# Core Concepts

## Basic

### Giải thích các giá trị loss, accuracy, val\_loss, val\_accuracy?

- Loss (hàm mất mát): đo lường độ chênh lệch giữa giá trị dự đoán và giá trị thực tế. Giá trị này **càng thấp càng tốt**.

- Accuracy (độ chính xác): đo lường tỷ lệ dự đoán đúng trên tổng số mẫu. Giá trị này **càng cao càng tốt**.

- Val\_loss: Đây là giá trị hàm mất mát (loss) trên tập kiểm tra (validation data). Đây là thông số quan trọng để đánh giá mô hình có đang overfit hay không (tức là mô hình có thể đang học quá mức đặc trưng của dữ liệu huấn luyện và không tổng quát tốt cho dữ liệu mới).

- Val\_accuracy: Đây là độ chính xác trên tập kiểm tra (validation data), thể hiện mô hình dự đoán đúng bao nhiêu phần trăm trên dữ liệu kiểm tra.

- Tóm lại:

* Loss nên thấp và gần nhau giữa train/val.
* Accuracy nên cao và không chênh lệch quá nhiều.

### Phân biệt giữa train\_loss và val\_loss trong quá trình huấn luyện?

train\_loss: Mất mát trên dữ liệu huấn luyện, cho biết mô hình đang học tốt như thế nào trên dữ liệu huấn luyện.

val\_loss: Mất mát trên dữ liệu validation, cho biết mô hình có khả năng tổng quát tốt hay không.

### Loss chênh lệch lớn?

- Loss chênh lệch lớn→ Mô hình đang overfit mạnh.

### Accuracy chênh lệch lớn?

- Accuracy chênh lệch lớn→ Mô hình nhớ dữ liệu train nhưng không áp dụng được vào dữ liệu mới.

### Một số cách giảm overfitting?

- **Data augmentation** (nếu bài toán là xử lý ảnh): Tăng cường dữ liệu bằng cách tạo ra các biến thể của hình ảnh gốc.

- **Dropout:** Bỏ qua ngẫu nhiên một số nơ-ron trong quá trình huấn luyện để tránh mô hình quá phụ thuộc vào một số nơ-ron cụ thể. Ví dụ: Dropout(0.5) để giảm ghi nhớ quá mức.

- **Sử dụng lớp FC (Fully Connected) ít hơn:** Lớp FC quá nhiều có thể dẫn đến overfitting.

- **Sử dụng early stopping:** Dừng huấn luyện nếu mô hình không cải thiện sau một số epoch nhất định.

- **Lớp BatchNormalization:** Giúp chuẩn hóa đầu vào của mỗi lớp và giảm overfitting.

- **Regularization:** là một kỹ thuật trong học máy nhằm giảm thiểu **overfitting** (quá khớp) của mô hình. Overfitting xảy ra khi mô hình học quá kỹ các đặc trưng của dữ liệu huấn luyện, dẫn đến việc không tổng quát được cho dữ liệu mới. Ví dụ **L2 Regularization** giúp giảm độ phức tạp của mô hình bằng cách phạt các trọng số lớn.

- Tăng kích thước dữ liệu huấn luyện để giúp mô hình học tốt hơn.

- Giảm bớt các lớp của mô hình cơ sở (ResNet50) thông qua fine-tuning.

### Epoch là gì?

- Là một thuật ngữ chỉ một vòng lặp mà mô hình sẽ duyệt toàn bộ tập dữ liệu huấn luyện để cập nhật trọng số của nó.

### Batch size là gì?

- Số lượng dữ liệu sử dụng trong 1 lần để cập nhật tham số.

- Khi huấn luyện code bằng python, bạn sẽ thấy dòng log như sau:

* “Found **2187** images belonging to 7 classes.”
* “**Epoch 1**/**5** **69**/**69** [==============================] - 66s 961ms/step - **loss**: 0.7071 - **accuracy**: 0.7494 - **val\_loss**: 3.9661 - **val\_accuracy**: 0.1365 - lr: 1.0000e-04”

- Số **69/69** xuất hiện trong log training của TensorFlow/Keras, có ý nghĩa như sau:

* **69** (bên trái dấu /): Là số **batch hiện tại** mà mô hình đang xử lý trong epoch đó.
* **69** (bên phải dấu /): Là tổng số **batch** trong toàn bộ tập dữ liệu huấn luyện.

- Trong quá trình training, dữ liệu được chia thành nhiều **batch nhỏ** để xử lý thay vì đưa toàn bộ dataset vào một lần. Số batch này được tính bằng công thức:

Bạn có **2187 ảnh trong tập train**, nếu batch size = 32, số batch sẽ là 2187/32 ~ 69

### Các tham số cấu hình khi tăng cường ảnh (data augementation)

**rescale:**

* **Mục đích:** Điều chỉnh giá trị pixel của ảnh. Đây là một tỉ lệ nhân với giá trị pixel ban đầu. Thường được dùng để chuẩn hóa giá trị pixel của ảnh (ví dụ: đưa giá trị pixel về [0,1] nếu chúng có giá trị gốc từ 0 đến 255).
* **Đơn vị:** Không có đơn vị cụ thể, đây là phép chia (hoặc nhân) giá trị pixel.
* **Range:** Phụ thuộc vào giá trị bạn cung cấp. Ví dụ 1.0/255 sẽ chia mọi pixel ảnh cho 255, đưa giá trị ảnh vào khoảng [0, 1].

**rotation\_range:**

* **Mục đích:** Xoay ảnh ngẫu nhiên trong một phạm vi góc xác định (đơn vị là độ).
* **Đơn vị:** Độ (°).
* **Range:** Tham số này xác định phạm vi góc có thể thay đổi trong khoảng [-X, +X] (X là giá trị bạn đặt). Ví dụ rotation\_range=40 có nghĩa là ảnh sẽ được xoay ngẫu nhiên từ -40 đến +40 độ.

**width\_shift\_range:**

* **Mục đích:** Di chuyển ảnh theo chiều ngang (trái/phải) một cách ngẫu nhiên. Tham số này cho phép bạn xác định phần trăm chiều rộng ảnh có thể thay đổi.
* **Đơn vị:** Tỉ lệ phần trăm chiều rộng của ảnh.
* **Range:** Từ -1.0 đến 1.0. Ví dụ, width\_shift\_range=0.2 có nghĩa là ảnh có thể dịch chuyển sang trái hoặc phải tối đa 20% chiều rộng ảnh.

**height\_shift\_range:**

* **Mục đích:** Di chuyển ảnh theo chiều dọc (lên/xuống) một cách ngẫu nhiên.
* **Đơn vị:** Tỉ lệ phần trăm chiều cao của ảnh.
* **Range:** Từ -1.0 đến 1.0. Ví dụ height\_shift\_range=0.2 có nghĩa là ảnh có thể dịch chuyển lên hoặc xuống tối đa 20% chiều cao của ảnh.

**shear\_range:**

* **Mục đích:** Áp dụng phép cắt (shearing) cho ảnh, tạo ra sự méo mó, thay đổi hình dạng của ảnh mà không thay đổi diện tích.
* **Đơn vị:** Độ (°).
* **Range:** Từ -180° đến +180°. Ví dụ shear\_range=0.2 có thể là giá trị được quy định bằng tỉ lệ nhỏ hơn 1, nhưng về lý thuyết có thể là một số góc lớn hơn.

**zoom\_range:**

* **Mục đích:** Phóng to hoặc thu nhỏ ảnh ngẫu nhiên trong một phạm vi nhất định.
* **Đơn vị:** Tỉ lệ kích thước ảnh.
* **Range:** Tham số có thể nhận giá trị là một số (ví dụ zoom\_range=0.2 có thể phóng to hoặc thu nhỏ ảnh từ 80% đến 120% kích thước ban đầu), hoặc nhận một tuple (min\_zoom, max\_zoom). Nếu zoom\_range=0.2, ảnh có thể thu nhỏ xuống 80% kích thước ban đầu hoặc phóng to lên 120%.

**horizontal\_flip:**

* **Mục đích:** Lật ảnh theo chiều ngang (trái/phải) một cách ngẫu nhiên.
* **Đơn vị:** Không có đơn vị cụ thể. Đây là thao tác lật ảnh.
* **Range:** Chỉ có hai giá trị: True (lật ảnh) hoặc False (không lật ảnh). Không có tỉ lệ hay giá trị số học.

**brightness\_range:**

* **Mục đích:** Cho phép thay đổi độ sáng của ảnh bằng cách nhân các giá trị pixel với một hệ số ngẫu nhiên trong phạm vi xác định. Nếu bạn cung cấp một phạm vi [min, max], độ sáng của mỗi ảnh sẽ được điều chỉnh bằng cách nhân giá trị pixel với một giá trị ngẫu nhiên trong phạm vi này.
* **Đơn vị:** Không có đơn vị vật lý cụ thể.
* **Range:** Đây là một danh sách hoặc tuple gồm hai giá trị [min\_brightness, max\_brightness]. Tham số brightness\_range=[0.5, 1.5] có nghĩa là độ sáng của ảnh sẽ được điều chỉnh ngẫu nhiên sao cho giá trị pixel của ảnh có thể bị giảm xuống 50% (nếu hệ số là 0.5) hoặc tăng lên 150% (nếu hệ số là 1.5) so với giá trị gốc.

**fill\_mode:**

* **Mục đích:** Cách điền các pixel bị thiếu sau khi áp dụng các phép biến đổi (như di chuyển hoặc xoay ảnh).
* **Đơn vị:** Không có đơn vị cụ thể.
* **Range:** Các giá trị có thể là:
  + 'nearest': Lấp đầy bằng pixel gần nhất.
  + 'constant': Lấp đầy bằng một giá trị cố định (được chỉ định bằng tham số cval).
  + 'reflect': Lấp đầy bằng cách phản chiếu ảnh.
  + 'wrap': Lấp đầy bằng cách lặp lại ảnh.

### Fine-tuning trên ResNet50 là gì?

- Là quá trình tiếp tục huấn luyện một mô hình đã được huấn luyện trước bằng cách điều chỉnh một số lớp cụ thể thay vì huấn luyện từ đầu. Cách làm này giúp mô hình học thêm các đặc trưng cụ thể của dữ liệu mới mà không làm mất đi kiến thức tổng quát từ ImageNet. Nếu ta không fine-tune, mô hình chỉ sử dụng các đặc trưng có sẵn của ResNet50 mà không điều chỉnh cho bài toán cụ thể.

Ví dụ: for layer in base\_model.layers[:-50]: # 🔥 Chỉ mở khóa 50 lớp cuối  
 layer.trainable = False

- Fine-tune 50 lớp cuối - 50 lớp cuối cùng của ResNet50 sẽ tiếp tục được huấn luyện trên dữ liệu mới thay vì giữ nguyên trọng số cũ.

### Tại sao mô hình có thể cần fine-tuning thay vì huấn luyện từ đầu?

Fine-tuning giúp sử dụng các trọng số đã học từ các bài toán trước (chẳng hạn như ImageNet), giảm thiểu số lượng dữ liệu cần thiết và tăng tốc quá trình huấn luyện.

### Khi nào ta cần sử dụng ModelCheckpoint và EarlyStopping trong huấn luyện mô hình?

**ModelCheckpoint:** Giúp lưu lại mô hình tốt nhất trong quá trình huấn luyện khi giá trị val\_loss (loss trên tập validation) thấp nhất.

**EarlyStopping:** Dừng huấn luyện sớm nếu không có sự cải thiện nào trong một số epoch nhất định để tránh overfitting.

### Tại sao ta sử dụng class\_weight khi huấn luyện mô hình?

Khi các lớp trong dữ liệu có sự phân bổ không đồng đều, việc sử dụng class\_weight giúp mô hình học các lớp ít mẫu tốt hơn bằng cách tăng cường trọng số cho các lớp đó.

### Giải thích về confusion\_matrix và classification\_report?

**Confusion matrix**: Bảng cho thấy số lượng mẫu được phân loại đúng và sai cho từng lớp.

**Classification report**: Cung cấp các chỉ số như precision, recall, F1-score cho từng lớp, giúp đánh giá chi tiết mô hình.

### Chỉ số Precision, Recall, và F1-score là gì?

**Precision:** Tỷ lệ mẫu dự đoán đúng so với tất cả các mẫu được dự đoán là đúng.

**Recall:** Tỷ lệ mẫu dự đoán đúng so với tất cả các mẫu thực sự đúng.

**F1-score:** Trung bình hài hòa của precision và recall, giúp đánh giá hiệu suất của mô hình khi có sự mất cân bằng giữa các lớp.

### Learning rate là gì và tại sao cần điều chỉnh nó?

- Learning rate là tốc độ mà mô hình thay đổi trọng số trong quá trình huấn luyện. Nếu learning rate quá cao, mô hình có thể bỏ qua các giá trị tối ưu. Nếu quá thấp, mô hình sẽ học rất chậm.

### Tại sao ta cần sử dụng Adam optimizer?

- **Adam** là một thuật toán tối ưu hóa được sử dụng phổ biến trong huấn luyện các mạng nơ-ron, vì nó kết hợp ưu điểm của các phương pháp tối ưu hóa khác như SGD và AdaGrad, giúp cải thiện tốc độ hội tụ.

### Class\_mode trong việc xác địch nhãn trong xử lý dữ liệu đầu vào?

Các giá trị có thể của class\_mode:

* 'categorical': Đây là cách sử dụng phổ biến khi bạn có bài toán phân loại đa lớp (multi-class classification). Dữ liệu nhãn sẽ được trả về dưới dạng one-hot encoding. Ví dụ, nếu bạn có 3 lớp (class 0, class 1, và class 2), nhãn sẽ được trả về dưới dạng [1, 0, 0] cho class 0, [0, 1, 0] cho class 1, và [0, 0, 1] cho class 2.
* 'binary': Dùng cho bài toán phân loại nhị phân (binary classification). Nhãn sẽ được trả về dưới dạng giá trị nhị phân (0 hoặc 1).
* 'sparse': Cũng dùng cho bài toán phân loại đa lớp, nhưng thay vì one-hot encoding, nhãn sẽ là một số nguyên chỉ ra lớp của mẫu. Ví dụ, đối với 3 lớp, bạn sẽ có nhãn là 0, 1, hoặc 2 thay vì [1, 0, 0], [0, 1, 0], hoặc [0, 0, 1].
* None: Nếu bạn không cần nhãn (ví dụ, trong các bài toán không giám sát), bạn có thể để class\_mode=None.

### Sự khác biệt giữa categorical\_crossentropy và binary\_crossentropy là gì?

**Categorical crossentropy** được sử dụng trong các bài toán phân loại nhiều lớp (multi-class classification).

**Binary crossentropy** được sử dụng trong các bài toán phân loại nhị phân (binary classification).

### Tại sao cần phải chuẩn hóa dữ liệu (rescale) về phạm vi [0, 1]?

Việc chuẩn hóa giúp đảm bảo rằng giá trị pixel của ảnh nằm trong một dải giá trị thống nhất, từ đó giúp mô hình huấn luyện hiệu quả hơn và giảm thời gian hội tụ.

### Có thể sử dụng các kỹ thuật tăng cường ảnh (augmentation) cho dữ liệu kiểm tra (tập test) không?

Không, vì việc tăng cường dữ liệu kiểm tra có thể làm sai lệch kết quả đánh giá mô hình. Dữ liệu kiểm tra phải phản ánh đúng sự phân bố thực tế của các lớp.

### Sự khác biệt giữa steps\_per\_epoch và validation\_steps khi huấn luyện là gì?

steps\_per\_epoch xác định số lượng bước trong một epoch (số lần cập nhật trọng số từ dữ liệu huấn luyện).

validation\_steps xác định số lượng bước trong mỗi epoch cho tập validation (số lần kiểm tra mô hình trên dữ liệu validation).

### Tại sao đôi khi ta cần sử dụng compute\_class\_weight trong huấn luyện?

compute\_class\_weight giúp tính toán trọng số cho từng lớp để xử lý các vấn đề mất cân bằng lớp trong dữ liệu, giúp mô hình học tốt hơn đối với các lớp ít dữ liệu.

### Các layer phổ biến trong CNN

#### Layer cơ bản

Conv2D, MaxPooling2D, Flatten, Dense là nền tảng của hầu hết các CNN.

#### Layer tối ưu

DepthwiseConv2D, SeparableConv2D giúp giảm tính toán, phù hợp cho thiết bị hạn chế tài nguyên.

#### Layer nâng cao

Attention, MultiHeadAttention xuất hiện trong các kiến trúc hiện đại như Vision Transformer.

#### Layer hỗ trợ

BatchNormalization, Dropout cải thiện quá trình huấn luyện và hiệu suất mô hình.

#### Layer đặc thù

Conv1D, Conv3D, LSTM/GRU mở rộng ứng dụng của CNN sang dữ liệu chuỗi và 3D.

#### Bảng tra cứu các layer

| **Layer** | **Mục đích** | **Ứng dụng** |
| --- | --- | --- |
| **Conv2D** | Trích xuất đặc trưng từ ảnh bằng cách áp dụng các bộ lọc (filter). | Cơ bản trong hầu hết các CNN như VGG, ResNet. |
| **MaxPooling2D** | Giảm kích thước không gian bằng cách lấy giá trị lớn nhất trong vùng. | Giảm kích thước feature map trong CNN. |
| **AveragePooling2D** | Giảm kích thước bằng cách lấy giá trị trung bình trong vùng. | Thay thế MaxPooling trong một số kiến trúc. |
| **GlobalAveragePooling2D** | Thay thế Flatten bằng cách lấy trung bình toàn cục (giảm chiều dữ liệu). | Dùng trong ResNet thay cho FC Layers |
| **GlobalMaxPooling2D** | Lấy giá trị lớn nhất của toàn bộ feature map. | Thay thế Flatten trong một số kiến trúc. |
| **Flatten** | Chuyển tensor đa chiều thành vector 1 chiều để kết nối với Dense layers. | Kết nối feature maps với fully connected layers. |
| **Dense** | Layer fully connected để học đặc trưng toàn cục và thực hiện phân loại. | **Layer cuối** trong CNN cho bài toán phân loại. |
| **Dropout** | Ngăn overfitting bằng cách ngẫu nhiên tắt một số neuron trong huấn luyện | Đặt giữa các Dense layers để giảm overfitting. |
| **SpatialDropout2D** | Tắt nguyên một kênh màu (channel) thay vì neuron riêng lẻ. | |  | | --- | | Hiệu quả với dữ liệu ảnh có kênh màu tương quan cao (VD: Segmentation). | |  | |
| **BatchNormalization** | Chuẩn hóa đầu ra của layer trước để ổn định và tăng tốc huấn luyện. | Sau Conv2D hoặc Dense layers trong ResNet, EfficientNet. |
| **Activation** | Thêm hàm kích hoạt (sigmoid, tanh) để tạo tính phi tuyến. | Sigmoid cho bài toán nhị phân, ReLU trong hầu hết CNN. |
| **LeakyReLU** | Biến thể của ReLU, cho phép gradient nhỏ khi giá trị âm. | Thay thế ReLU để tránh dying ReLU. |
| **PReLU** | Parametric ReLU, hệ số cho giá trị âm được học trong huấn luyện. | Cải thiện hiệu suất trong các mạng sâu. |
| **ELU** | Exponential Linear Unit, giúp giảm vanishing gradient. | Thay thế ReLU trong một số kiến trúc. |
| **Softmax** | Chuyển đổi đầu ra thành xác suất cho bài toán phân loại đa lớp. | Layer đầu ra trong bài toán phân loại. |
| **DepthwiseConv2D** | Tích chập riêng từng kênh màu, giảm tham số. | MobileNet, mô hình nhẹ cho mobile. |
| **SeparableConv2D** | Kết hợp Depthwise + Pointwise Conv, tối ưu tính toán. | EfficientNet, mô hình hiệu quả. |
| **Conv2DTranspose** | Tích chập ngược để tăng kích thước không gian (deconvolution). | Mô hình U-Net cho segmentation, GANs. |
| **UpSampling2D** | Tăng kích thước không gian (dùng trong mạng Generative hoặc Segmentation). | Mô hình U-Net, super-resolution. |
| **ZeroPadding2D** | Thêm padding 0 vào viền để kiểm soát kích thước đầu ra. | Duy trì kích thước sau Conv2D. |
| **Cropping2D** | Cắt bớt một phần của đầu vào để điều chỉnh kích thước. | Điều chỉnh kích thước trong các kiến trúc phức tạp. |
| **Concatenate** | Ghép các tensor theo một trục, giữ lại nhiều thông tin. | U-Net, DenseNet. |
| **Add** | Cộng các tensor, thường dùng cho skip connections. | ResNet, giúp gradient lan truyền tốt hơn. |
| **Reshape** | Thay đổi hình dạng của tensor mà không thay đổi dữ liệu. | Chuẩn bị dữ liệu cho layer tiếp theo. |
| **Lambda** | Áp dụng hàm tùy ý lên đầu vào. | Thực hiện các biến đổi đặc thù không có sẵn trong các layer chuẩn. |
| **Conv1D** | Tích chập 1D cho dữ liệu chuỗi. | Xử lý tín hiệu âm thanh, chuỗi thời gian. |
| **Conv3D** | Tích chập 3D cho dữ liệu video hoặc ảnh 3D. | Phân tích video, y học (MRI). |
| **MaxPooling1D/3D** | Pooling cho dữ liệu 1D hoặc 3D. | Giảm kích thước trong CNN 1D/3D. |
| **AveragePooling1D/3D** | Pooling trung bình cho dữ liệu 1D hoặc 3D. | Tương tự MaxPooling nhưng dùng trung bình. |
| **Attention** | Tập trung vào các phần quan trọng của đầu vào. | Transformer, mô hình ngôn ngữ kết hợp CNN. |
| **MultiHeadAttention** | Học từ nhiều không gian biểu diễn khác nhau. | Vision Transformer (ViT). |
| **LSTM/GRU** | Xử lý dữ liệu chuỗi kết hợp với CNN (video) | Phân tích video, mô tả ảnh. |
| **TimeDistributed** | Áp dụng layer lên từng lát thời gian của đầu vào. | Kết hợp CNN và RNN cho video. |
| **Bidirectional** | Xử lý chuỗi theo cả hai hướng (forward và backward). | Cải thiện hiệu suất trong RNN cho chuỗi. |
| **RepeatVector** | Lặp lại vector đầu vào một số lần. | Kiến trúc encoder-decoder trong sequence-to-sequence. |
| **Embedding** | Chuyển đổi index thành vector dense. | Xử lý ngôn ngữ tự nhiên, có thể kết hợp với CNN. |
| **LocallyConnected2D** | Tương tự Conv2D nhưng không chia sẻ trọng số. | Khi các vùng khác nhau cần trọng số riêng. |
| **GroupConv2D** | Chia kênh đầu vào thành nhóm và áp dụng Conv2D riêng. | Giảm tham số trong các mạng lớn. |
| **DilatedConv2D** | Tích chập với dilation rate >1, mở rộng receptive field. | Semantic segmentation, WaveNet. |
| **CoordConv2D** | Thêm thông tin tọa độ vào đầu vào. | Cải thiện trong các bài toán liên quan đến vị trí. |
| **Unsampling2D** | Tăng kích thước không gian (dùng trong mạng Generative hoặc Segmentation) | Mô hình U-Net. |

### Thứ tự sắp xếp các layers phổ biến

#### Khối cơ bản

Conv2D – BatchNorm – ReLU – MaxPooling

Conv2D – BatchNorm – ReLU – MaxPooling

Flatten – Dense – Softmax

**Mục đích:** Trích xuất đặc trưng → Giảm kích thước → Phân loại.

**Ví dụ:** Phân loại ảnh đơn giản.

#### Khối Residual (ResNet)

Conv2D – BatchNorm – ReLU – Conv2D – BatchNorm – Add – ReLU

Conv2D – BatchNorm – ReLU – Conv2D – BatchNorm – Add – ReLU

**Mục đích:** Thêm skip connection (Add) để tránh vanishing gradient.

**Ví dụ:** ResNet34, ResNet50.

#### Kết thúc mạng (Thay thế FC layers)

GlobalAveragePooling – Dense – Softmax

**Mục đích:** Giảm chiều dữ liệu → Phân loại.

**Ví dụ:** ResNet50, MobileNet.

#### Khối Inception (Google Net)

Conv1x1 – BatchNorm – ReLU

Conv3x3 – BatchNorm – ReLU

Conv5x5 – BatchNorm – ReLU

MaxPooling – Conv1x1 – BatchNorm – ReLU

Concatenate

**Mục đích:** Kết hợp đặc trưng đa tỷ lệ.

**Ví dụ:** GoogleNet, InceptionV3.

#### Khối U-Net (Segmentation)

Conv2D – BatchNorm – ReLU – MaxPooling

Conv2D – BatchNorm – ReLU – MaxPooling

Conv2DTranspose – BatchNorm – ReLU – Concatenate

Conv2D – BatchNorm – ReLU

**Mục đích:** Giảm kích thước (encoder) → Phục hồi kích thước (decoder).

**Ví dụ:** U-Net, mạng phân vùng ảnh y tế.

#### Khối Separable Convolution (Mobile Net)

DepthwiseConv2D – BatchNorm – ReLU

PointwiseConv2D (Conv1x1) – BatchNorm – ReLU

**Mục đích:** Giảm tham số, tối ưu cho thiết bị di động.

**Ví dụ:** MobileNetV2, EfficientNet.

#### Khối kết hợp CNN + RNN (Video/Time-Series)

Conv2D – BatchNorm – ReLU – MaxPooling

Reshape – LSTM/GRU – Dense – Softmax

**Mục đích:** Xử lý dữ liệu chuỗi hoặc video.

**Ví dụ:** Phân tích video, mô tả ảnh.

#### Khối với Dropout (Chống overfitting)

Dense – BatchNorm – ReLU – Dropout

Dense – BatchNorm – ReLU – Dropout

Dense – Softmax

**Mục đích:** Giảm overfitting trong các lớp FC.

**Ví dụ:** AlexNet, mạng có dữ liệu nhỏ.

### Các loại regularization

- Regularization (chính quy hóa) là một kỹ thuật trong machine learning nhằm ngăn chặn hiện tượng **overfitting** (quá khớp). Overfitting xảy ra khi mô hình học quá tốt trên dữ liệu huấn luyện nhưng không thể tổng quát hóa hiệu quả trên dữ liệu mới (ví dụ: tập kiểm tra). Để giải quyết vấn đề này, regularization thêm một **hình phạt** vào **hàm mất mát (loss function)**, giúp hạn chế độ phức tạp của mô hình và làm cho nó đơn giản hơn, từ đó cải thiện khả năng tổng quát hóa.

- Các loại regularization phổ biến:

#### L1 Regularization (Lasso)

Thêm tổng giá trị tuyệt đối của các trọng số vào hàm mất mát.

Công thức: loss + λ \* Σ|w|, trong đó λ là hệ số điều chỉnh mức độ phạt.

Hiệu quả: Làm một số trọng số bằng 0, giúp giảm số lượng đặc trưng (feature selection).

#### L2 Regularization (Ridge)

Thêm tổng bình phương của các trọng số vào hàm mất mát.

Công thức: loss + λ \* Σw².

Hiệu quả: Giảm giá trị các trọng số lớn, làm mô hình ít nhạy cảm với các đặc trưng cụ thể, nhưng không loại bỏ hoàn toàn đặc trưng.

#### L1\_L2 Regularization (Elastic Net)

Kết hợp cả L1 và L2.

Công thức: loss + λ1 \* Σ|w| + λ2 \* Σw².

Hữu ích khi có nhiều đặc trưng tương quan với nhau.

#### Dropout

Áp dụng chủ yếu trong mạng nơ-ron (neural networks), dropout ngẫu nhiên "tắt" một tỷ lệ neuron trong quá trình huấn luyện.

Hiệu quả: Ngăn mô hình phụ thuộc quá nhiều vào một số neuron cụ thể, tăng tính tổng quát.

#### Batch Normalization

Chuẩn hóa đầu vào của các tầng để ổn định quá trình huấn luyện.

#### Early Stopping

Dừng quá trình huấn luyện sớm khi hiệu suất trên tập validation không còn cải thiện sau một số epoch nhất định.

Hiệu quả: Ngăn mô hình học quá kỹ trên dữ liệu huấn luyện, giảm overfitting.

#### Data Augmentation

Tăng cường dữ liệu bằng cách tạo thêm các phiên bản biến đổi của dữ liệu huấn luyện (xoay, lật, phóng to, v.v.).

Hiệu quả: Giúp mô hình học được các đặc trưng bất biến, cải thiện khả năng tổng quát hóa.

# TĂNG CƯỜNG DỮ LIỆU

XOAY: 40

DỊCH CHUYỂN CHIỀU NGANG: 20%

DỊCH CHUYỂN CHIỀU DỌC: 20%

XÉN HÌNH ẢNH, HÌNH ẢNH SẼ BỊ NGHIÊNG 1 GÓC: 0.2

PHÓNG TO/THU NHỎ: 20%

LẬT NGANG HÌNH ẢNH (GIỐNG ĐỐI XỨNG QUA GƯƠNG)

ĐIỀN VÀO CÁC PIXEL BỊ THIẾU: PIXEL GẦN NHẤT

# Tập validate là gì?

Tập **validate** (validation set) là một **tập dữ liệu dùng để đánh giá mô hình** trong quá trình huấn luyện (training), **nhưng không dùng để huấn luyện trực tiếp**. Nó giúp theo dõi xem mô hình có **đang học tốt** hay **bắt đầu bị overfitting** hay không.

# Overfitting là như thế nào?

Nếu mô hình học quá tốt trên tập train nhưng sai nhiều ở tập validate → bị overfitting.

# Fine-tune là gì?

"**Fine-tune**" (tinh chỉnh) là một bước **tối ưu hóa thêm** mô hình đã huấn luyện sẵn, nhằm **nâng cao hiệu suất** cho một bài toán cụ thể.

Fine-tune = Huấn luyện tiếp một mô hình đã học trước đó, để **phù hợp hơn với dữ liệu mới**.

Fine-tune thường dùng trong Transfer Learning (Học chuyển giao):

**Không cần huấn luyện từ đầu** (tốn thời gian, dữ liệu).

Thay vào đó, **sử dụng mô hình có sẵn** (ví dụ: ResNet, BERT, GPT...) và fine-tune nó với dữ liệu mới.

**Các bước fine-tune cơ bản:**

1. **Nạp mô hình pre-trained** (đã học trước).
2. **Giữ lại phần quan trọng** (ví dụ: lớp trích đặc trưng).
3. **Thay lớp output** theo số lớp bài toán của bạn.
4. **Huấn luyện tiếp (fine-tune)** với tập dữ liệu mới.

**Vì sao nên fine-tune?**

* Tiết kiệm thời gian
* Cần ít dữ liệu hơn
* Kết quả tốt hơn vì tận dụng kiến thức từ mô hình lớn

# Loss là gì?

Dùng để **đo lường mức độ sai lệch giữa dự đoán của mô hình và giá trị thực tế**.

* Nếu **dự đoán càng gần đúng**, loss **càng nhỏ**.
* Nếu **dự đoán sai nhiều**, loss **càng lớn**.
* Mục tiêu của mô hình là **giảm loss thấp nhất có thể** trong quá trình huấn luyện.

# Confusion matrix là gì?

Mô hình dự đoán đúng bao nhiêu mẫu ở mỗi lớp.

Mô hình **dự đoán sai bao nhiêu mẫu**, và **sai ở đâu** (nhầm lớp nào sang lớp nào).

Từ confusion matrix, ta có thể tính được: Accuracy, Precision, Recall, F1-Score. Những chỉ số này **đặc biệt quan trọng** khi dữ liệu **mất cân bằng (imbalanced)**, vì accuracy không còn là thước đo tốt.

Từ đó, bạn có thể cải thiện mô hình: **bổ sung dữ liệu huấn luyện**, **tăng số đặc trưng phân biệt**, v.v.

# Xử lý mất cân bằng dữ liệu?

Áp dụng kỹ thuật class weighting với công thức

* + Trong đó:
    - : trọng số của lớp j
    - : tổng số mẫu trong tập dữ liệu
    - : tổng số lớp
    - : số mẫu của lớp j

Nhờ kỹ thuật này, mô hình sẽ phạt nặng hơn khi dự đoán sai các lớp có ít mẫu, buộc mô hình phải học cách nhận diện tốt hơn các lớp này. Từ đó giảm thiểu hiện tượng “lớp phổ biến hơn sẽ dễ được dự đoán hơn”.

# Biểu đồ ROC Curve cho multiple class

Giúp đánh giá khả năng phân biệt của mô hình với từng lớp. Đường ROC càng gần góc trái trên, mô hình phân biệt càng tốt lớp đó. Nếu ROC nằm gần đường chéo → mô hình phân biệt lớp đó **kém hiệu quả**.

**AUC = diện tích dưới đường ROC**, thể hiện **mức độ phân biệt** giữa các lớp. AUC càng gần 1 càng tốt.

Nếu huấn luyện nhiều mô hình, có thể so sánh **AUC trung bình (macro/micro)** của từng mô hình để chọn cái tốt nhất.

AUC trung bình ResNet-50: 0.944

AUC trung bình MobileNet-V2: 0.973

AUC trung bình EfficientNet-B0: 0.980

# Biểu đồ Precision/Recall cho từng lớp

Thể hiện mối quan hệ giữa **Precision** (Độ chính xác): tỷ lệ dự đoán đúng trong tất cả những gì mô hình dự đoán là **positive** và **Recall** (Độ nhạy): tỷ lệ dự đoán đúng trong tất cả các **positive thật sự**.

Đặc biệt hữu ích khi dữ liệu mất cân bằng, cho thấy sự đánh đổi (trade-off) giữa precision và recall, giúp chọn **ngưỡng phân loại (threshold)** phù hợp.

# Phân loại

- thành 4 nhóm: tái chế (nhựa, giấy, bìa carton, kim loại, thuỷ tinh), hữu cơ thực phẩm hữu cơ (rau củ, vỏ trái cây, …), không tái chế/nguy hại (pin), tái sử dụng (quần áo, giày dép)

- để đạt mục tiêu, có những phương pháp

* **Phương pháp 1: Phân loại trực tiếp thành 4 nhóm**
  + Mô hình được huấn luyện để trực tiếp phân loại hình ảnh vào một trong bốn nhóm chức năng.
  + Ưu điểm: Đơn giản, hiệu quả về mặt tính toán, ít phức tạp trong kiến trúc mô hình.
  + Nhược điểm: Không tận dụng được thông tin chi tiết về loại rác cụ thể, có thể dẫn đến giảm độ chính xác khi các nhóm có đặc tính thị giác đa dạng.
* **Phương pháp 2: Phân loại thành 9 lớp, sau đó ánh xạ vào 4 nhóm**
  + Mô hình sẽ phân loại thành 9 lớp chi tiết sau đó ánh xạ kết quả đầu ra vào 4 nhóm chức năng.
  + Ánh xạ được thực hiện thông qua cấu trúc dữ liệu ánh xạ hoặc các câu lệnh điều kiện rẽ nhánh.
  + Ưu điểm: Tận dụng được thông tin chi tiết về đặc tính của từng loại rác, dễ mở rộng và điều chỉnh quy tắc ánh xạ.
  + Nhược điểm: Yêu cầu hai bước xử lý (phân loại và ánh xạ).
* **Phương pháp 3: Sử dụng mô hìnnh multi-output**
  + Mô hình vừa phân loại 9 lớp, vừa phân loại thành 4 nhóm chức năng.
  + Ưu điểm: Tận dụng được thông tin đa cấp độ, cung cấp kết quả đồng thời ở hai mức độ chi tiết.

Nhược điểm: Kiến trúc mô hình phức tạp hơn, khó tối ưu hóa, tăng khả năng overfitting.

# Chuẩn hoá màu sắc

**ResNet-50:** ResNet-50 sử dụng hàm tiền xử lý đặc thù thay vì phép rescale thông thường. Cụ thể, dữ liệu đầu vào được chuyển từ không gian màu RGB sang BGR, sau đó trừ đi giá trị trung bình của ImageNet. Phương pháp này đảm bảo sự đồng bộ với cách huấn luyện ban đầu của ResNet-50 trên ImageNet, tối ưu hóa hiệu suất khi áp dụng transfer learning cho bài toán phân loại rác thải.

**MobileNet-V2:** Hình ảnh được chuyển từ không gian màu RGB sang BGR, sau đó mỗi kênh màu được trừ đi giá trị trung bình của ImageNet (103.939, 116.779, 123.68 cho các kênh B, G, R tương ứng). Phương pháp này thay thế cho phép rescale đơn giản (ví dụ, chia cho 255), giúp mô hình tận dụng tối đa các trọng số đã được huấn luyện trước.

**EfficientNet-B0:** Giá trị pixel của hình ảnh được chuẩn hóa từ dải [0, 255] sang khoảng [-1, 1] theo công thức . Phương pháp này thay thế phép rescale đơn giản, giúp điều chỉnh dữ liệu phù hợp ới phân phối mà EfficientNet-B0 đã được huấn luyện, cải thiện tốc độ hội tụ và độ chính xác.