UNIVERSITY OF INFORMATION TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE

# FINAL REPORT

## Deep learning techniques and applications

## Efficiency of MobileNet model families for ASL Gesture Classification

Ho Chi Minh City, 06/2025

# Acknowledgement

Nhóm chúng tôi xin gửi lời cảm ơn chân thành đến thầy **Nguyễn Vinh Tiệp** và thầy **Trần Gia Nghia**, giảng viên khoa Khoa học máy tính, trường Đại học Công nghệ thông tin, Đại học Quốc gia Thành phố Hồ Chí Minh. Với tư cách là giảng viên của môn học **CS431.P22**, sự giảng dạy tận tâm và phản hồi sâu sắc của các thầy đã vô cùng quý giá đối với sự thành công của dự án.

Trong suốt quá trình hoàn thành dự án này, chúng tôi đã nỗ lực hết mình để mang lại kết quả tốt nhất có thể. Tuy nhiên, bất chấp những nỗ lực của chúng tôi, một số lỗi vô ý có thể đã xảy ra. Chúng tôi trân trọng yêu cầu phản hồi thẳng thắn và những đề xuất mang tính xây dựng từ giáo sư để giúp chúng tôi cải thiện hơn nữa công việc của mình.

Nếu có bất kỳ câu hỏi hoặc phản hồi nào, vui lòng liên hệ với chúng tôi qua email theo địa chỉ sau:

- 22521333@gm.uit.edu.vn (Nguyễn Duy Thắng)

Chúng tôi rất trân trọng thời gian và sự hướng dẫn của bạn!

# Mục lục

# 1 Giới thiệu

## 1.1 Lý do chọn đề tài

Ngôn ngữ ký hiệu (Sign Language) là công cụ giao tiếp quan trọng đối với cộng đồng người khiếm thính và người câm, trong đó American Sign Language (ASL) là một trong những hệ thống ngôn ngữ được sử dụng rộng rãi trên thế giới (chủ yếu sẽ ở Hoa Kỳ và nhiều nơi ở Canada) [12].

Trong lĩnh vực thị giác máy tính, các mô hình học sâu hiện đại như ResNet [5], EfficientNet [18] hay Transformer-based models [20, 3] đã đạt độ chính xác cao trong bài toán phân loại ảnh, trong đó có nhận diện ASL. Tuy nhiên, các mô hình này thường có số lượng tham số lớn và đòi hỏi tài nguyên tính toán cao, gây khó khăn trong việc triển khai trên thiết bị di động hoặc nhúng.

Ngược lại, dòng mô hình MobileNet [7] được thiết kế đặc biệt cho môi trường tài nguyên hạn chế, như điện thoại thông minh, IoT, hoặc hệ thống nhúng.

Đề tài "Efficiency of MobileNet model families for ASL Gesture Classification" nhằm phân tích, so sánh khả năng cân bằng giữa độ chính xác và hiệu năng của ba mô hình tiêu biểu: MobileNetV2 [15], MobileNetV3-Small [6] và MobileNetV4-Conv-Small [14].

## 1.2 Problem - Input - Output

**Định nghĩa bài toán** Xây dựng một hệ thống phân loại cử chỉ tay tĩnh trong ngôn ngữ ký hiệu Mỹ (ASL), sử dụng hình ảnh RGB đầu vào, với yêu cầu chỉ nhận diện các ảnh có chứa bàn tay. Hệ thống cần phân loại ảnh đầu vào thành một trong 28 lớp ASL, bao gồm 26 chữ cái (A–Z) và hai ký hiệu chức năng (del, space). Các ảnh không có bàn tay sẽ bị loại bỏ khỏi bài toán nhằm đảm bảo tính nhất quán và tính khả thi trong việc nhận dạng bằng hình ảnh cử chỉ tay.

**Input**

- 1 ảnh số hình ảnh bàn tay.
- 1 dataset gồm các ảnh bàn tay đã được gán nhãn thuộc 1 trong 28 lớp (A-Z, del, space).

**Output**

- Nhãn (label) của ảnh số đầu vào.

**Constrains**

- Ảnh số đầu vào phải có bàn tay.
- Kích thước bàn tay tối thiểu là 128x128 pixel.
- Có tối đa 1 bàn tay trong ảnh đầu vào.

**Requirements** Mô hình phân lớp cần có:

- Có kích thước nhỏ ( <5MB).
- Tốc độ xử lý nhanh ( <1s/ảnh).

## 1.3 Mục tiêu - Phạm vi nghiên cứu

**Mục tiêu nghiên cứu**

- So sánh hiệu quả và đặc điểm của các phiên bản MobileNet trong bài toán phân loại ảnh bàn tay ASL, tập trung vào độ chính xác, tốc độ xử lý và kích thước mô hình.
- Phát triển mô hình phân loại có kích thước nhỏ gọn phù hợp cho việc triển khai trên các thiết bị di động hoặc thiết bị nhúng với giới hạn tài nguyên.

**Phạm vi nghiên cứu**

- Nghiên cứu tập trung vào phân loại ảnh tĩnh chứa một bàn tay duy nhất với kích thước tối thiểu 128x128 pixel, thuộc một trong 28 lớp ký hiệu ASL.
- Không mở rộng sang nhận dạng cử chỉ từ video hay chuỗi động tác liên tục.
- Giới hạn dữ liệu đầu vào là các ảnh đã được tiền xử lý, đảm bảo có tối đa một bàn tay trong ảnh và có đầy đủ nhãn.

# 2 Tổng quan lý thuyết

## 2.1 Ngôn ngữ ký hiệu American Sign Language

American Sign Language (ASL) là một ngôn ngữ tự nhiên thuộc hệ ngôn ngữ thủ công, được sử dụng bởi cộng đồng người khiếm thính tại Hoa Kỳ và Canada. Đây không phải là bản dịch trực tiếp của tiếng Anh mà có cấu trúc cú pháp, ngữ pháp, và các quy tắc riêng biệt [17, 19].

ASL kết hợp các yếu tố vận động như hình dạng bàn tay (handshape), hướng tay, chuyển động, vị trí tương đối với cơ thể, và biểu cảm khuôn mặt. Trong bài toán phân loại ảnh tĩnh, nghiên cứu này tập trung vào nhận diện bảng chữ cái ASL thông qua fingerspelling, tức là sử dụng bàn tay để đánh vần từng chữ cái [1].

Đặc điểm của ASL fingerspelling trong bài toán:

- Gồm 26 ký hiệu tay tương ứng với các chữ cái tiếng Anh.
- Bổ sung thêm một số ký hiệu chức năng phổ biến như "space", "delete".
- Các ký hiệu fingerspelling mang tính hình học cao – chỉ cần sự khác biệt nhỏ ở một ngón tay là có thể phân biệt thành ký hiệu khác.

Thách thức trong nhận diện ASL từ ảnh:

- Biến thiên cá nhân: tay mỗi người có kích cỡ, màu da, hình dạng khác nhau – gây ảnh hưởng đến khả năng học mô hình.
- Nhiễu hình ảnh: ánh sáng phức tạp, phông nền không đồng nhất, các vật thể lạ có thể ảnh hưởng đến mô hình phân loại.
- Biến thể biểu diễn: cùng một ký hiệu nhưng người thể hiện có thể xoay góc tay, đưa tay gần/xa camera, tạo ra sự khác biệt không mong muốn.

Ứng dụng thực tế:

- Công cụ hỗ trợ giáo dục ngôn ngữ ký hiệu.
- Ứng dụng chuyển đổi ASL thành văn bản/thông điệp nói.
- Nền tảng cho các hệ thống giao tiếp người–máy, đặc biệt trong các môi trường cần tương tác không lời (robot hỗ trợ, giao diện thông minh, v.v.).

Việc nhận diện chính xác fingerspelling là bước đầu tiên quan trọng trong xây dựng hệ thống nhận diện ASL hoàn chỉnh. Trong bối cảnh nghiên cứu này, ảnh tĩnh là lựa chọn phù hợp vì có thể dễ dàng thu thập, đánh nhãn, và huấn luyện mô hình trong điều kiện hạn chế về dữ liệu và tài nguyên tính toán.

## 2.2 Mô hình học sâu trong phân loại ảnh

Phân loại ảnh là một trong những bài toán nền tảng trong thị giác máy tính. Nhiệm vụ chính của nó là gán nhãn chính xác cho ảnh đầu vào dựa trên nội dung thị giác. Các phương pháp truyền thống dựa vào đặc trưng thủ công (handcrafted features) như SIFT, HOG, hoặc SURF có độ chính xác giới hạn do thiếu khả năng tổng quát và thích nghi với dữ liệu đa dạng [11, 2].

Sự xuất hiện của mạng nơ-ron tích chập (Convolutional Neural Network – CNN) đã thay đổi hoàn toàn cách tiếp cận bài toán này. CNN học trực tiếp các đặc trưng từ dữ liệu đầu vào thông qua quá trình huấn luyện, từ đó tự động tối ưu hóa việc trích xuất đặc trưng, giảm thiểu nhu cầu tiền xử lý thủ công [10, 9].

Các kiến trúc CNN tiêu biểu:

- LeNet-5: Một trong những mạng CNN đầu tiên, thiết kế bởi LeCun cho bài toán nhận dạng chữ viết tay [10].
- AlexNet: Được xem là cột mốc bùng nổ của deep learning trong thị giác máy tính sau chiến thắng tại ImageNet 2012 [9].
- VGGNet: Mở rộng chiều sâu mạng bằng cách sử dụng nhiều lớp tích chập 3×3, mang lại hiệu năng mạnh mẽ nhưng tiêu tốn tài nguyên [16].
- ResNet: Đưa ra khái niệm residual connection, giúp huấn luyện mạng sâu dễ dàng hơn và tránh hiện tượng suy giảm gradient [5].

- EfficientNet: Sử dụng chiến lược compound scaling để cân bằng giữa chiều sâu, chiều rộng và độ phân giải ảnh, đạt hiệu năng cao với số tham số thấp [18].

Xu hướng mới: Transformer trong thị giác máy tính Các mô hình Transformer, vốn xuất phát từ xử lý ngôn ngữ tự nhiên (NLP), đã được mở rộng sang thị giác với kiến trúc Vision Transformer (ViT) [3]. ViT chia ảnh thành các patch nhỏ và xử lý như chuỗi, cho phép mô hình học các quan hệ toàn cục mà không cần tích chập.

Tuy nhiên, các kiến trúc như ViT, Swin Transformer hay DeiT thường có lượng tham số rất lớn, đòi hỏi phần cứng mạnh và dữ liệu huấn luyện lớn. Do đó, chúng chưa thực sự phù hợp với các bài toán nhúng hoặc triển khai thực tế trên thiết bị di động.

## 2.3 Kiến trúc MobileNet

MobileNet là dòng kiến trúc CNN được thiết kế với tiêu chí nhẹ, nhanh và hiệu quả,phù hợp cho các thiết bị di động và nhúng. Các phiên bản kế tiếp của MobileNet liên tục cải tiến về hiệu suất tính toán, độ chính xác, cũng như khả năng tối ưu hóa trên phần cứng thực tế. với sự khác nhau chủ yếu sau:

- MobileNetV1 : Depthwise Separable Convolution (đã được học trên lớp) [7].
- MobileNetV2 : Inverted Residual và Linear Bottleneck [15].
- MobileNetV3 : NAS, Swish và SE block [6].
- MobileNetV4 : Universal Inverted Bottlenecks [14].

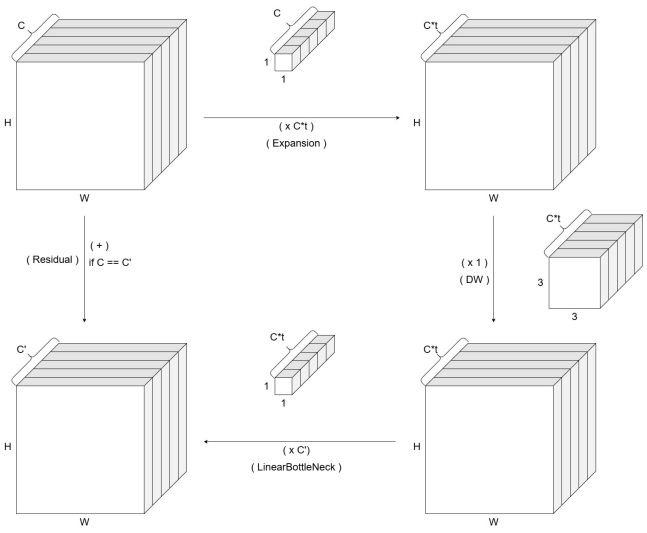
### 2.3.1 MobileNetV2

MobileNetV2 là phiên bản cải tiến của MobileNetV1 được giới thiệu bởi Sandler và cộng sự vào năm 2018, nhằm khắc phục hạn chế về khả năng biểu diễn của kiến trúc gốc và tăng hiệu năng trên các thiết bị di động [15].Mô hình được xây dựng trên hai thành phần chính:

**Inverted Residual Block 1** Khác với kiến trúc residual truyền thống (ResNet), nơi đặc trưng đầu vào được nén lại và sau đó mở rộng, MobileNetV2 đảo ngược quy trình này – tức là:

- Mở rộng chiều bằng 1×1 convolution (gọi là expansion layer), thường nhân số kênh lên từ 6 đến 10 lần.
- Áp dụng Depthwise Separable Convolution (DSC) – một phép tích chập nhẹ chỉ áp dụng trên mỗi kênh riêng biệt, giúp giảm đáng kể số tham số và FLOPs.
- Thu hẹp chiều về số kênh gốc bằng một lớp 1×1 khác (projection layer).

Quá trình này được gọi là Inverted Bottleneck vì nó ngược lại với Bottleneck Block trong ResNet.




Hình 1: Inverted Residual Block in MobileNetV2

**Linear Bottleneck và Skip Connection** Ở cuối block, MobileNetV2 không sử dụng hàm kích hoạt ReLU (trái với các kiến trúc CNN truyền thống). Thay vào đó, lớp projection giữ đầu ra linear, giúp tránh mất thông tin ở những đặc trưng có giá trị nhỏ [15]. Hơn nữa, nếu đầu vào và đầu ra có cùng số kênh, kiến trúc sẽ sử dụng skip connection để cộng trực tiếp tín hiệu đầu vào với đầu ra (như trong ResNet [5]), giúp tăng khả năng truyền gradient và học tốt hơn.

### 2.3.2 MobileNetV3-Small

MobileNetV3 là phiên bản tiếp theo trong dòng kiến trúc MobileNet, được giới thiệu bởi Howard và cộng sự vào năm 2019 [6]. Điểm nổi bật của phiên bản này là việc kết hợp tự động thiết kế kiến trúc bằng NAS (Neural Architecture Search) cùng với các kỹ thuật tối ưu sâu đã được kiểm chứng, như:

- Squeeze-and-Excitation (SE) block.
- Hard-Swish (h-swish) activation.
- Efficient depthwise separable convolutions.

Phiên bản MobileNetV3-Small được thiết kế chuyên biệt cho các tác vụ có yêu cầu tốc độ cao và tài nguyên hạn chế (low-latency), rất phù hợp cho các thiết bị di động và nhúng.

**Neural Architecture Search (NAS)** Khác với MobileNetV2 được thiết kế thủ công, MobileNetV3 sử dụng platform-aware NAS, nghĩa là quá trình thiết kế kiến trúc được tối ưu không chỉ cho độ chính xác mà còn dựa trên các yếu tố thực tế như:

- Độ trễ khi triển khai trên vi xử lý cụ thể (ARM CPU, DSP).
- Kích thước mô hình.
- Số FLOPs

Do đó, MobileNetV3-Small có kiến trúc bất đối xứng và nhiều tầng được thiết kế không đồng nhất – khác với V2 vốn có dạng block lặp đều.

**Squeeze-and-Excitation (SE) Block 2** MobileNetV3 tích hợp SE block vào trong hầu hết các residual unit. SE block hoạt động theo cơ chế attention theo chiều kênh:

- Tính trung bình theo không gian (global average pooling).
- Nén và mở rộng đặc trưng (bottleneck FC layers).
- Áp dụng hàm kích hoạt sigmoid để tạo "mặt nạ"chú ý (attention map).
- Nhân hệ số này lên từng kênh đầu vào.

Mục tiêu là giúp mô hình tự học được kênh nào quan trọng, kênh nào nên giảm tầm ảnh hưởng [8].

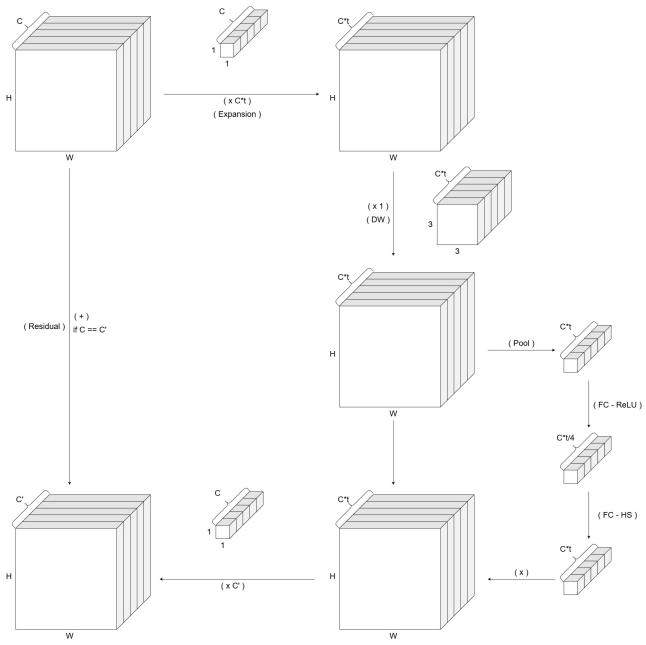
**Hàm kích hoạt Hard-Swish** MobileNetV3 thay thế ReLU/ReLU6 trong MobileNetV2 bằng h-swish – một hàm kích hoạt phi tuyến liên tục nhưng dễ tính toán hơn so với swish gốc. Hàm này có các đặc điểm:

- Tính liên tục và khả vi (smooth), giúp mô hình học hiệu quả hơn.
- Tính toán đơn giản và thân thiện với phần cứng (dễ tối ưu trên mobile CPU).

$$\text{h-swish}(x) = x \cdot \frac{\text{ReLU6}(x+3)}{6}, \quad \text{ReLU6}(x) = \min(\max(0, x), 6)$$

### 2.3.3 MobileNetV4-Conv-S

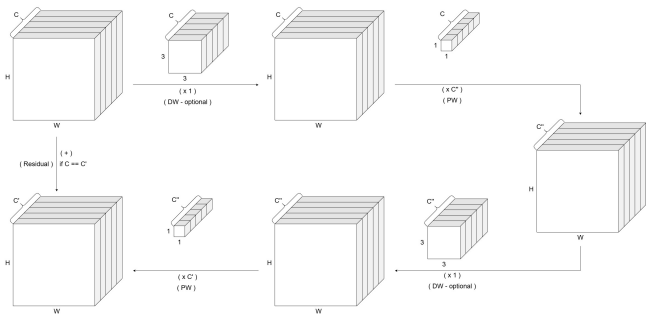
MobileNetV4-Conv-Small là một trong các biến thể hiện đại thuộc thế hệ kế tiếp của dòng mô hình MobileNet, hướng tới việc tối ưu hóa hiệu năng suy luận và khả năng triển khai thực tế trên thiết bị biên (edge devices). Dù chưa có bản phát hành chính thức như các phiên bản trước (V2, V3), kiến trúc này được xây dựng dựa trên tư tưởng kế thừa và mở rộng của các thành phần như Inverted Bottleneck, ConvNext, Feed-Forward Network (FFN) và các khối tích chập sâu đặc biệt [14].



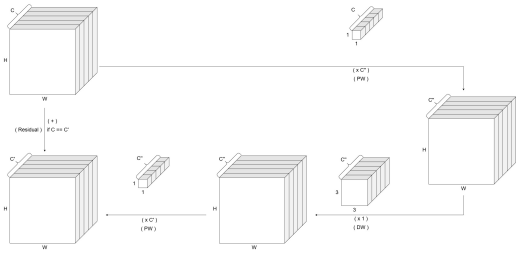

Hình 2: Squeeze-and-Excitation (SE) Block in MobileNetV3

**Universal Inverted Bottleneck (UIB) Block 3** Trái với Inverted Bottleneck cố định trong MobileNetV2, khối UIB (Universal Inverted Bottleneck) trong MobileNetV4 mang tính cấu hình linh hoạt cao hơn, được thiết kế để chọn lựa giữa nhiều cấu trúc con khác nhau, nhằm đạt được cân bằng giữa tốc độ suy luận và độ chính xác. Các biến thể của UIB block bao gồm:

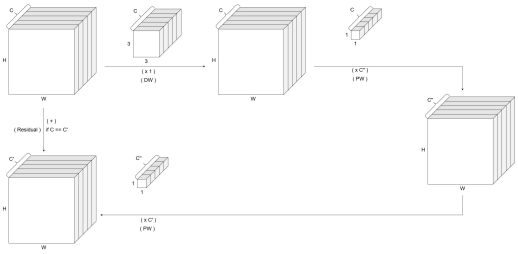
- Inverted Bottleneck (IB) 4: Cấu trúc mở rộng → Depthwise Convolution → nén lại (projection), giống như V2 nhưng có tinh chỉnh về nonlinear activation và batch norm.
- ConvNext-like Block 5: Trộn không gian (spatial mixing) trước khi mở rộng số kênh, giúp mô hình học được đặc trưng không gian mạnh hơn, đặc biệt hữu ích khi có kernel lớn như 7×7.
- Extra Depthwise (ExtraDW) 6: Thêm một tầng depthwise convolution phụ để tăng độ sâu hiệu dụng mà không tăng số lượng tham số đáng kể.
- Feed-Forward Network (FFN) 7: Bao gồm hai tầng 1×1 convolution, tương tự như block MLP trong Transformer, có ưu điểm là khả năng song song hóa cao và tương thích tốt với phần cứng như NPU hoặc TPU.
- FuseIB 8: Kết hợp một lớp 2D convolution kernel lớn k×k với một lớp 1×1 convolution, nhằm thực hiện mở rộng kênh và trộn không gian trong một bước duy nhất.



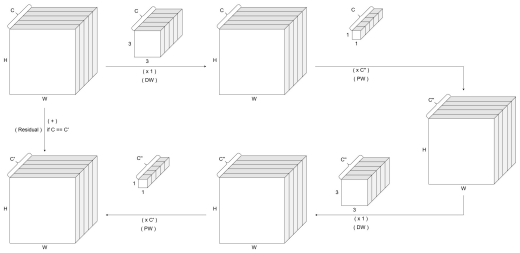

Hình 3: Universal Inverted Bottleneck (UIB) Block in MobileNetV4



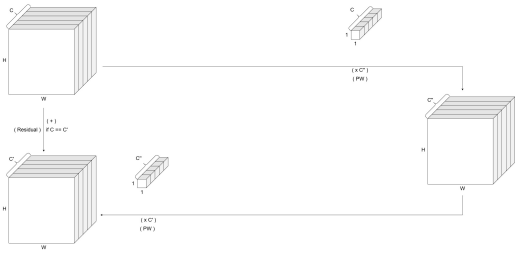
Hình 4: Inverted Bottleneck of UIB



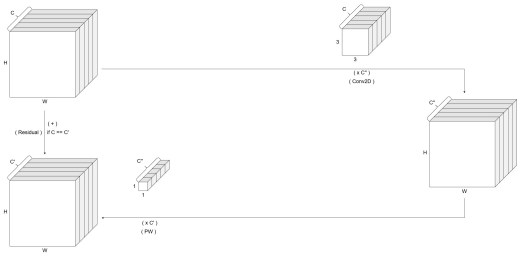
Hình 5: ConvNext-like Block of UIB



Hình 6: Extra Depthwise (ExtraDW) Block of UIB



Hình 7: Feed-Forward Network (FFN) Block of UIB



Hình 8: FuseIB Block of UIB

# 3 Phương pháp nghiên cứu

## 3.1 Dataset

Việc lựa chọn và xử lý tập dữ liệu đầu vào đóng vai trò quyết định trong hiệu quả huấn luyện mô hình học sâu. Trong nghiên cứu này, ba bộ dữ liệu khác nhau đã được sử dụng để đảm bảo mô hình học được các đặc trưng phong phú và tổng quát hóa tốt:

**ASL Alphabet Dataset (Gốc)** Tập dữ liệu chính được sử dụng là ASL Alphabet Dataset, công khai trên nền tảng Kaggle bởi người dùng grassknoted [4]. Bộ dữ liệu bao gồm hình ảnh các ký hiệu tay trong bảng chữ cái tiếng Anh (A–Z), cùng với ba lớp bổ sung là "space", "delete", và "nothing". Mỗi lớp được lưu trong một thư mục riêng biệt, tổng cộng 29 lớp.

Tuy nhiên, nghiên cứu này chỉ sử dụng 28 lớp, loại bỏ lớp "nothing" vì không liên quan đến bài toán phân loại ký hiệu tay.

- Mỗi ảnh là ảnh màu RGB, kích thước cố định 200×200 pixel.
- Hình ảnh được chụp trong điều kiện tương đối lý tưởng: nền sáng, tay ở trung tâm, không có nhiễu.
- Dữ liệu được chia như sau:

| Tập dữ liệu | Số lượng ảnh |
|---|---|
| Train | 50 422 |
| Val | 14 441 |
| Test | 7 192 |

**ASL Augmentation Dataset** Để tăng khả năng tổng quát hóa của mô hình, nhóm đã tăng cường (augmentation) từ ASL Alphabet bằng các kỹ thuật phổ biến trong thị giác máy tính:

- Xoay ảnh một góc ngẫu nhiên trong phạm vi ±30 độ.
- Lật ảnh ngang để tạo đối xứng gương.
- Thay đổi độ sáng (brightness shift) để mô phỏng điều kiện ánh sáng đa dạng.
- Dữ liệu được chia như sau:

| Tập dữ liệu | Số lượng ảnh |
|---|---|
| Train | 119 378 |
| Val | 34 092 |
| Test | 17 051 |

**ASL Dataset Tự Sinh** Để kiểm tra mô hình trong điều kiện gần với thực tế, nhóm đã tự chụp ảnh bàn tay thật trong nhiều môi trường khác nhau:

- Phông nền phức tạp hơn, điều kiện ánh sáng đa dạng.
- Ảnh được gán nhãn thủ công, dựa vào ký hiệu ASL chuẩn.
- Dữ liệu được chia như sau:

| Tập dữ liệu | Số lượng ảnh |
|---|---|
| Train | 38 928 |
| Val | 11 131 |
| Test | 5 557 |

Việc kết hợp ba bộ dữ liệu này cho phép mô hình học được đặc trưng từ môi trường lý tưởng đến thực tế, đồng thời kiểm chứng độ ổn định và tính khái quát hóa của từng phiên bản MobileNet.

## 3.2 Mô hình và cài đặt

Do sử dụng các mô hình MobileNet đã được huấn luyện trên tập dữ liệu lớn (ImageNet) nên nhóm sẽ sử dụng phương pháp Transfer Learning để tiết kiệm thời gian huấn luyện cũng như tránh việc mô hình overfit vào tập dữ liệu ASL.

Môi trường thực nghiệm được thực hiện trên Google Colab 12.7GB RAM.

Cấu hình huấn luyện như sau:

- Batch size: 64
- Class weight: được tính riêng cho từng dataset.
- Optimizer: Adam.
- Epoch: 10.

Các pretrained MobileNet được bỏ từ sau lớp Pooling để làm feature extractor. Cấu hình chi tiết của từng mô hình như các hình 9, 10, 11:

| Input | Operator | $t$ | $c$ | $n$ | $s$ |
|---|---|---|---|---|---|
| $224^2 \times 3$ | conv2d | - | 32 | 1 | 2 |
| $112^2 \times 32$ | bottleneck | 1 | 16 | 1 | 1 |
| $112^2 \times 16$ | bottleneck | 6 | 24 | 2 | 2 |
| $56^2 \times 24$ | bottleneck | 6 | 32 | 3 | 2 |
| $28^2 \times 32$ | bottleneck | 6 | 64 | 4 | 2 |
| $14^2 \times 64$ | bottleneck | 6 | 96 | 3 | 1 |
| $14^2 \times 96$ | bottleneck | 6 | 160 | 3 | 2 |
| $7^2 \times 160$ | bottleneck | 6 | 320 | 1 | 1 |
| $7^2 \times 320$ | conv2d 1x1 | - | 1280 | 1 | 1 |
| $7^2 \times 1280$ | avgpool 7x7 | - | - | 1 | - |
| $1 \times 1 \times 1280$ | conv2d 1x1 | - | k | - | |

Hình 9: Detail MobileNetV2 [15]

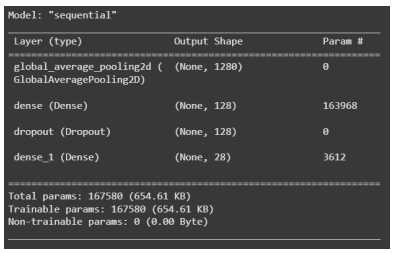
| Input | Operator | exp size | $\#out$ | SE | NL | $s$ |
|---|---|---|---|---|---|---|
| $224^2 \times 3$ | conv2d, 3x3 | - | 16 | - | HS | 2 |
| $112^2 \times 16$ | bneck, 3x3 | 16 | 16 | ✓ | RE | 2 |
| $56^2 \times 16$ | bneck, 3x3 | 72 | 24 | - | RE | 2 |
| $28^2 \times 24$ | bneck, 3x3 | 88 | 24 | - | RE | 1 |
| $28^2 \times 24$ | bneck, 5x5 | 96 | 40 | ✓ | HS | 2 |
| $14^2 \times 40$ | bneck, 5x5 | 240 | 40 | ✓ | HS | 1 |
| $14^2 \times 40$ | bneck, 5x5 | 240 | 40 | ✓ | HS | 1 |
| $14^2 \times 40$ | bneck, 5x5 | 120 | 48 | ✓ | HS | 1 |
| $14^2 \times 48$ | bneck, 5x5 | 144 | 48 | ✓ | HS | 1 |
| $14^2 \times 48$ | bneck, 5x5 | 288 | 96 | ✓ | HS | 2 |
| $7^2 \times 96$ | bneck, 5x5 | 576 | 96 | ✓ | HS | 1 |
| $7^2 \times 96$ | bneck, 5x5 | 576 | 96 | ✓ | HS | 1 |
| $7^2 \times 96$ | conv2d, 1x1 | - | 576 | ✓ | HS | 1 |
| $7^2 \times 576$ | pool, 7x7 | - | - | - | - | 1 |
| $1^2 \times 576$ | conv2d 1x1, NBN | - | 1024 | - | HS | 1 |
| $1^2 \times 1024$ | conv2d 1x1, NBN | - | k | - | - | 1 |

Hình 10: Detail MobileNetV3 [6]

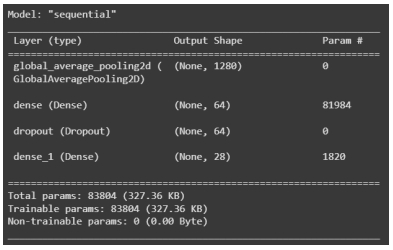
| Input | Block | DW $K_1$ | DW $K_2$ | Expanded Dim | Output Dim | Stride |
|---|---|---|---|---|---|---|
| $224^2 \times 3$ | Conv2D | - | $3 \times 3$ | - | 32 | 2 |
| $112^2 \times 32$ | FusedIB | - | $3 \times 3$ | 32 | 32 | 2 |
| $56^2 \times 32$ | FusedIB | - | $3 \times 3$ | 96 | 64 | 2 |
| $28^2 \times 64$ | ExtraDW | $5 \times 5$ | $5 \times 5$ | 192 | 96 | 2 |
| $14^2 \times 96$ | IB | - | $3 \times 3$ | 192 | 96 | 1 |
| $14^2 \times 96$ | IB | - | $3 \times 3$ | 192 | 96 | 1 |
| $14^2 \times 96$ | IB | - | $3 \times 3$ | 192 | 96 | 1 |
| $14^2 \times 96$ | ConvNext | $3 \times 3$ | - | 384 | 96 | 1 |
| $14^2 \times 96$ | ExtraDW | $3 \times 3$ | $3 \times 3$ | 576 | 128 | 2 |
| $7^2 \times 128$ | ExtraDW | $5 \times 5$ | $5 \times 5$ | 512 | 128 | 1 |
| $7^2 \times 128$ | IB | - | $5 \times 5$ | 512 | 128 | 1 |
| $7^2 \times 128$ | IB | - | $5 \times 5$ | 384 | 128 | 1 |
| $7^2 \times 128$ | IB | - | $3 \times 3$ | 512 | 128 | 1 |
| $7^2 \times 128$ | IB | - | $3 \times 3$ | 512 | 128 | 1 |
| $7^2 \times 128$ | Conv2D | - | $1 \times 1$ | - | 960 | 1 |
| $7^2 \times 960$ | AvgPool | - | $7 \times 7$ | - | 960 | 1 |
| $1^2 \times 960$ | Conv2D | - | $1 \times 1$ | - | 1280 | 1 |
| $1^2 \times 1280$ | Conv2D | - | $1 \times 1$ | - | 1000 | 1 |

Hình 11: Detail MobileNetV4 [14]

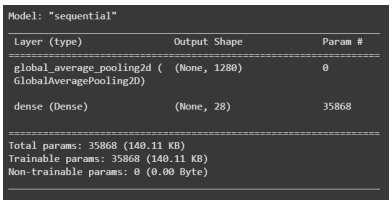
Các kiến trúc để phân loại được sử dụng để đánh giá bao gồm các kiến trúc như hình 12, 13, 14:




Hình 12: Classify Large




Hình 13: Classify Medium




Hình 14: Classify Small

### 3.3 Chỉ số đánh giá

Để đánh giá hiệu quả của các mô hình học sâu trong bài toán phân loại cử chỉ tay ASL, nghiên cứu này sử dụng ba chỉ số đánh giá chính: **Accuracy**, **F1-score**, và **tốc độ thực thi**. Mỗi chỉ số mang lại góc nhìn khác nhau về khả năng học và suy luận của mô hình.

**Accuracy** được định nghĩa là tỷ lệ giữa số lượng mẫu được phân loại đúng và tổng số mẫu trong tập kiểm tra. Đây là thước đo phổ biến nhất trong các bài toán phân loại.

$$\text{Accuracy} = \frac{\text{Số lượng dự đoán đúng}}{\text{Tổng số mẫu}}$$

Mặc dù đơn giản và dễ hiểu, accuracy có thể gây hiểu nhầm trong các tập dữ liệu mất cân bằng (tức là số lượng mẫu của các lớp chênh lệch nhau nhiều). Do đó, để có cái nhìn toàn diện hơn về từng lớp trong 28 lớp ASL, ta cần thêm các chỉ số vi mô.

**F1-score** là trung bình điều hòa giữa precision và recall – hai chỉ số rất quan trọng trong phân loại nhiều lớp.

- **Precision**: Tỷ lệ mẫu dự đoán đúng trong tất cả các mẫu mà mô hình dự đoán là thuộc lớp đó.
- **Recall**: Tỷ lệ mẫu được dự đoán đúng trong tất cả các mẫu thực sự thuộc lớp đó.

$$\text{F1-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Trong nghiên cứu này, F1-score được tính theo macro average, tức là tính trung bình F1 của từng lớp và sau đó lấy trung bình không trọng số. Điều này đảm bảo rằng mọi lớp đều được đánh giá công bằng, kể cả những lớp có ít dữ liệu hơn [13].

**Tốc độ thực thi** được đo bằng thời gian trung bình mà mô hình cần để xử lý một ảnh đầu vào và đưa ra dự đoán đầu ra. Đây là chỉ số đặc biệt quan trọng trong bối cảnh triển khai thực tế, nơi yêu cầu thời gian phản hồi gần như tức thì (thường $< 1$ giây/ảnh). Tốc độ được đo bằng đơn vị milliseconds/batch trong thực nghiệm.

## 4 Kết quả thực nghiệm

### 4.1 Kết quả train và validation

Các mô hình được huấn luyện trong 10 epoch trên cùng một cấu hình phần cứng (Google Colab, 12.7GB RAM), cùng các siêu tham số. Kết quả được ghi nhận trong bảng 1, 2, 3, 4, gồm:

- Train Loss: mất mát trên tập huấn luyện ở cuối epoch cuối cùng.
- Validation Loss: mất mát trên tập kiểm định.
- Train Accuracy: độ chính xác trên tập huấn luyện.
- Validation Accuracy: độ chính xác trên tập kiểm định.

Bảng 1: So sánh Train Loss giữa các mô hình MobileNet trên 3 tập dữ liệu

| Feature Extractor | Classifier | ASL Alphabet | ASL Alphabet Augmentation | ASL Alphabet tự sinh |
|---|---|---|---|---|
| MobileNetV2 | Small | 0.0030 | 0.0244 | 0.0033 |
| | Medium | 0.0313 | 0.1375 | 0.0283 |
| | Large | 0.0205 | 0.0804 | 0.0180 |
| MobileNetV3 | Small | 0.0254 | 0.0828 | 0.0336 |
| | Medium | 0.0507 | 0.1603 | 0.0514 |
| | Large | 0.0263 | 0.0949 | 0.0270 |
| MobileNetV4 | Small | 0.1035 | 0.1251 | 0.1163 |
| | Medium | 0.0638 | 0.1070 | 0.0553 |
| | Large | 0.0278 | 0.0636 | 0.0243 |

Bảng 2: So sánh Validation Loss giữa các mô hình MobileNet trên 3 tập dữ liệu

| Feature Extractor | Classifier | ASL Alphabet | ASL Alphabet Augmentation | ASL Alphabet tự sinh |
|---|---|---|---|---|
| MobileNetV2 | Small | 0.0159 | 0.1460 | 0.0090 |
| | Medium | 0.0243 | 0.2355 | 0.0134 |
| | Large | 0.0197 | 0.1456 | 0.0181 |
| MobileNetV3 | Small | 0.0421 | 0.2504 | 0.0453 |
| | Medium | 0.0324 | 0.2738 | 0.0244 |
| | Large | 0.0222 | 0.2506 | 0.0229 |
| MobileNetV4 | Small | 0.1245 | 0.3056 | 0.1152 |
| | Medium | 0.0387 | 0.1774 | 0.0282 |
| | Large | 0.0280 | 0.1474 | 0.0163 |

Bảng 3: So sánh Train Accuracy giữa các mô hình MobileNet trên 3 tập dữ liệu

| Feature Extractor | Classifier | ASL Alphabet | ASL Alphabet Augmentation | ASL Alphabet tự sinh |
|---|---|---|---|---|
| MobileNetV2 | Small | 0.9999 | 0.9924 | 0.9998 |
| | Medium | 0.9902 | 0.9547 | 0.9910 |
| | Large | 0.9932 | 0.9738 | 0.9935 |
| MobileNetV3 | Small | 0.9956 | 0.9746 | 0.9945 |
| | Medium | 0.9830 | 0.9460 | 0.9835 |
| | Large | 0.9919 | 0.9677 | 0.9915 |
| MobileNetV4 | Small | 0.9846 | 0.9697 | 0.9818 |
| | Medium | 0.9829 | 0.9651 | 0.9865 |
| | Large | 0.9931 | 0.9790 | 0.9942 |

Bảng 4: So sánh Validation Accuracy giữa các mô hình MobileNet trên 3 tập dữ liệu

| Feature Extractor | Classifier | ASL Alphabet | ASL Alphabet Augmentation | ASL Alphabet tự sinh |
|---|---|---|---|---|
| MobileNetV2 | Small | 0.9964 | 0.9576 | 0.9975 |
| | Medium | 0.9933 | 0.9268 | 0.9957 |
| | Large | 0.9951 | 0.9558 | 0.9955 |
| MobileNetV3 | Small | 0.9907 | 0.9206 | 0.9910 |
| | Medium | 0.9910 | 0.9130 | 0.9937 |
| | Large | 0.9939 | 0.9223 | 0.9935 |
| MobileNetV4 | Small | 0.9807 | 0.9120 | 0.9809 |
| | Medium | 0.9901 | 0.9453 | 0.9928 |
| | Large | 0.9933 | 0.9526 | 0.9962 |

## 4.2 Kết quả trên tập test

Kết quả dưới bảng 5, 6, 7 được đo trên tập test tách biệt hoàn toàn với dữ liệu train và validation, giúp đánh giá khả năng tổng quát hóa (generalization) của từng mô hình đối với các mẫu chưa từng thấy.

Bảng 5: So sánh Accuracy giữa các mô hình MobileNet trên 3 tập dữ liệu

| Feature Extractor | Classifier | ASL Alphabet | ASL Alphabet Augmentation | ASL Alphabet tự sinh |
|---|---|---|---|---|
| MobileNetV2 | Small | 0.9966 | 0.9569 | 0.9972 |
| | Medium | 0.9947 | 0.9268 | 0.9960 |
| | Large | 0.9956 | 0.9556 | 0.9955 |
| MobileNetV3 | Small | 0.9899 | 0.9240 | 0.9901 |
| | Medium | 0.9900 | 0.9192 | 0.9921 |
| | Large | 0.9927 | 0.9267 | 0.9946 |
| MobileNetV4 | Small | 0.9807 | 0.9110 | 0.9810 |
| | Medium | 0.9904 | 0.9453 | 0.9926 |
| | Large | 0.9926 | 0.9549 | 0.9967 |

Bảng 6: So sánh F1-Score giữa các mô hình MobileNet trên 3 tập dữ liệu

| Feature Extractor | Classifier | ASL Alphabet | ASL Alphabet Augmentation | ASL Alphabet tự sinh |
|---|---|---|---|---|
| MobileNetV2 | Small | 0.9963 | 0.9570 | 0.9972 |
| | Medium | 0.9946 | 0.9276 | 0.9960 |
| | Large | 0.9954 | 0.9554 | 0.9955 |
| MobileNetV3 | Small | 0.9895 | 0.9245 | 0.9902 |
| | Medium | 0.9896 | 0.9202 | 0.9921 |
| | Large | 0.9925 | 0.9267 | 0.9946 |
| MobileNetV4 | Small | 0.9802 | 0.9119 | 0.9813 |
| | Medium | 0.9902 | 0.9451 | 0.9927 |
| | Large | 0.9924 | 0.9547 | 0.9967 |

Bảng 7: So sánh Tốc độ thực thi giữa các mô hình MobileNet trên 3 tập dữ liệu

| Feature Extractor | Classifier | ASL Alphabet | ASL Alphabet Augmentation | ASL Alphabet tự sinh |
|---|---|---|---|---|
| MobileNetV2 | Small | 84 | 81 | 61 |
| | Medium | 70 | 58 | 61 |
| | Large | 198 | 147 | 173 |
| MobileNetV3 | Small | 61 | 51 | 54 |
| | Medium | 52 | 53 | 49 |
| | Large | 156 | 102 | 136 |
| MobileNetV4 | Small | 69 | 65 | 63 |
| | Medium | 68 | 63 | 65 |
| | Large | 203 | 118 | 164 |

# 5 Nhận xét

## 5.1 Ưu điểm và hạn chế của từng phiên bản MobileNet

MobileNetV2 và MobilenetV3-Small đã được triển khai trên Tensorflow với nên có thể có nhiều cách tối ưu hơn.

MobileNetV4-Conv-S mới ra mắt gần đây nên chỉ có pre-trained trên HuggingFace nên cách triển khai khác 2 phiên bản trước đó.

Các mô hình được sử dụng là các phiên bản có số lượng tham số ít nhất nên so sánh về thời gian sẽ có phần khách quan hơn so với so sánh về độ chính xác.

## 5.2 Ứng dụng thực tế

Để minh hoạ cách hệ thống phân loại ký hiệu ASL hoạt động trong thực tế, nhóm đã xây dựng một video trình diễn mô hình hoạt động trên ảnh thực tế. Video thể hiện quá trình dự đoán từng ký hiệu tay từ webcam và hiển thị nhãn dự đoán tương ứng.

**Link video demo:** https://www.youtube.com/watch?v=AU8EJxgciBo

# 6 Kết luận và hướng phát triển

## 6.1 Tóm tắt kết quả đạt được

Đã xây dựng được pipeline xử lý gồm bước phát hiện bàn tay (để lọc ảnh không phù hợp) và bước phân loại cử chỉ ASL với 28 lớp đầu ra.

Áp dụng và huấn luyện mô hình phân lớp với feature extractors là ba mô hình nhẹ: MobileNetV2, MobileNetV3-Small, và MobileNetV4-Conv-Small trên tập dữ liệu đã làm sạch.

Kết quả so sánh cho thấy:

- MobileNetV2 đạt độ chính xác cao nhất trên tập kiểm tra.
- MobileNetV3-Small có thời gian suy luận nhanh nhất, phù hợp hơn với thiết bị có tài nguyên thấp.
- MobileNetV4-Conv-Small có hiệu năng cân bằng nhưng thua kém về tốc độ và độ chính xác.

**Kết luận:** MobileNetV2 là lựa chọn phù hợp nếu ưu tiên độ chính xác; MobileNetV3-Small là lựa chọn ưu tiên nếu cần suy luận thời gian thực trên thiết bị hạn chế tài nguyên.

## 6.2 Hướng phát triển tương lai

Do kết quả đạt được trên ảnh khá cao ( $>$0.9 acc và f1) nên có thể hướng tới thực hiện trên video và stream để giống với thực tiễn hơn.

Có thể hướng tới thực hiện với dataset có nhiều động tác khó với mức độ ngữ nghĩa cao hơn (thay vì dataset về bảng chữ cái như hiện tại).

Thực nghiệm với nhiều phiên bản MobileNetV3 và MobileNetV4 hơn để có cái nhìn khách quan hơn so với sử dụng mô hình có ít tham số nhất của mỗi phiên bản.

# 7 References


[1] Diane Brentari. *A Prosodic Model of Sign Language Phonology*. MIT Press, 1998.

[2] Navneet Dalal **and** Bill Triggs. "Histograms of oriented gradients for human detection". **in***Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*: IEEE. 2005, **pages** 886–893.

[3] Alexey Dosovitskiy **andothers**. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". **in***International Conference on Learning Representations (ICLR)*: 2021. URL: https://arxiv.org/abs/2010.11929.

[4] Grassknoted. *ASL Alphabet*. https://www.kaggle.com/datasets/grassknoted/asl-alphabet. Accessed: 2025-06-13. 2017.

[5] Kaiming He **andothers**. "Deep Residual Learning for Image Recognition". **in***2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*: 2016, **pages** 770–778. DOI: 10.1109/CVPR.2016.90.

[6] Andrew Howard **andothers**. "Searching for MobileNetV3". **in***arXiv preprint arXiv:1905.02244*: (2019). URL: https://arxiv.org/abs/1905.02244.

[7] Andrew G. Howard **andothers**. "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications". **in***arXiv preprint arXiv:1704.04861*: 2017. URL: https://arxiv.org/abs/1704.04861.

[8] Jie Hu, Li Shen **and** Gang Sun. "Squeeze-and-Excitation Networks". **in***Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*: 2018, **pages** 7132–7141.

[9] Alex Krizhevsky, Ilya Sutskever **and** Geoffrey E Hinton. "ImageNet classification with deep convolutional neural networks". **in***Advances in Neural Information Processing Systems*: **volume** 25. 2012.

[10] Yann LeCun **andothers**. "Gradient-based learning applied to document recognition". **in***Proceedings of the IEEE*: 86.11 (1998), **pages** 2278–2324.

[11] David G Lowe. "Distinctive image features from scale-invariant keypoints". **in***International Journal of Computer Vision*: **volume** 60. 2. 2004, **pages** 91–110.

[12] National Association of the Deaf. *What is American Sign Language?* Truy cập ngày 10 tháng 6 năm 2025. 2022. URL: https://www.nad.org/resources/american-sign-language/what-is-american-sign-language/.

[13] David M. W. Powers. "Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation". **in***Journal of Machine Learning Technologies*: 2.1 (2011), **pages** 37–63.

[14] Danfeng Qin **andothers**. "MobileNetV4: Universal Models for the Mobile Ecosystem". **in***arXiv preprint arXiv:2404.10518*: (2024). v2, submitted Apr 16 2024, updated Sep 29 2024. URL: https://arxiv.org/abs/2404.10518.

[15] Mark Sandler **andothers**. "MobileNetV2: Inverted Residuals and Linear Bottlenecks". **in***Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*: 2018, **pages** 4510–4520. URL: https://openaccess.thecvf.com/content_cvpr_2018/html/Sandler_MobileNetV2_Inverted_Residuals_CVPR_2018_paper.html.

[16] Karen Simonyan **and** Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". **in***International Conference on Learning Representations*: 2015.

[17] William C Stokoe. *Sign Language Structure: An Outline of the Visual Communication Systems of the American Deaf*. Department of Anthropology **and** Linguistics, University of Buffalo, 1960.

[18] Mingxing Tan **and** Quoc V. Le. "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks". **in***Proceedings of the 36th International Conference on Machine Learning (ICML)*: **byeditor**Kamalika Chaudhuri **and** Ruslan Salakhutdinov. **volume** 97. Proceedings of Machine Learning Research. PMLR, **june** 2019, **pages** 6105–6114. URL: https://proceedings.mlr.press/v97/tan19a.html.

[19] Clayton Valli **andothers**. *Linguistics of American Sign Language: An Introduction*. Gallaudet University Press, 2000.

[20] Ashish Vaswani **andothers**. "Attention Is All You Need". **in***Advances in Neural Information Processing Systems (NeurIPS)*: **volume** 30. 2017. URL: https://papers.nips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.