**Tập Train**

  Đây là tập dữ liệu mà mô hình được học và điều chỉnh các tham số (weights và biases), nhiều để mô hình có đầy đủ thông tin để học.

**Tập Validation**

Điều chỉnh hoặc Đánh giá mô hình trong quá trình huấn luyện nhưng không dùng để cập nhật tham số. DỮ LIỆU KHÔNG TRÙNG LẶP VỚI TRAIN. ⇒ XEM MÔ HÌNH CÓ BỊ OVER HAY UNDER

**Tập Test**

Đánh giá cuối cùng mô hình sau khi hoàn thành quá trình huấn luyện và điều chỉnh trên tập validation. DỮ LIỆU PHẢI KHÁC BIỆT SO VỚI TẬP TRAIN VÀ TEST

**Prefetching**

Prefetching để tăng tốc độ load dữ liệu bằng cách tải trước các batch dữ liệu cho train\_dataset, validation\_dataset, và test\_dataset, giúp tối ưu hóa hiệu suất huấn luyện mô hình trên CPU và GPU.

A screen shot of a computer

Description automatically generated

⇒ Dùng thay thế nếu không có hàm propreocess\_input đưa giá trị về [-1, 1]

A screen shot of a computer

Description automatically generated

⇒ include\_top = False,  Không bao gồm các lớp phân loại ở phần "top" của mô hình, giúp chỉ giữ lại các lớp trích xuất đặc trưng.

⇒ Khởi tạo MobileNet V2 đã huấn luyện trước trên ImageNet, loại bỏ các lớp phân loại, để dùng phần trích xuất đặc trưng (bottleneck layer).

A computer code with text

Description automatically generated with medium confidence

⇒ Lấy một batch ảnh từ tập huấn luyện, truyền qua bộ trích xuất đặc trưng, từ  Từ kích thước **160x160x3**, mỗi ảnh được nén lại thành một khối đặc trưng **5x5x1280**

⇒ Giữ được thông tin cần thiết và quan trọng

A screenshot of a computer

Description automatically generated

⇒ Đóng băng (freeze) các lớp của mô hình cơ s**ở** MobileNet V2 để sử dụng nó như một bộ trích xuất đặc trưng cố định.

⇒ Dòng code tức là không có lớp nào trong base model được cập nhật trọng số trong quá trình huấn luyện. CÁC TRỌNG SỐ CỦA MOBILENET V2 KHÔNG THAY ĐỔI, MÔ HÌNH GIỮ NGUYÊN KIẾN THỨC TỪ IMAGENET => CHỈ TẬP VÀO HỌC PHẦN BỘ PHÂN LOẠI Ở TRÊN CÙNG.

**Important note about BatchNormalization layers**

Sử dụng để chuẩn hóa các đầu vào của một lớp trong mạng nơ-ron

⇒ Tăng tốc độ huấn luyện và ổn định quá trình học.

Một mô hình chứa các lớp Batch Normalization được đóng băng (layer.trainable = False), chúng sẽ hoạt động ở chế độ suy diễn (inference mode). ⇒ Không cập nhật các thống kê (mean và variance) của đầu vào, mà sử dụng các giá trị đã học được từ quá trình huấn luyện trước.

Khi mở khóa mô hình để tinh chỉnh (unfreeze), nếu chưa các lớp Batch Normalization,

cần phải giữ các lớp này ở (inference mode), thực hiện bằng cách truyền tham số training=False.

**Không thì việc cập  nhật trọng số không thể trainable (các lớp đã đóng băng) sẽ làm hỏng những gì mô hình học được, vì lớp đó cố gắng điều chỉnh thống kê mà không có dữ liệu huấn luyện đúng.**

**Add a classification head**

A screenshot of a computer

Description automatically generated

* input\_layer : là lớp đầu vào của mô hình
* sequential : lớp chứa tăng cường dữ liệu
* true\_divide : lớp trong quá trình tiền xử lý pixel,  chia giá trị pixel cho 255
* subtract : lớp thực hiện phép trừ, trung hòa giá trị pixel (như trừ đi một giá trị trung bình)
* MobileNetV2 : nãy chuyển đổi ảnh đầu vào thành 1 khối đặc trưng.
* global\_average\_pooling2d : tính toán giá trị trung bình trên các vị trí không gian của đặc trưng, biến đổi đầu ra từ MobileNetV2 thành một vector 1280 chiều.

**GlobalAveragePooling2D**

**⇒ Giảm chiều dữ liệu :  tính giá trị trung bình của các đặc trưng (feature maps) trong mỗi kênh của đầu ra từ lớp trước đó.**

**Kích thước (32, 5, 5, 1280) (trong đó 32 là kích thước lô, 5x5 là kích thước không gian, và 1280 là số kênh). ⇒ , đầu ra sẽ trở thành (32, 1280)**

**A screenshot of a computer

Description automatically generated**

**⇒** Số lượng các biến có thể huấn luyện trong mô hình. Kết quả sẽ là 2.

**Mục đích của Fine-tuning**

Huấn luyện một bộ phân loại (classifier) ở trên cùng của mô hình mà không cập nhật các trọng số (weights) của mô hình cơ bản.

⇒ Cải thiện hiệu suất bằng cách tinh chỉnh (fine-tune) các trọng số của các lớp trên cùng của mô hình đã được huấn luyện trước đó.

⇒ ĐIỀU CHỈNH CÁC TRỌNG SỐ ĐỂ CHUYỂN TỪ CÁC FEATURE MAP CHUNG SANG CÁC ĐẶC TRƯNG LIÊN QUAN CỤ THỂ ĐẾN TẬP DỮ LIỆU.

**Thời điểm Fine-tuning**

Nên được thực hiện sau khi bạn đã huấn luyện bộ phân loại ở trên cùng.

Nếu bạn thêm một bộ phân loại mới với trọng số ngẫu nhiên (randomly initialized classifier) và cố gắng huấn luyện tất cả các lớp cùng một lúc, độ lớn của các cập nhật gradient sẽ quá lớn.

⇒ Khiến mô hình đã được huấn luyện trước quên những gì nó đã học được từ trước.

**Lựa chọn Lớp để Tinh chỉnh - Lớp trên cùng**

Trong các mạng nơ-ron tích chập (convolutional networks), các lớp càng cao thường càng chuyên biệt hơn. Các lớp đầu tiên học các đặc trưng rất đơn giản và chung chung, có thể áp dụng cho hầu hết các loại hình ảnh.

Điều chỉnh những đặc trưng chuyên biệt này để làm việc hiệu quả hơn với tập dữ liệu mới mà không làm mất đi những kiến thức tổng quát mà mô hình đã học được.

**Tại sao cần tăng cường dữ liệu?**

**Tăng cường dữ liệu là điều cần thiết vì một số lý do:**

**Tăng kích thước tập dữ liệu: Bằng cách tạo hình ảnh mới từ hình ảnh hiện có, bạn có thể nhân đôi hiệu quả tập dữ liệu của mình.**

**Cải thiện độ tin cậy của mô hình: Tăng cường tạo ra tính biến thiên, giúp mô hình có khả năng phục hồi tốt hơn trước các điều kiện và nhiễu khác nhau.**

**Giảm tình trạng quá khớp: Với tập dữ liệu đào tạo đa dạng hơn, mô hình ít có khả năng ghi nhớ dữ liệu đào tạo và có khả năng khái quát hóa tốt hơn đối với dữ liệu chưa biết.**

**Các kỹ thuật tăng cường dữ liệu phổ biến**

**Sau đây là một số kỹ thuật tăng cường dữ liệu được sử dụng rộng rãi:**

**1. Xoay**

**Xoay hình ảnh theo nhiều góc khác nhau (ví dụ: -90, -45, 0, 45, 90 độ) giúp mô hình trở nên bất biến với hướng của các đối tượng.**

**2. Lật**

**Lật ngang: Lật hình ảnh từ trái sang phải.**

**Lật dọc: Lật hình ảnh từ trên xuống dưới.**

**3. Cắt**

**Cắt ngẫu nhiên: Chọn ngẫu nhiên một phần của hình ảnh.**

**Cắt ở giữa: Tập trung vào phần trung tâm của hình ảnh.**

**4. Scale**

**Tăng hoặc giảm tỷ lệ hình ảnh trong khi vẫn giữ nguyên tỷ lệ khung hình.**

**5. Translation**

**Chuyển đổi hình ảnh theo chiều ngang hoặc chiều dọc.**

**6. Shearing**

**Áp dụng các phép biến đổi shear để nghiêng hình ảnh.**

**7. Zooming**

**Phóng to hoặc thu nhỏ các phần của hình ảnh.**

**8. Brightness Adjustment**

**Tăng hoặc giảm độ sáng của hình ảnh một cách ngẫu nhiên.**

**9. Contrast Adjustment**

**Điều chỉnh độ tương phản của hình ảnh một cách ngẫu nhiên.**

**10. Saturation Adjustment**

**Điều chỉnh ngẫu nhiên các mức độ bão hòa của hình ảnh.**

**11. Hue Adjustment**

**Thay đổi nhẹ sắc thái của hình ảnh.**

**12. Noise Injection**

**Thêm nhiễu ngẫu nhiên (Gaussian, salt and pepper, v.v.) vào hình ảnh.**

**13. Blurring**

**Áp dụng các hiệu ứng làm mờ như Gaussian blur.**

**14. Sharpness Adjustment**

**Điều chỉnh ngẫu nhiên độ sắc nét của hình ảnh.**

**15. Color Jitter**

**Thay đổi ngẫu nhiên độ sáng, độ tương phản, độ bão hòa và sắc thái.**

**16. Cutout**

**Ngẫu nhiên che khuất các vùng hình vuông của hình ảnh.**

**17. Elastic Deformation**

**Áp dụng các phép biến đổi đàn hồi để làm cong nhẹ hình ảnh.**

**18. Random Erasesing**

**Xóa ngẫu nhiên các phần của hình ảnh, làm cho một số vùng bằng không.**

**19. Grayscale**

**Chuyển đổi hình ảnh sang grayscale.**

**20. Mixup**

**Kết hợp hai hình ảnh với một tỷ lệ nhất định để tạo ra một hình ảnh mới.**

**# Các biến đổi cơ bản**

**augmented\_images.append(datagen.random\_transform(img)) # Tăng cường bằng ImageDataGenerator**

**# Các kỹ thuật khác**

**pil\_img = Image.fromarray(img)**

**augmented\_images.append(np.array(random\_crop(pil\_img))) # Cắt ngẫu nhiên**

**augmented\_images.append(np.array(adjust\_brightness(pil\_img))) # Điều chỉnh độ sáng**

**augmented\_images.append(np.array(adjust\_contrast(pil\_img))) # Điều chỉnh độ tương phản**

**augmented\_images.append(np.array(adjust\_saturation(pil\_img))) # Điều chỉnh độ bão hòa**

**augmented\_images.append(np.array(adjust\_hue(pil\_img))) # Điều chỉnh sắc thái**

**augmented\_images.append(np.array(add\_noise(pil\_img))) # Thêm nhiễu**

**augmented\_images.append(np.array(blur\_image(pil\_img))) # Làm mờ ảnh**

**augmented\_images.append(np.array(adjust\_sharpness(pil\_img))) # Điều chỉnh độ sắc nét**

**augmented\_images.append(np.array(grayscale(pil\_img))) # Chuyển đổi sang đen trắng**

**augmented\_images.append(np.array(cutout(pil\_img))) # Cắt các vùng ngẫu nhiên**

**augmented\_images.append(np.array(elastic\_deformation(pil\_img))) # Biến dạng đàn hồi**

**1. Thiết lập ImageDataGenerator**

python

Sao chép mã

datagen = ImageDataGenerator(

rotation\_range=45,

width\_shift\_range=0.2,

height\_shift\_range=0.2,

shear\_range=0.2,

zoom\_range=0.2,

horizontal\_flip=True,

vertical\_flip=True,

fill\_mode='nearest'

)

**Giải thích**: ImageDataGenerator được sử dụng để tạo dữ liệu hình ảnh tăng cường cho các mô hình học sâu. Các tham số cụ thể:

* rotation\_range=45: Quay ngẫu nhiên hình ảnh trong khoảng từ -45 đến 45 độ.
* width\_shift\_range=0.2, height\_shift\_range=0.2: Dịch chuyển ngang và dọc ngẫu nhiên hình ảnh tối đa 20% so với kích thước ban đầu.
* shear\_range=0.2: Áp dụng biến dạng cắt xén (shear) lên hình ảnh.
* zoom\_range=0.2: Phóng to hoặc thu nhỏ ngẫu nhiên hình ảnh trong phạm vi 20%.
* horizontal\_flip=True, vertical\_flip=True: Lật ngang và lật dọc ngẫu nhiên.
* fill\_mode='nearest': Điền vào các pixel còn thiếu sau khi biến đổi bằng cách lặp lại các pixel gần nhất.

**2. Hàm random\_crop**

python

Sao chép mã

def random\_crop(image, size=(224, 224)):

width, height = image.size

new\_width, new\_height = size

left = random.randint(0, width - new\_width)

top = random.randint(0, height - new\_height)

return image.crop((left, top, left + new\_width, top + new\_height))

**Giải thích**: Hàm này cắt ngẫu nhiên một phần hình ảnh với kích thước cố định (224x224). left và top xác định vị trí bắt đầu cắt trong hình ảnh gốc.

**3. Hàm adjust\_brightness**

python

Sao chép mã

def adjust\_brightness(image):

enhancer = ImageEnhance.Brightness(image)

return enhancer.enhance(random.uniform(0.5, 1.5))

**Giải thích**: Điều chỉnh độ sáng của hình ảnh. Giá trị ngẫu nhiên trong khoảng từ 0.5 (giảm sáng) đến 1.5 (tăng sáng).

**4. Hàm adjust\_contrast**

python

Sao chép mã

def adjust\_contrast(image):

enhancer = ImageEnhance.Contrast(image)

return enhancer.enhance(random.uniform(0.5, 1.5))

**Giải thích**: Điều chỉnh độ tương phản của hình ảnh. Giá trị ngẫu nhiên từ 0.5 (giảm tương phản) đến 1.5 (tăng tương phản).

**5. Hàm adjust\_saturation**

python

Sao chép mã

def adjust\_saturation(image):

enhancer = ImageEnhance.Color(image)

return enhancer.enhance(random.uniform(0.5, 1.5))

**Giải thích**: Điều chỉnh độ bão hòa màu của hình ảnh. Giá trị ngẫu nhiên trong khoảng từ 0.5 (màu ít bão hòa) đến 1.5 (màu bão hòa hơn).

**6. Hàm adjust\_hue**

python

Sao chép mã

def adjust\_hue(image):

np\_image = np.array(image.convert('HSV'))

np\_image[:, :, 0] = np\_image[:, :, 0] + random.randint(-10, 10)

return Image.fromarray(np\_image, 'HSV').convert('RGB')

**Giải thích**: Điều chỉnh màu sắc (hue) của hình ảnh. Màu sắc được thay đổi trong khoảng từ -10 đến 10 đơn vị trên kênh màu H (Hue) trong hệ màu HSV.

**7. Hàm add\_noise**

python

Sao chép mã

def add\_noise(image):

np\_image = np.array(image)

noise = np.random.normal(0, 25, np\_image.shape)

np\_image = np.clip(np\_image + noise, 0, 255).astype(np.uint8)

return Image.fromarray(np\_image)

**Giải thích**: Thêm nhiễu Gaussian vào hình ảnh. Nhiễu có độ lệch chuẩn là 25, các giá trị pixel được giữ trong khoảng [0, 255].

**8. Hàm blur\_image**

python

Sao chép mã

def blur\_image(image):

return image.filter(ImageFilter.GaussianBlur(radius=random.uniform(0.5, 2.0)))

**Giải thích**: Làm mờ hình ảnh bằng bộ lọc Gaussian với bán kính ngẫu nhiên từ 0.5 đến 2.0.

**9. Hàm adjust\_sharpness**

python

Sao chép mã

def adjust\_sharpness(image):

enhancer = ImageEnhance.Sharpness(image)

return enhancer.enhance(random.uniform(0.5, 2.0))

**Giải thích**: Điều chỉnh độ sắc nét của hình ảnh. Giá trị từ 0.5 (ít sắc nét hơn) đến 2.0 (sắc nét hơn).

**10. Hàm grayscale**

python

Sao chép mã

def grayscale(image):

return image.convert('L').convert('RGB')

**Giải thích**: Chuyển đổi hình ảnh thành thang độ xám và sau đó chuyển lại thành định dạng RGB.

**11. Hàm cutout**

python

Sao chép mã

def cutout(image, num\_patches=1, patch\_size=32):

np\_image = np.array(image)

for \_ in range(num\_patches):

x = random.randint(0, np\_image.shape[1] - patch\_size)

y = random.randint(0, np\_image.shape[0] - patch\_size)

np\_image[y:y+patch\_size, x:x+patch\_size] = 0

return Image.fromarray(np\_image)

**Giải thích**: Thực hiện kỹ thuật Cutout, thêm các vùng đen vào hình ảnh để làm mất thông tin. Kích thước của mỗi vùng là patch\_size, số lượng là num\_patches.

**12. Hàm elastic\_deformation**

python

Sao chép mã

def elastic\_deformation(image):

return image # Placeholder cho biến dạng đàn hồi

**Giải thích**: Đây là một phần dành để thực hiện biến dạng đàn hồi, tuy nhiên, hiện tại chỉ là một placeholder, chưa có logic cụ thể.

4o