK Face.

Lý do trên thúc đẩy chúng tôi đào tạo trên một kiến trúc neural network tùy chỉnh (custom NN), vì những mô hình hiện có đều có kích thước và số tham số cũng rất lớn vì vậy mà không thể nào tương thích với cấu hình của một điện thoại di động bình thường.

* + 1. **Mô hình dự đoán giới tính**

1. Tổng quan:

Mô hình phân loại giới tính có đầu vào là một ảnh màu 3 kênh RGB kích thước 200x200x3 có gương mặt một người, và đầu ra là một vector hai chiều thể hiện phân phối xác suất của hai nhãn đầu ra: Nam và Nữ. Mô hình có hai thành phần chính nối tiếp nhau: (1) các khối convolution rút trích đặc trưng cấp thấp và (2) các khối Dense rút trích đặc trưng cấp cao (\*).

Diagram

Description automatically generated

*Hình 4: Input -> Convolution Blocks -> Dense Blocks -> Output*

(\*) Lưu ý: Trong bài này, các khối convolution không chỉ gồm các phép convolution mà còn có kĩ thuật batch normalization, hàm kích hoạt leaky relu, ... Tương tự vậy với các khối Dense.

1. Thiết kế mô hình:

Khối convolution:

Để có thể thực hiện phân loại giới tính chạy trên các thiết bị di động, mục tiêu chính của chúng tôi là thiết kế được một mạng CNNs (cơ bản) với số lượng tham số ít hơn nhưng vẫn cố gắng đảm bảo độ chính xác của mô hình.

Theo đó, mỗi khối Convolution bao gồm một chuỗi các layer sau:

Conv2D BatchNorm LeakyReLU

Lớp Conv2D ở đây có thể sử dụng Convolution cơ bản, tuy nhiên với mục tiêu giảm bớt số lượng tham số, chúng tôi thêm một tùy chọn để có thể thay thế Convolution cơ bản bằng lớp Separable Convolution.

Ý tưởng chính của lớp Separable Convolution là phân tách kernel gốc thành các kernel nhỏ hơn, từ đó thực hiện ít phép tính hơn. Chẳng hạn với kernel quen thuộc Sobel, ta có có thực hiện như sau:

A picture containing diagram

Description automatically generated

Từ đó, thay vì thực hiện Convolution 9 phép tính thì chúng ta chỉ cần thực hiện 6 phép tính nên mô hình sẽ giảm được đáng kể thời gian huấn luyện và độ phức tạp.

BatchNorm (Batch Normalization) là một kỹ thuật chuẩn hóa các mini-batch đầu vào của một layer từ đó giúp ổn định và giảm thời gian huấn luyện mô hình.

//Hàm activation Leaky ReLU giúp tránh trường hợp Dying-ReLU khi mà có một số dữ liệu đầu vào làm cho các đơn vị ReLU không hoạt động.

Chúng tôi chọn hàm kích hoạt là leaky relu thay vì chọn relu như thông thường bởi vì leaky relu giúp giải quyết tình trạng dying-relu trong quá trình huấn luyện mô hình. Dying-relu là hiện tượng đơn vị Relu luôn cho đầu ra là 0 trên mọi trường hợp dữ liệu đầu vào. Đầu ra bằng 0 dẫn đến đạo hàm bằng 0 nên gradient descent không cập nhật lại được giá trị các tham số gây ra hiện tượng dying-relu này.

Diagram

Description automatically generated

*Hình 5: ReLU và Leaky-ReLU*

ReLU (trái) và Leaky-ReLU (phải). Khi đầu vào là số âm thì Leaky-ReLU cho đầu ra là một số âm, dẫn đến đạo hàm khác 0 nên các tham số liên quan có thể được cập nhật lại.

Khối Dense:

Sau khối Convolution, chúng tôi định nghĩa thêm khối Dense để huấn luyện mô hình bằng các đặc trưng trích xuất từ khối Convolution.

Dense -> Dropout -> LeakyReLU

Khối Dense bao gồm một fully-connected layer theo sau là hàm kích hoạt Leaky-ReLU và áp dụng thêm kĩ thuật dropout nhằm giảm overfitting và tăng khả năng tổng quát hóa của mô hình. Khi dùng dropout thì trong khi huấn luyện mô hình sẽ ngẫu nhiên bỏ qua một số đơn vị trong mạng nơ-ron của mô hình. Ở lớp Dropout, chúng tôi cài đặt tham số rate giảm dần để phù hợp với quá trình huấn luyện ở lớp Dense khi mà số lượng đơn vị tính toán cũng giảm dần sau mỗi epoch.

Diagram

Description automatically generated

*Hình 6: Kỹ thuật Dropout*

Cuối cùng, chúng tôi thêm một lớp Dense có hai đơn vị và áp dụng hàm activation softmax để tính toán phân phối cho hai nhãn đầu ra: Nam và Nữ.

1. Huấn luyện mô hình:

Để thực hiện quá trình huấn luyện mô hình Nhận diện giới tính, chúng tôi tiến hành cài đặt mô hình trên với các thông tin sau:

- Hàm loss là CategoricalCrossentropy

- Thuật toán tối ưu tham số Adam.

- Độ đo Accuracy để đánh giá mô hình

- Một số hàm callback để lưu lại mô hình tại một số thời điểm huấn luyện.

- Áp dụng hai kỹ thuật là Early Stopping và Learning rate schedule:

Early Stopping: kỹ thuật này giúp dừng việc train khi mô hình bắt đầu overfit dữ liệu train. Cụ thể, sau p epoch liên tiếp mà độ lỗi trên tập validation không giảm thì dừng train mô hình. Hyperparameter p ở đây trong early stopping thường được gọi là patience.

Diagram

Description automatically generated

*Hình 7: Kỹ thuật early stopping*.

Learning rate schedule: kỹ thuật này thay đổi learning rate trong quá trình train mô hình. Thực nghiệm cho thấy so với việc giữ nguyên learning rate thì thay đổi learning rate thường giúp mô hình đạt kết quả tốt hơn đồng thời giảm ảnh hưởng của giá trị khởi tạo learning rate đến kết quả train mô hình. Có nhiều cách để thay đổi learning rate (tăng dần, giảm dần, tăng dần đến ngưỡng nhất định rồi giảm dần, v.v ), chúng tôi chọn cách giảm dần learning rate sau mỗi lần cập nhật tham số.

Chart, line chart

Description automatically generated

*Hình 8: Một số dạng learning rate schedule*.

A collage of a person's face

Description automatically generated with medium confidence

*Hình 9: Một số kết quả của mô hình dự đoán giới tính*

* + 1. **Mô hình dự đoán tuổi**

Mô hình dự đoán tuổi có kiến trúc và cách huấn luyện giống như mô hình phân loại giới tính ngoại trừ các điểm sau:

Ở cuối mô hình là một lớp Dense có một đơn vị và hàm kích hoạt là ReLU để dự đoán tuổi.

Độ đo mean square error để đánh giá mô hình.

Hàm loss là mean square error.

A collage of a person

Description automatically generated with medium confidence

*Hình 10: Một số kết quả của mô hình dự đoán độ tuổi*

* + 1. **Mô hình phát hiện khuôn mặt**

ML Kit là một SDK di động mang kiến ​​thức chuyên môn về máy học của Google cho các ứng dụng Android và iOS trong một gói mạnh mẽ nhưng dễ sử dụng. Cho dù bạn là người mới hay đã có kinh nghiệm về học máy, bạn có thể triển khai chức năng mình cần chỉ trong một vài dòng mã. Không cần phải có kiến ​​thức sâu về mạng nơ-ron hoặc tối ưu hóa mô hình để bắt đầu. Mặt khác, nếu bạn là một nhà phát triển máy học có kinh nghiệm, ML Kit cung cấp các API tiện lợi giúp bạn sử dụng các mô hình TensorFlow Lite tùy chỉnh trong các ứng dụng dành cho thiết bị di động của mình. Vì vậy chúng tôi gọi ML Kit Face Detection để dùng luôn.

* 1. **Tensorflow Lite**

TensorFlow Lite là một bộ công cụ cho phép học máy trên thiết bị bằng cách giúp các nhà phát triển chạy mô hình của họ trên các thiết bị di động, nhúng và IoT. Vì vậy, chúng tôi quyết định sử dụng mô hình của TensorFlow Lite

Mô hình TensorFlow Lite được thể hiện ở định dạng di động hiệu quả đặc biệt. Nó cung cấp một số lợi thế so với định dạng mô hình thường của TensorFlow như giảm kích thước ít tham số, suy luận nhanh hơn mà không làm giảm đáng kể hiệu suất của mô hình. Điều đó, cho phép mô hình TensorFlow Lite thực thi hiệu quả trên các thiết bị có tài nguyên bộ nhớ và tính toán hạn chế như thiết bị điện thoại di động.

|  |  |  |
| --- | --- | --- |
| Mô hình suy luận tuổi | model | Lite model |
| Số lượng tham số | **1.069.297** | **200.956** |
| MAE | **0.0205** | **0.0458** |

*Bảng 1: So sánh 2 mô hình suy luận tuổi.*

|  |  |  |
| --- | --- | --- |
| Mô hình phân loại giới tính | model | Lite model |
| Số lượng tham số | **675.090** | **328.733** |
| Accuracy (%) | **99.8** | **96.4** |

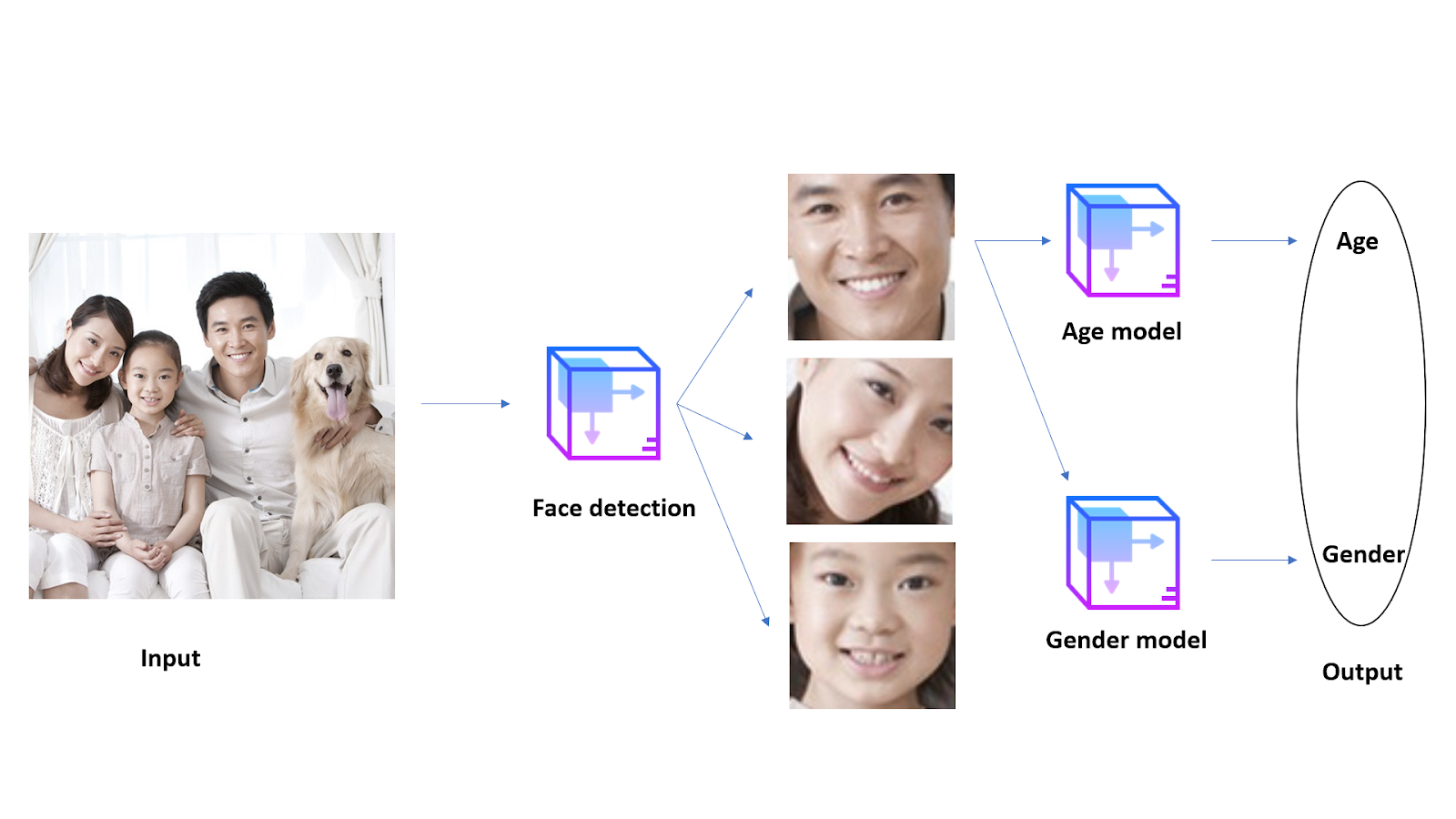
*Bảng 2: So sánh 2 mô hình phân loại giới tính.*

* 1. **Ứng dụng Android**

Để trực quan hóa, cũng như cho mọi người có thể sử dụng hai mô hình trên (kể cả người không biết gì về lập trình), chúng tôi đã triển khai mô hình lên một ứng dụng trên Android có tên là “Đoán tuổi-giới tính”.

Chúng tôi sử dụng ngôn ngữ lập trình Kotlin. Giống với Java, Kotlin chạy trên Java Virtual Machine và sử dụng được các công cụ và thư viện có sẵn trên Java. Kotlin có code ngắn gọn, giảm thiểu các đoạn không cần thiết, cấu trúc ngắn, trực quan, dễ học, dễ làm, dễ sử dụng, tiết kiệm thời gian. Đó cũng chính là lý do chúng tôi lựa chọn Kotlin để viết ứng dụng Android.

Ứng dụng “Đoán tuổi-giới tính” có một số chức năng như tùy chỉnh mô hình trước khi suy luận, nút chụp ảnh, nút tải ảnh lên và khoảng để hiển thị hình ảnh và kết quả.

*Hình 11: Cấu trúc triển khai mô hình trên thiết bị di động.*

Để suy luận ra được độ tuổi và giới tính tương ứng. 

* Bước thứ nhất, hình ảnh chụp được hoặc hình ảnh tải lên được cho qua model Face Detection, kết quả đầu ra là một loạt khuôn mặt được, lấy khuôn mặt đầu tiên cắt ra rồi làm đầu vào của bước hai.
* Bước thứ hai, ảnh khuôn mặt được detect ở bước trước được cho vào hai model và thực hiện suy luận tuổi, giới tính.
* Bước cuối cùng là hiển thị kết quả như hình dưới. (Tuổi đúng của huấn luận viên Park Hang-seo là 63)

Trong phần xây dựng ứng dụng android, chúng tôi cũng đã tìm hiểu về GPU trên android nhưng do kết qua khi có GPU và không có GPU không có khác biệt gì nhiều. Bên cạnh đó, khi sử dụng GPU thì ứng dụng nhanh bị thoát ra hơn vì tràn RAM. Vì vậy, chúng tôi quyết định không thêm phần GPU khi suy luận nữa.

*Hình 12: Kết quả dự đoán trên điện thoại.*

1. **Kết luận**

Trong đồ án này, chúng tôi đã xây dựng hai mạng neural network tùy chỉnh để có thể giảm được tham số của mô hình. Bên cạnh đó, chúng tôi sử dụng Tensorflow Lite để chuyển mô hình từ định dạng .h5 sang .tflite là định dạng thích hợp có thể chạy trên thiết bị di động. Và cuối cùng, chúng tôi sử dụng ngôn ngữ lập trình kotlin để xây dựng ứng dụng “Đoán tuổi – giới tính”. Ứng dụng cho phép chúng ta suy đoán ra độ tuổi và giới tính của một người từ ảnh đầu vào. Và để chạy được trơn tru trên máy điện thoại.

Ngoài những kết quả đạt được như trên, cũng có một số khó khăn và vấn đề chưa giải quyết được. Để đảm bảo cả ba mô hình chạy ổn định trên điện thoại nên mỗi lần chúng tôi chỉ suy luận một gương mặt duy nhất từ ảnh đầu vào. Và một điều nữa là kết quả chưa đúng khá nhiều. Điều này có vẻ là do bộ dữ liệu chưa sát với dữ liệu thực tế gây ra tình trạng overfitting.

1. **Tài liệu tham khảo**
   * 1. [A Basic Introduction to Separable Convolutions | by Chi-Feng Wang | Towards Data Science](https://towardsdatascience.com/a-basic-introduction-to-separable-convolutions-b99ec3102728)
     2. [UTKFace Dataset | Kaggle](https://www.kaggle.com/jangedoo/utkface-new)
     3. [A Gentle Introduction to Batch Normalization for Deep Neural Networks | by Jason Brownlee | Machine Learning Mastery](https://machinelearningmastery.com/batch-normalization-for-training-of-deep-neural-networks/)
     4. [CS231n Convolutional Neural Networks for Visual Recognition Course Notes | Stanford CS](http://cs231n.github.io/neural-networks-1)
     5. [Dropout in (Deep) Machine learning | by Amar Budhiraja | Medium](https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5)
     6. [Bài 15: Overfitting | by Vũ Hữu Tiệp | Machine Learning cơ bản](https://machinelearningcoban.com/2017/03/04/overfitting/#-early-stopping)
     7. [Keras learning rate schedules and decay | by Adrian Rosebrock | PyImageSearch](https://www.pyimagesearch.com/2019/07/22/keras-learning-rate-schedules-and-decay/)