# 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.