**BỘ CÔNG THƯƠNG**

**TRƯỜNG ĐẠI HỌC CÔNG THƯƠNG TP. HỒ CHÍ MINH**

**KHOA CÔNG NGHỆ THÔNG TIN**

**----o0o---**



**ĐỒ ÁN TỐT NGHIỆP**

TÊN ĐỀ TÀI: **TÌM HIỂU THUẬT TOÁN OBJECT DECTION**

**VÀ XÂY DỰNG ỨNG DỤNG XÁC ĐỊNH GIỐNG CHÓ NUÔI**

***NHÓM***

Giảng viên hướng dẫn : Thầy TRẦN ĐÌNH TOÀN

Sinh viên thực hiện: 1. Lương Liêm Phong – Nhóm trưởng

2. Văn Trọng Dương

3. Trần Khánh Vũ

Thành phố Hồ Chí Minh, tháng 07 năm 2025

**BỘ CÔNG THƯƠNG**

**TRƯỜNG ĐẠI HỌC CÔNG THƯƠNG TP. HỒ CHÍ MINH**

**KHOA CÔNG NGHỆ THÔNG TIN**

**----o0o---**



**TÊN ĐỀ TÀI:** TÌM HIỂU THUẬT TOÁN OBJECT DECTION

VÀ XÂY DỰNG ỨNG DỤNG XÁC ĐỊNH GIỐNG CHÓ NUÔI

|  |  |
| --- | --- |
| **Sinh viên thực hiện** | **Giáo viên hướng dẫn:** |
| 1. Văn Trọng Dương - 2001220817 2. Lương Liêm Phong - 2001223664 3. Trần Khánh Vũ - 2001225914 | *Ths* Trần Đình Toàn |

Thành phố Hồ Chí Minh, tháng 07 năm 2025

# **MỤC LỤC**

[MỤC LỤC 3](#_Toc202529504)

[CHƯƠNG I. Cơ Sở Lý Thuyết và Phương Pháp Nghiên Cứu 5](#_Toc202529505)

[1. Giới thiệu chung về các kiến trúc được lựa chọn 5](#_Toc202529506)

[2. Cơ sở lý thuyết về mạng CNN và các mô hình được lựa chọn 5](#_Toc202529507)

[2.1. Mạng nơ-ron tích chập (Convolutional Neural Networks - CNN) 5](#_Toc202529508)

[2.2. EfficientNetV2 6](#_Toc202529509)

[2.3. MobileNetV3 6](#_Toc202529510)

[2.4. Ứng dụng trong Object Detection 7](#_Toc202529511)

[3. Bảng tóm tắt so sánh 7](#_Toc202529512)

[4. Triển khai và Tinh chỉnh trong nghiên cứu 8](#_Toc202529513)

[4.1. Thu thập dữ liệu 8](#_Toc202529514)

[4.2. Triển khai mô hình 8](#_Toc202529515)

[4.3. Đánh giá hiệu suất 8](#_Toc202529516)

[4.4. Xây dựng ứng dụng web 9](#_Toc202529517)

[5. Danh mục Tài liệu tham khảo 9](#_Toc202529518)

LỜI CAM ĐOAN

Chúng tôi xin cam đoan đề tài đồ án: **“ Tìm hiểu thuật toán object dection và xây dựng ứng dụng xác định giống chó nuôi ”** do nhóm nghiên cứu và thực hiện. Chúng tôi đã kiểm tra dữ liệu theo quy định hiện hành.

Kết quả bài làm của đề tài: **Tìm hiểu thuật toán object dection và xây dựng ứng dụng xác định giống chó nuôi** là trung thực và không sao chép từ bất kỳ bài tập của nhóm khác.

Các tài liệu được sử dụng trong tiểu luận có nguồn gốc, xuất xứ rõ ràng.

**(Ký và ghi rõ họ tên)**

Phong

Lương Liêm Phong

# **CHƯƠNG I. Cơ Sở Lý Thuyết và Phương Pháp Nghiên Cứu**

## **1. Giới thiệu chung về các kiến trúc được lựa chọn**

Trong bài toán xác định giống chó nuôi, nghiên cứu sử dụng các mạng nơ-ron tích chập (Convolutional Neural Networks - CNN) làm nền tảng cho các thuật toán phát hiện và phân loại đối tượng (object detection). Hai mô hình pretrained được lựa chọn là **EfficientNetV2** và **MobileNetV3**, được tích hợp vào hai thuật toán object detection: **Faster R-CNN** và **YOLO**.

* **EfficientNetV2 kết hợp với Faster R-CNN**: Được chọn để tối ưu hóa độ chính xác trên tập kiểm tra (validation accuracy). EfficientNetV2 cung cấp khả năng trích xuất đặc trưng chi tiết, kết hợp với cơ chế Region Proposal Network (RPN) của Faster R-CNN, đảm bảo độ chính xác cao trong việc định vị và phân loại giống chó, phù hợp với các hệ thống có tài nguyên tính toán mạnh.
* **MobileNetV3 kết hợp với YOLO**: Được chọn để tối ưu hóa tốc độ suy luận, lý tưởng cho các ứng dụng yêu cầu xử lý nhanh trên thiết bị tài nguyên hạn chế như điện thoại di động hoặc hệ thống nhúng. MobileNetV3, với thiết kế nhẹ và hiệu quả, kết hợp với kiến trúc single-stage của YOLO, đảm bảo tốc độ suy luận nhanh chóng.

## **2. Cơ sở lý thuyết về mạng CNN và các mô hình được lựa chọn**

### **2.1. Mạng nơ-ron tích chập (Convolutional Neural Networks - CNN)**

**Khái niệm và Nguồn gốc**: Mạng nơ-ron tích chập (CNN) là một lớp mạng nơ-ron sâu được thiết kế chuyên biệt cho các bài toán xử lý hình ảnh, được Yann LeCun và cộng sự giới thiệu vào năm 1989 [6]. CNN sử dụng các phép tích chập (convolution) để trích xuất đặc trưng từ hình ảnh, giảm số lượng tham số so với mạng fully connected, đồng thời bảo toàn thông tin không gian.

**Cơ chế hoạt động**:

* **Lớp tích chập (Convolutional Layer):**
  + **Công thức:** Cho hình ảnh đầu vào ( I ) kích thước ( H \times W \times C ) (chiều cao, chiều rộng, số kênh), và bộ lọc ( K ) kích thước ( k \times k \times C ), phép tích chập tạo ra feature map ( O ) theo công thức: [ O(i,j) = \sum\_{m=0}^{k-1} \sum\_{n=0}^{k-1} \sum\_{c=0}^{C-1} I(i+m, j+n, c) \cdot K(m, n, c) + b ] trong đó ( b ) là bias, ( (i,j) ) là tọa độ trên feature map.
  + **Chức năng:** Bộ lọc trích xuất các đặc trưng như cạnh, góc, hoặc mẫu phức tạp (ví dụ: mắt, mũi của chó). Nhiều bộ lọc tạo ra nhiều feature maps, mỗi feature map tập trung vào một loại đặc trưng.
* **Hàm kích hoạt:** Hàm ReLU (( f(x) = \max(0, x) )) được áp dụng sau mỗi lớp tích chập để thêm tính phi tuyến, giúp mô hình học các đặc trưng phức tạp và tăng tốc độ hội tụ.
* **Lớp gộp (Pooling Layer):**
  + **Công thức (MaxPooling):** Cho một vùng ( k \times k ) trên feature map, MaxPooling lấy giá trị lớn nhất: [ O(i,j) = \max\_{m=0}^{k-1} \max\_{n=0}^{k-1} I(i \cdot s + m, j \cdot s + n) ] trong đó ( s ) là stride (bước trượt).
  + **Chức năng:** Giảm kích thước không gian của feature maps, giảm chi phí tính toán, tăng tính bất biến với dịch chuyển, và ngăn ngừa quá khớp.
* **Lớp fully connected:**
  + **Công thức:** Biến feature maps thành vector, sau đó áp dụng phép nhân ma trận: [ y = W \cdot x + b ] trong đó ( W ) là ma trận trọng số, ( x ) là vector đặc trưng, ( b ) là bias.
  + **Hàm Softmax**: Cho bài toán phân loại đa lớp (ví dụ: 120 giống chó), Softmax chuyển đầu ra thành xác suất: [ P(y\_i) = \frac{e^{z\_i}}{\sum\_{j=1}^N e^{z\_j}} ]
* **Chính quy hóa:**
  + **Dropout**: Bỏ ngẫu nhiên một tỷ lệ nơ-ron (ví dụ: 40%) trong quá trình huấn luyện để ngăn quá khớp.
  + **Batch Normalization**: Chuẩn hóa đầu ra của mỗi lớp để ổn định quá trình huấn luyện: [ \hat{x} = \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} \cdot \gamma + \beta ] trong đó ( \mu, \sigma^2 ) là trung bình và phương sai, ( \gamma, \beta ) là tham số học được.

**Ứng dụng trong bài toán**: CNN đóng vai trò backbone trong Faster R-CNN và YOLO, trích xuất các đặc trưng cấp thấp (cạnh, góc) và cấp cao (hình dạng, kết cấu của giống chó), cung cấp thông tin cần thiết cho việc định vị và phân loại đối tượng.

**2.2. Faster R-CNN**

**Khái niệm và Nguồn gốc**: Faster R-CNN, được đề xuất bởi Shaoqing Ren và cộng sự vào năm 2015 [8], là một thuật toán object detection thuộc nhóm two-stage, cải tiến từ R-CNN và Fast R-CNN. Thuật toán này tích hợp một **Region Proposal Network (RPN)** để tự động tạo các vùng quan tâm (regions of interest - RoI), kết hợp với backbone CNN để trích xuất đặc trưng, phân loại, và tinh chỉnh bounding box.

**Cơ chế hoạt động**:

* **Backbone CNN**:
  + Sử dụng một mạng CNN như EfficientNetV2 để trích xuất feature maps từ hình ảnh đầu vào ( I \in \mathbb{R}^{H \times W \times 3} ).
  + **Công thức**: Feature maps ( F \in \mathbb{R}^{H' \times W' \times C} ) được tạo ra sau các lớp tích chập và gộp, trong đó ( H', W' ) nhỏ hơn ( H, W ), và ( C ) là số kênh (ví dụ: 1280 trong EfficientNetV2-S).
  + **Chức năng**: Feature maps chứa thông tin đặc trưng cấp cao, như hình dạng hoặc kết cấu của chó, cần thiết cho việc phát hiện và phân loại.
* **Region Proposal Network (RPN)**:
  + **Cơ chế**: RPN là một mạng nơ-ron tích chập nhỏ, áp dụng một cửa sổ trượt (sliding window) trên feature maps để tạo các vùng đề xuất (proposals).
  + **Anchor Boxes**: Tại mỗi vị trí trên feature map, RPN sử dụng ( k ) anchor boxes (hộp neo) với các tỷ lệ (aspect ratios) và kích thước khác nhau (ví dụ: ( k=9 ), với 3 tỷ lệ và 3 kích thước).
  + **Dự đoán**:
    - **Objectness Score**: Dự đoán xác suất vùng chứa đối tượng (so với nền): [ p = \text{sigmoid}(W\_o \cdot F + b\_o) ] trong đó ( W\_o, b\_o ) là trọng số và bias.
    - **Bounding Box Regression**: Tinh chỉnh tọa độ anchor box: [ t\_x = (x - x\_a)/w\_a, \quad t\_y = (y - y\_a)/h\_a ] [ t\_w = \log(w/w\_a), \quad t\_h = \log(h/h\_a) ] trong đó ( (x, y, w, h) ) là tọa độ và kích thước dự đoán, ( (x\_a, y\_a, w\_a, h\_a) ) là tọa độ và kích thước của anchor box.
  + **Hàm mất mát RPN**: [ L\_{\text{RPN}} = \frac{1}{N\_{\text{cls}}} \sum\_i L\_{\text{cls}}(p\_i, p\_i^*) + \lambda \frac{1}{N\_{\text{reg}}} \sum\_i p\_i^* L\_{\text{reg}}(t\_i, t\_i^*) ] trong đó ( L\_{\text{cls}} ) là mất mát phân loại (Cross-Entropy), ( L\_{\text{reg}} ) là mất mát hồi quy (Smooth L1 Loss), ( p\_i^* ) là nhãn thực (1 nếu có đối tượng, 0 nếu là nền), ( t\_i^\* ) là tọa độ thực.
* **RoI Align**:
  + **Cơ chế**: Chuẩn hóa kích thước của các vùng đề xuất thành kích thước cố định (ví dụ: 7x7) bằng nội suy song tuyến (bilinear interpolation), tránh lỗi làm tròn như trong RoI Pooling.
  + **Công thức**: Cho vùng đề xuất ( R ), RoI Align chia vùng thành lưới ( 7 \times 7 ), tính giá trị tại mỗi ô bằng nội suy từ feature map.
* **Phân loại và Tinh chỉnh**:
  + **Phân loại**: Các vùng đề xuất được đưa qua lớp fully connected để dự đoán lớp (ví dụ: giống chó cụ thể) bằng Softmax.
  + **Tinh chỉnh Bounding Box**: Tinh chỉnh tọa độ bounding box bằng hồi quy: [ t\_x = W\_x \cdot F\_{\text{RoI}} + b\_x ] tương tự cho ( t\_y, t\_w, t\_h ).
  + **Hàm mất mát đầu ra**: [ L = L\_{\text{cls}}(p, p^*) + \lambda L\_{\text{reg}}(t, t^*) ]
* **Tổng hàm mất mát**: [ L\_{\text{total}} = L\_{\text{RPN}} + L ]

**Giả định**:

* Mối quan hệ tuyến tính trong hồi quy bounding box.
* Sai số trong dự đoán bounding box có phân phối chuẩn.
* Các vùng đề xuất độc lập với nhau.

**Ứng dụng trong bài toán**: Faster R-CNN, kết hợp với **EfficientNetV2**, tối ưu hóa độ chính xác trên tập kiểm tra nhờ khả năng trích xuất đặc trưng chi tiết và cơ chế RPN mạnh mẽ. Trong bài toán xác định giống chó nuôi, Faster R-CNN định vị chính xác vị trí của chó và phân loại đúng giống chó (trong 120 giống), đặc biệt phù hợp với các hệ thống có tài nguyên mạnh.

**2.3. YOLO (You Only Look Once)**

**Khái niệm và Nguồn gốc**: YOLO, được Joseph Redmon và cộng sự giới thiệu vào năm 2016 [9], là một thuật toán object detection thuộc nhóm single-stage, thực hiện phát hiện và phân loại đối tượng trong một lần forward pass duy nhất. Các phiên bản cải tiến như YOLOv5 và YOLOv8 nâng cao hiệu suất và tốc độ.

**Cơ chế hoạt động**:

* **Backbone CNN**:
  + Sử dụng một mạng CNN như MobileNetV3 để trích xuất feature maps từ hình ảnh đầu vào ( I \in \mathbb{R}^{H \times W \times 3} ).
  + **Công thức**: Feature maps ( F \in \mathbb{R}^{H' \times W' \times C} ) được tạo ra sau các lớp tích chập và gộp.
* **Lưới (Grid) và Anchor Boxes**:
  + Hình ảnh được chia thành lưới ( S \times S ) (ví dụ: 13x13).
  + Mỗi ô lưới dự đoán ( B ) anchor boxes, mỗi anchor box dự đoán:
    - **Tọa độ Bounding Box**: ( (x, y, w, h) ), tinh chỉnh tương tự như trong RPN của Faster R-CNN.
    - **Objectness Score**: Xác suất vùng chứa đối tượng: [ p\_o = \text{sigmoid}(W\_o \cdot F + b\_o) ]
    - **Xác suất lớp**: Xác suất đối tượng thuộc vào mỗi lớp (ví dụ: 120 giống chó): [ p\_c = \text{softmax}(W\_c \cdot F + b\_c) ]
* **Dự đoán đồng thời**:
  + **Công thức dự đoán**: [ x = \text{sigmoid}(t\_x) + c\_x, \quad y = \text{sigmoid}(t\_y) + c\_y ] [ w = p\_w \cdot e^{t\_w}, \quad h = p\_h \cdot e^{t\_h} ] trong đó ( (c\_x, c\_y) ) là tọa độ ô lưới, ( (p\_w, p\_h) ) là kích thước anchor box, ( (t\_x, t\_y, t\_w, t\_h) ) là giá trị dự đoán.
* **Hàm mất mát**:
  + **Mất mát Bounding Box**: Sử dụng Smooth L1 Loss hoặc IoU Loss: [ L\_{\text{box}} = \sum\_i \text{SmoothL1}(t\_i, t\_i^\*) ]
  + **Mất mát Objectness**: Binary Cross-Entropy: [ L\_{\text{obj}} = -\sum\_i [p\_i^\* \log(p\_i) + (1-p\_i^\*) \log(1-p\_i)] ]
  + **Mất mát phân loại**: Cross-Entropy: [ L\_{\text{cls}} = -\sum\_i \sum\_c p\_{i,c}^\* \log(p\_{i,c}) ]
  + **Tổng hàm mất mát**: [ L = \lambda\_{\text{box}} L\_{\text{box}} + \lambda\_{\text{obj}} L\_{\text{obj}} + \lambda\_{\text{cls}} L\_{\text{cls}} ]
* **Non-Maximum Suppression (NMS)**:
  + Loại bỏ các bounding box trùng lặp, giữ lại box có điểm số objectness cao nhất: [ \text{IoU}(A, B) = \frac{A \cap B}{A \cup B} ] Nếu IoU vượt ngưỡng (ví dụ: 0.5), loại bỏ box có điểm số thấp hơn.

**Giả định**:

* Mối quan hệ tuyến tính trong hồi quy bounding box.
* Sai số trong dự đoán bounding box có phân phối chuẩn.
* Các ô lưới độc lập với nhau.

**Ứng dụng trong bài toán**: YOLO, kết hợp với **MobileNetV3**, tối ưu hóa tốc độ suy luận nhờ kiến trúc single-stage và backbone nhẹ. Trong bài toán xác định giống chó nuôi, YOLO giúp phát hiện và phân loại giống chó nhanh chóng, phù hợp cho triển khai trên thiết bị di động hoặc nhúng.

**2.4. EfficientNetV2**

**Khái niệm và Nguồn gốc**: EfficientNetV2, được Mingxing Tan và Quoc V. Le đề xuất vào năm 2021 [1], là phiên bản cải tiến của EfficientNet [2]. Mô hình này tối ưu hóa hiệu suất và tốc độ huấn luyện, đạt độ chính xác cao với số tham số hợp lý, phù hợp làm backbone cho Faster R-CNN.

**Cơ chế hoạt động**:

* **Compound Scaling**:
  + **Cơ chế**: Co giãn đồng bộ độ sâu (số lớp), độ rộng (số kênh), và độ phân giải ảnh đầu vào theo một hệ số tỷ lệ ( \phi ): [ \text{depth} = \alpha^\phi, \quad \text{width} = \beta^\phi, \quad \text{resolution} = \gamma^\phi ] trong đó ( \alpha, \beta, \gamma ) là các hằng số được tối ưu hóa bằng tìm kiếm kiến trúc.
  + **Chức năng**: Đảm bảo cân bằng giữa độ chính xác và chi phí tính toán.
* **Fused-MBConv**:
  + **Cơ chế**: Thay thế khối MBConv (Mobile Inverted Bottleneck Convolution) bằng Fused-MBConv ở các tầng đầu. MBConv gồm tích chập sâu 3x3, tích chập điểm 1x1, và Squeeze-and-Excitation (SE). Fused-MBConv tích hợp tích chập 3x3 và 1x1 thành một lớp duy nhất: [ y = \text{Conv3x3}(x) + b ]
  + **Chức năng**: Giảm độ trễ trên GPU/TPU, tăng tốc độ suy luận.
* **Progressive Training**:
  + **Cơ chế**: Tăng dần kích thước ảnh đầu vào (từ 224x224 đến 384x384) và mức độ chính quy hóa (Dropout, Data Augmentation) trong quá trình huấn luyện.
  + **Chức năng**: Giúp mô hình hội tụ nhanh và đạt độ chính xác cao.
* **Hàm kích hoạt Swish**:
  + **Công thức**: ( f(x) = x \cdot \text{sigmoid}(x) ).
  + **Chức năng**: Cải thiện độ chính xác so với ReLU.
* **Hiệu quả tham số**: EfficientNetV2-S, với khoảng 20 triệu tham số, đạt độ chính xác Top-1 là 83.9% trên ImageNet-1K [1].

**Ứng dụng trong bài toán**: EfficientNetV2-S làm backbone cho **Faster R-CNN**, cung cấp các đặc trưng chi tiết (hình dạng, màu sắc, kết cấu của giống chó), tối ưu hóa độ chính xác trên tập kiểm tra. Phù hợp với các hệ thống có tài nguyên mạnh.

**2.5. MobileNetV3**

**Khái niệm và Nguồn gốc**: MobileNetV3, được Andrew Howard và cộng sự giới thiệu vào năm 2019 [3], là phiên bản cải tiến của MobileNetV1 [4] và MobileNetV2 [5]. Mô hình này được thiết kế cho các thiết bị tài nguyên hạn chế, tập trung vào giảm độ trễ và chi phí tính toán, lý tưởng làm backbone cho YOLO.

**Cơ chế hoạt động**:

* **Neural Architecture Search (NAS)**:
  + **Cơ chế**: Sử dụng thuật toán tìm kiếm kiến trúc để tối ưu hóa cấu trúc mạng, chọn số lớp, số kênh, và kích thước bộ lọc tối ưu.
  + **Chức năng**: Đảm bảo hiệu quả tối đa về tốc độ và độ chính xác.
* **Depth-wise Separable Convolutions**:
  + **Cơ chế**: Tách tích chập thông thường thành:
    - **Depth-wise Convolution**: Áp dụng một bộ lọc 3x3 cho mỗi kênh: [ y\_c = \sum\_{m=0}^{2} \sum\_{n=0}^{2} x\_c(i+m, j+n) \cdot K(m, n) ]
    - **Point-wise Convolution**: Áp dụng tích chập 1x1 để kết hợp các kênh: [ y = \sum\_{c=0}^{C-1} x\_c \cdot W\_c + b ]
  + **Chức năng**: Giảm số lượng tham số và chi phí tính toán.
* **Squeeze-and-Excitation (SE)**:
  + **Cơ chế**: Tính trọng số cho mỗi kênh dựa trên mức độ quan trọng: [ s = \text{sigmoid}(\text{FC2}(\text{ReLU}(\text{FC1}(\text{GlobalAvgPool}(x))))) ] [ y = x \cdot s ]
  + **Chức năng**: Tăng cường khả năng chú ý vào các kênh đặc trưng quan trọng.
* **Hàm h-swish**:
  + **Công thức**: ( f(x) = x \cdot \frac{\text{ReLU6}(x+3)}{6} ).
  + **Chức năng**: Giảm độ trễ so với Swish, phù hợp cho thiết bị di động.
* **Hiệu quả tham số**: MobileNetV3-Large, với khoảng 5.4 triệu tham số, đạt độ chính xác Top-1 là 75.2% trên ImageNet-1K [3].

**Ứng dụng trong bài toán**: MobileNetV3-Large làm backbone cho **YOLO**, tối ưu hóa tốc độ suy luận nhờ thiết kế nhẹ. Trong bài toán xác định giống chó nuôi, MobileNetV3-Large giúp trích xuất đặc trưng nhanh chóng, phù hợp cho triển khai trên thiết bị di động hoặc nhúng.

**2.6. Ứng dụng trong Object Detection**

Trong bài toán xác định giống chó nuôi:

* **EfficientNetV2-S với Faster R-CNN**: Tối ưu hóa độ chính xác trên tập kiểm tra nhờ khả năng trích xuất đặc trưng chi tiết và cơ chế RPN mạnh mẽ. Phù hợp với các hệ thống có tài nguyên mạnh, đảm bảo phân loại chính xác 120 giống chó.
* **MobileNetV3-Large với YOLO**: Tối ưu hóa tốc độ suy luận nhờ kiến trúc single-stage và backbone nhẹ. Phù hợp cho triển khai trên thiết bị di động hoặc nhúng, đảm bảo xử lý nhanh trong các ứng dụng thực tế.

Cả hai mô hình được tiền huấn luyện trên ImageNet, cung cấp các đặc trưng cấp cao như hình dạng, kết cấu, và màu sắc, phù hợp cho việc nhận diện giống chó.

## **3. Bảng tóm tắt so sánh**

**Bảng X.X: So sánh các đặc điểm chính của EfficientNetV2-S và MobileNetV3-Large**

|  |  |  |
| --- | --- | --- |
| **Đặc điểm** | **EfficientNetV2-S** | **MobileNetV3-Large** |
| **Tác giả/Năm** | Tan & Le (2021) [1] | Howard et al. (2019) [3] |
| **Mục tiêu chính** | Cân bằng độ chính xác và tốc độ huấn luyện | Giảm độ trễ, tối ưu cho thiết bị di động |
| **Đổi mới chính** | Fused-MBConv, Progressive Training, Compound Scaling | NAS, Squeeze-and-Excitation, h-swish |
| **Số tham số (khoảng)** | ~20 triệu | ~5.4 triệu |
| **Top-1 Acc. (ImageNet)** | 83.9% | 75.2% |
| **Lý do lựa chọn** | Độ chính xác cao, phù hợp làm backbone cho Faster R-CNN | Nhẹ, nhanh, phù hợp cho YOLO và thiết bị di động |
| **Ứng dụng trong bài toán** | Trích xuất đặc trưng chi tiết, phù hợp với phần cứng mạnh | Trích xuất đặc trưng nhanh, triển khai trên thiết bị hạn chế |

## **4. Triển khai và Tinh chỉnh trong nghiên cứu**

### **4.1. Thu thập dữ liệu**

Bộ dữ liệu được sử dụng là **Stanford Dogs Dataset** [7], chứa khoảng **20,580 hình ảnh** thuộc **120 giống chó** khác nhau, với nhãn phân loại và bounding box để hỗ trợ object detection. Dữ liệu được chia thành tập huấn luyện (70%), tập kiểm tra (20%), và tập xác thực (10%). Các hình ảnh được tiền xử lý (resize về 224x224 hoặc 300x300 tùy mô hình, chuẩn hóa giá trị pixel về [0, 1]).

### **4.2. Triển khai mô hình**

Các mô hình được triển khai bằng **Python 3.8**, sử dụng **TensorFlow 2.10.0** và **Keras**. Quy trình triển khai và tinh chỉnh bao gồm:

1. **Tải mô hình pretrained**:
   * Tải **EfficientNetV2-S** và **MobileNetV3-Large** từ tf.keras.applications, loại bỏ lớp phân loại trên cùng (include\_top=False).
   * Sử dụng trọng số tiền huấn luyện trên **ImageNet** để khởi tạo.
2. **Tích hợp vào thuật toán object detection**:
   * **Faster R-CNN**: Sử dụng EfficientNetV2-S làm backbone trong framework như Detectron2. Region Proposal Network (RPN) được tinh chỉnh để phát hiện bounding box của chó.
   * **YOLO**: Sử dụng MobileNetV3-Large làm backbone trong YOLOv5 hoặc YOLOv8, tối ưu cho tốc độ suy luận.
3. **Tinh chỉnh (Fine-tuning)**:
   * **Bước 1**: Đóng băng các lớp tích chập của backbone, chỉ huấn luyện phần đầu phân loại (hoặc head của Faster R-CNN/YOLO) trong **10 epochs**.
   * **Bước 2**: Mở đóng băng một số lớp cuối của backbone, tiếp tục huấn luyện với **learning rate 1e-5** trong **40 epochs**.
   * Sử dụng optimizer **Adam** và hàm mất mát **Categorical Crossentropy** cho phân loại, kết hợp với **IoU loss** cho bounding box trong object detection.
   * Áp dụng các kỹ thuật chính quy hóa như **Dropout (0.4)** và **Data Augmentation** (xoay, lật, thay đổi độ sáng) để chống quá khớp.
4. **Phần cứng**: Huấn luyện trên GPU NVIDIA RTX 3060, với batch size là 16 cho EfficientNetV2-S và 32 cho MobileNetV3-Large.

### **4.3. Đánh giá hiệu suất**

Hiệu suất được đánh giá bằng các chỉ số:

* **mAP (mean Average Precision)**: Đo lường độ chính xác của object detection.
* **FPS (Frames Per Second)**: Đánh giá tốc độ suy luận.
* **Tổng tham số và FLOPs**: So sánh chi phí tính toán.  
  Kết quả sẽ được so sánh giữa EfficientNetV2-S (Faster R-CNN) và MobileNetV3-Large (YOLO) để xác định mô hình tối ưu cho bài toán.

### **4.4. Xây dựng ứng dụng web**

Ứng dụng web được phát triển bằng **Flask** hoặc **Django** (Python) để triển khai mô hình tốt nhất. Người dùng có thể tải lên hình ảnh chó, và hệ thống sẽ trả về giống chó được dự đoán cùng với bounding box. Giao diện sử dụng HTML/CSS/JavaScript, với backend tích hợp mô hình pretrained thông qua **TensorFlow Serving** hoặc **ONNX Runtime**.

## **5. Danh mục Tài liệu tham khảo**

[1] Tan, M., & Le, Q. V. (2021). *EfficientNetV2: Smaller Models and Faster Training*. Proceedings of the 38th International Conference on Machine Learning (ICML).  
[2] Tan, M., & Le, Q. V. (2019). *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*. Proceedings of the 36th International Conference on Machine Learning (ICML).  
[3] Howard, A., Sandler, M., Chu, G., et al. (2019). *Searching for MobileNetV3*. Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV).  
[4] Howard, A. G., Zhu, M., Chen, B., et al. (2017). *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. arXiv preprint arXiv:1704.04861.  
[5] Sandler, M., Howard, A., Zhu, M., et al. (2018). *MobileNetV2: Inverted Residuals and Linear Bottlenecks*. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).  
[6] LeCun, Y., Boser, B., Denker, J. S., et al. (1989). *Backpropagation Applied to Handwritten Zip Code Recognition*. Neural Computation, 1(4), 541-551.  
[7] Khosla, A., Jayadevaprakash, N., Yao, B., & Fei-Fei, L. (2011). *Novel Dataset for Fine-Grained Image Categorization*. Stanford Dogs Dataset.