**· VGG16 với Transfer Learning:**

· Chúng ta sử dụng mô hình VGG16 đã được huấn luyện sẵn trên bộ dữ liệu ImageNet và loại bỏ lớp phân loại cuối cùng (lớp softmax).

VGG16 sẽ được sử dụng để trích xuất đặc trưng từ hình ảnh đầu vào.

Các lớp của VGG16 sẽ được đóng băng trong quá trình huấn luyện (không cập nhật các trọng số trong VGG16), điều này giúp tiết kiệm thời gian huấn luyện và tránh overfitting.

**· Lớp toàn cục (GlobalAveragePooling2D):**

· Lớp này giúp giảm kích thước của các đặc trưng không gian và giảm số lượng tham số trong mô hình.

**· Lớp Dropout:**

· Để giảm overfitting, chúng ta thêm một lớp Dropout với tỉ lệ 50% giữa các lớp Dense.

**· Lớp đầu ra (Dense):**

Lớp đầu ra có một nơ-ron với hàm kích hoạt sigmoid để phân loại ra hai lớp: "rác tái chế" và "rác không tái chế".

### 1. ****Đầu vào (Input)****:

Mô hình của bạn nhận vào các ảnh có kích thước (224, 224, 3) – nghĩa là ảnh có chiều cao 224 pixel, chiều rộng 224 pixel, và 3 kênh màu (RGB).

* **Công thức toán học liên quan:** Các pixel của ảnh được mã hóa thành một ma trận 3D với giá trị từ 0 đến 255 cho mỗi kênh màu (RGB).

Giả sử ảnh đầu vào là một tensor X có kích thước (224, 224, 3).

### 2. ****Lớp tích chập (Convolutional Layer)****:

Các lớp **Conv2D** sẽ áp dụng **phép toán tích chập** (convolution) lên ảnh đầu vào. Lý do mô hình sử dụng các lớp này là để học các đặc trưng cơ bản từ ảnh như cạnh, góc, hoặc các mẫu đơn giản.

#### Phép toán tích chập (Convolution):

Giả sử bạn có một bộ lọc (filter) W có kích thước (3, 3) và ảnh đầu vào X có kích thước (224, 224, 3), khi bộ lọc được áp dụng lên ảnh, phép toán **convolution** sẽ trượt bộ lọc qua ảnh và tính toán phép nhân chéo giữa bộ lọc và một phần ảnh (mảng con) tại mỗi bước.

Công thức toán học cơ bản của phép toán tích chập:

Trong đó:

* Y(i,j)Y(i,j)Y(i,j) là giá trị tại pixel (i,j)(i,j)(i,j) của ảnh đầu ra sau khi áp dụng bộ lọc WWW.
* W(p,q)W(p,q)W(p,q) là trọng số của bộ lọc tại vị trí (p,q)(p,q)(p,q).
* X(i+p,j+p,k)X(i+p, j+p, k)X(i+p,j+p,k) là giá trị pixel của ảnh đầu vào tại vị trí tương ứng.

Sau khi tích chập, một giá trị được tạo ra cho mỗi vị trí của ảnh, và cuối cùng một ảnh mới được tạo ra (gọi là **feature map**).

#### Lý do sử dụng nhiều bộ lọc:

* Việc sử dụng nhiều bộ lọc (ví dụ, 32 bộ lọc, 64 bộ lọc...) giúp mô hình học được nhiều loại đặc trưng khác nhau (ví dụ: cạnh, kết cấu, các vùng có màu sắc đặc biệt).

### 3. ****Max Pooling****:

Sau mỗi lớp **Conv2D**, có một lớp **MaxPooling2D** để giảm kích thước của đặc trưng đã học và giảm tính toán trong các lớp tiếp theo. Pooling giúp **tóm tắt thông tin quan trọng** và giảm độ phức tạp.

Công thức toán học của **MaxPooling** (với kích thước kernel là 2x2):

* Tại mỗi vùng 2x2 của ảnh, hàm **MaxPooling** sẽ chọn giá trị lớn nhất làm đại diện.

Lý do sử dụng pooling:

* Giảm kích thước ảnh (down-sampling) giúp tiết kiệm bộ nhớ và tính toán.
* Giảm overfitting bằng cách giữ lại các đặc trưng quan trọng nhất.

### 4. ****Flatten****:

Sau khi các lớp tích chập và pooling, bạn có một ma trận các đặc trưng học được. Tuy nhiên, các lớp **Dense** (fully connected) yêu cầu đầu vào phải là một vector 1D. Vì vậy, lớp **Flatten** sẽ chuyển ma trận 2D hoặc 3D thành một vector 1D.

### 5. ****Lớp Fully Connected (Dense Layer)****:

Các lớp **Dense** có nhiệm vụ học các mối quan hệ phức tạp giữa các đặc trưng đã được học từ các lớp trước. Mỗi lớp Dense gồm nhiều **neuron** (tế bào thần kinh), và mỗi neuron nhận đầu vào từ tất cả các neuron của lớp trước.

Công thức tính toán của lớp Dense:

y=W⋅x+by = W \cdot x + by=W⋅x+b

* WWW là ma trận trọng số của lớp.
* xxx là vector đầu vào (được lấy từ các đặc trưng đã học).
* bbb là vector bias.
* yyy là đầu ra của lớp, và giá trị này được đưa qua một hàm kích hoạt (như ReLU hoặc Sigmoid) để tạo ra đầu ra của lớp.

#### Lý do có nhiều lớp Dense:

* **Lớp Dense đầu tiên (128 neuron)** học các mối quan hệ phức tạp giữa các đặc trưng từ các lớp trước.
* **Lớp Dense thứ hai (64 neuron)** giúp mô hình học sâu hơn.
* **Lớp Dense thứ ba (32 neuron)** tiếp tục làm sâu sắc thêm các mối quan hệ.
* Các lớp **Dropout** giúp giảm overfitting bằng cách ngắt kết nối ngẫu nhiên một số neuron trong quá trình huấn luyện.

### 6. ****Lớp đầu ra (Output Layer)****:

python

Sao chépChỉnh sửa

layers.Dense(1, activation='sigmoid')

* Lớp cuối cùng có một neuron với hàm kích hoạt **Sigmoid**, vì bài toán của bạn là phân loại nhị phân (chẳng hạn phân loại "rác" hoặc "không phải rác").
* **Sigmoid activation** giúp mô hình đưa ra xác suất thuộc lớp "1" (rác) hoặc lớp "0" (không phải rác). Output này sẽ có giá trị trong khoảng [0,1], và nếu giá trị này lớn hơn một ngưỡng (thường là 0.5), mô hình sẽ dự đoán là "rác", nếu nhỏ hơn thì sẽ dự đoán là "không phải rác".

### Tại sao phải dùng từng lớp, từng số lượng neuron?

* **Các lớp Conv2D và MaxPooling2D**: Cần thiết để mô hình có thể học và trích xuất các đặc trưng từ ảnh. Mỗi lớp Conv2D học các đặc trưng khác nhau (cạnh, kết cấu, màu sắc). MaxPooling giúp giảm kích thước và giữ lại các đặc trưng quan trọng.
* **Sử dụng nhiều lớp (Convolutional Layers)** giúp mô hình học các đặc trưng ngày càng phức tạp hơn từ ảnh đầu vào.
* **Lớp Fully Connected (Dense)**: Cần thiết để mô hình học các mối quan hệ giữa các đặc trưng đã học được từ các lớp trước. Các lớp Dense càng sâu, mô hình có thể học các mối quan hệ càng phức tạp, nhưng cũng cần phải cẩn thận với overfitting.
* **Dropout**: Giúp ngăn chặn overfitting, khi mà mô hình học quá nhiều chi tiết từ bộ dữ liệu huấn luyện và không tổng quát tốt trên bộ dữ liệu mới.
* **Số lượng neuron trong mỗi lớp**: Thường được quyết định dựa trên độ phức tạp của bài toán. Mô hình càng phức tạp, số lượng neuron và lớp càng nhiều để học được các mối quan hệ phức tạp hơn giữa các đặc trưng.

### Tổng kết:

* Mô hình này sử dụng **lớp tích chập (Conv2D)** để học các đặc trưng từ ảnh, sau đó sử dụng **max pooling** để giảm kích thước ảnh.
* Sau đó, các đặc trưng được học từ các lớp tích chập được truyền qua **các lớp fully connected (Dense)** để học các mối quan hệ phức tạp giữa các đặc trưng này.
* Cuối cùng, mô hình sử dụng **hàm kích hoạt sigmoid** để đưa ra dự đoán nhị phân (rác hay không phải rác).

Mỗi lớp và số lượng neuron được thiết kế để tối ưu hóa khả năng học các đặc trưng từ ảnh và tránh overfitting, nhằm đạt được hiệu suất tốt nhất cho bài toán phân loại nhị phân của bạn.