

# Mục lục

|          |   |           |
|----------|---|-----------|
| <b>1</b> | <b>Đặt vấn đề</b>                           | <b>2</b>  |
| <b>2</b> | <b>Mô tả dữ liệu</b>                        | <b>3</b>  |
| 2.1      | Tổng quan về dữ liệu . . . . .              | 3         |
| 2.2      | Trực quan dữ liệu . . . . .                 | 4         |
| <b>3</b> | <b>Tiền xử lý dữ liệu</b>                   | <b>6</b>  |
| 3.1      | Chuẩn hóa kích thước và định dạng . . . . . | 6         |
| 3.2      | Phân đoạn hình ảnh . . . . .                | 6         |
| <b>4</b> | <b>Cơ sở lý thuyết</b>                      | <b>9</b>  |
| 4.1      | K-Means . . . . .                           | 9         |
| 4.2      | Softmax Regression . . . . .                | 12        |
| 4.3      | Convolutional Neural Network . . . . .      | 16        |
| 4.4      | Phương pháp Transfer Learning . . . . .     | 22        |
| <b>5</b> | <b>Xây dựng mô hình</b>                     | <b>23</b> |
| 5.1      | Phân cụm dữ liệu . . . . .                  | 23        |
| 5.2      | Phân loại dữ liệu . . . . .                 | 25        |
| 5.2.1    | Softmax Regression . . . . .                | 25        |
| 5.2.2    | Convolutional Neural Network . . . . .      | 26        |
| <b>6</b> | <b>Kết quả và thảo luận</b>                 | <b>28</b> |
| 6.1      | Phân cụm dữ liệu . . . . .                  | 28        |
| 6.2      | Phân loại dữ liệu . . . . .                 | 30        |

# Chương 1

## Đặt vấn đề

Trong lĩnh vực nghiên cứu và quản lý nguồn lợi thủy sản, việc phân loại và phân cụm cá đóng vai trò quan trọng để đảm bảo việc quản lý và khai thác bền vững các nguồn lợi thủy sản. Tuy nhiên, việc phân loại và phân cụm cá thủ công dựa trên các đặc trưng truyền thống như kích thước, hình dạng và màu sắc đòi hỏi nhiều công sức và mất thời gian. Đồng thời, sự chính xác và hiệu quả của phương pháp truyền thống này còn hạn chế do sự phụ thuộc vào kinh nghiệm và kiến thức của nhân viên.

Với sự phát triển nhanh chóng của công nghệ Học máy, việc ứng dụng các thuật toán và mô hình Học máy trong bài toán phân loại và phân cụm cá đang trở thành một phương pháp tiềm năng để cải thiện hiệu quả và độ chính xác của quá trình này. Học máy có khả năng tự động học và tìm ra các mẫu và quy luật từ dữ liệu huấn luyện, từ đó xây dựng các mô hình dự đoán và phân loại cá dựa trên các đặc trưng tự động được học từ dữ liệu thực tế.

Nhận thức được điều này, nhóm chúng tôi thực hiện xây dựng và khảo sát một số mô hình học máy trên bộ dữ liệu 9 loại hải sản khác nhau, được thu nhập từ một siêu thị ở Izmir, Thổ Nhĩ Kỳ.

## Chương 2

# Mô tả dữ liệu

### 2.1 Tổng quan về dữ liệu

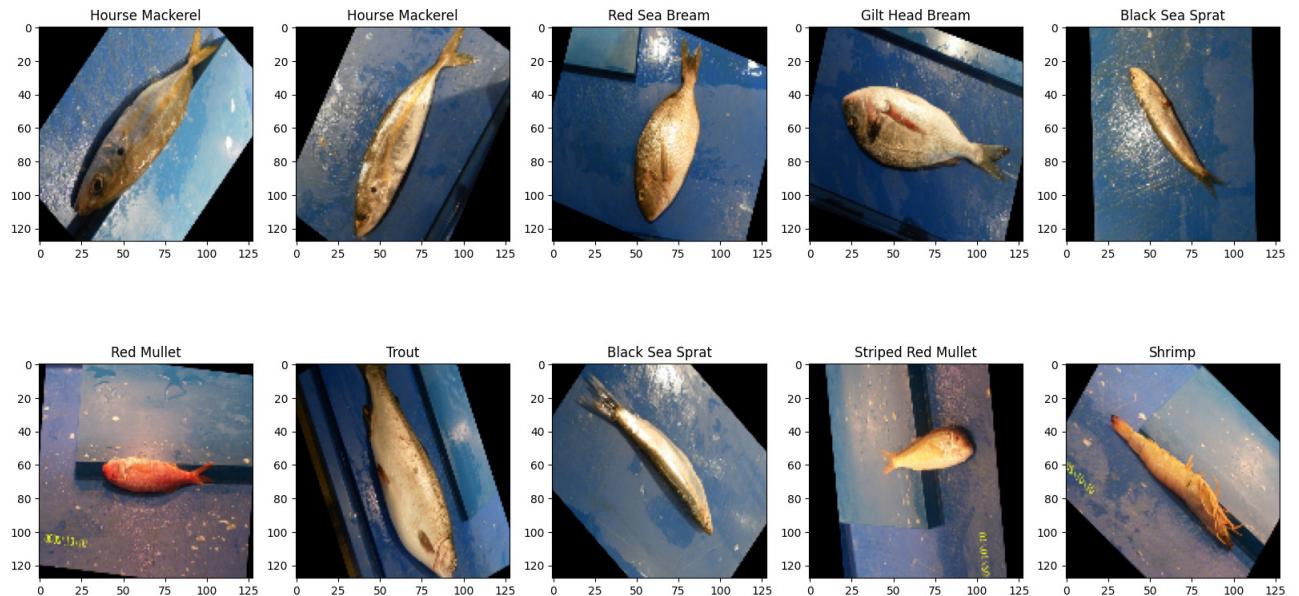
Bộ dữ liệu "A Large Scale Fish Dataset" là một tập dữ liệu lớn (3GB) chứa 9 loại hải sản khác nhau được thu nhập từ một siêu thị Izmir, Thổ Nhĩ Kỳ và được xuất bản trong ASYU 2020 (tác giả: O. Ulucan, D. Karakaya, M. Turkan). Bộ dữ liệu này được sử dụng để phân loại và phân đoạn hải sản. Những hình ảnh có trong bộ dữ liệu bao gồm 9 loại cá khác nhau, bao gồm: Gilt head bream (cá tráp đầu vàng), Red sea bream (cá tráp đỏ), Sea bass (cá vược), Red mullet (cá đồi đỏ), Horse mackerel (cá thu ngựa), Black sea sprat (cá trích biển đen), Striped red mullet (cá đồi đỏ sọc), Trout (cá hồi), Shrimp (tôm). Bộ dữ liệu gồm 2 thư mục sử dụng để huấn luyện và kiểm tra mô hình:

**Fish\_Dataset:** Thư mục bao gồm 9 thư mục con ứng với 9 loại cá, với mỗi thư mục con sẽ chứa hai thư mục dùng để lưu trữ hai loại ảnh RGB và Ground Truth. Thư mục này bao gồm các ảnh có kích thước 590x445 pixel và đã được tăng cường bằng cách lật, xoay ảnh với mục đích tăng cường độ đa dạng và chính xác của dữ liệu. Sau khi hoàn thành việc tăng cường, với mỗi loại cá bao gồm 2000 hình ảnh cho mỗi lớp, bao gồm 1000 ảnh RGB và 1000 ảnh Ground Truth. Tất cả các hình ảnh cho mỗi loại được đánh số từ "00001.png" đến "01000.png".

**NA\_Fish\_Dataset\_Raw:** Thư mục gồm 9 thư mục con chứa các ảnh RGB với định dạng png hoặc jpg. Tập dữ liệu bao gồm 30 ảnh loại Trout và 50 ảnh các loại cá còn lại với kích thước 2128x2832 pixel hoặc 768x1024 pixel.

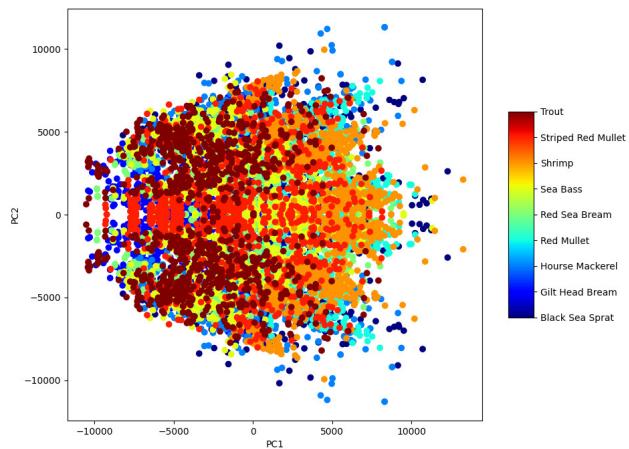
## 2.2 Trực quan dữ liệu

Trực quan dữ liệu ảnh gốc:



Hình 2.1: Hình ảnh một số loại cá sau khi đọc và thu nhỏ kích thước

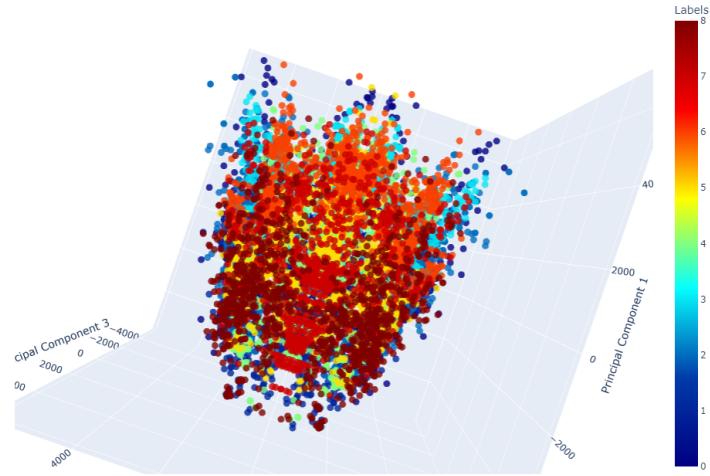
Trực quan dữ liệu dưới dạng 2D:



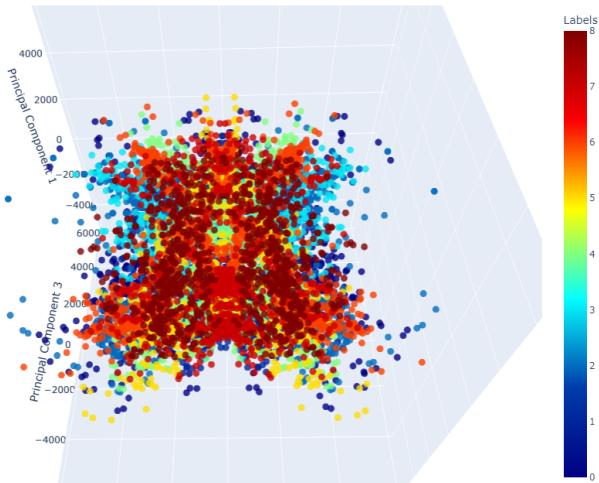
Hình 2.2: Dữ liệu sau khi sử dụng PCA giảm về 2 chiều

Trực quan dữ liệu dưới dạng 3D:

PCA Visualization in 3D



PCA Visualization in 3D



Hình 2.3: Dữ liệu sau khi sử dụng PCA giảm về 3 chiều dưới 2 góc độ khác nhau

## Chương 3

# Tiền xử lý dữ liệu

### 3.1 Chuẩn hóa kích thước và định dạng

Loại dữ liệu là hình ảnh, sau khi thực hiện đọc, dữ liệu được biểu diễn dưới dạng ma trận pixel.

Bước đầu tiên, do kích thước, định dạng cũng như tên của các file ảnh trong thư mục NA\_Fish\_Dataset chưa được đồng nhất, nhóm sử dụng thư viện opencv, thực hiện đưa chúng về cùng kích thước 590x445 pixel và đổi tên và định dạng thành: "0001.png" - "0050.png" và lưu lại chúng vào thư mục có sẵn.

Sau khi dữ liệu đã được đồng nhất với nhau, thực hiện đọc dữ liệu và thu nhỏ kích thước về 128x128 pixel cho tất cả các ảnh của tập huấn luyện (ảnh RGB trong Fish\_Dataset) và kiểm tra (ảnh RGB trong NA\_Fish\_Dataset), sử dụng tên thư mục làm nhãn của dữ liệu rồi tiến hành đưa nhãn đó về dạng số (0 - 8).

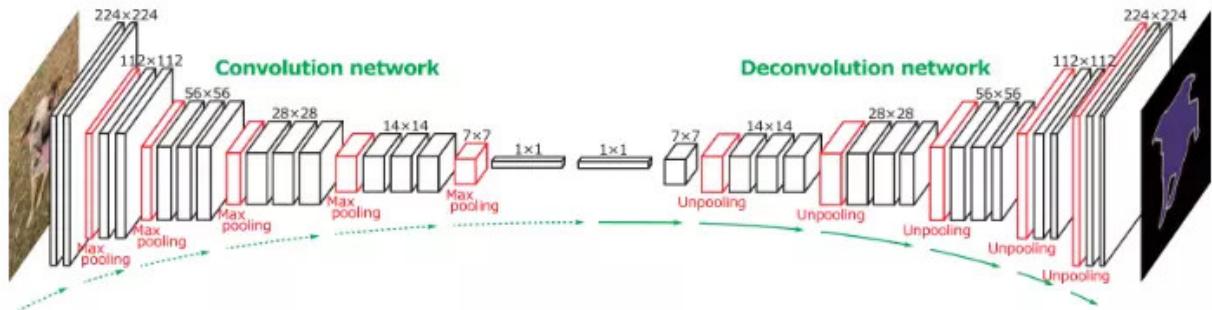
Bước tiếp theo, thực hiện chuẩn hóa các ảnh, đưa các giá trị trong ma trận ảnh về đoạn [0,1] bằng cách chia chúng cho 255.0

Cuối cùng, thực hiện chia dữ liệu huấn luyện thành hai phần: train (7200 ảnh) và validation (1800 ảnh)

### 3.2 Phân đoạn hình ảnh

Phân đoạn hình ảnh đóng vai trò quan trọng trong xử lý hình ảnh. Nó là quá trình chia nhỏ một bức ảnh thành nhiều phần giúp người dùng tách riêng các đối tượng hoặc vùng trong ảnh, đồng thời loại bỏ nhiễu trong ảnh.

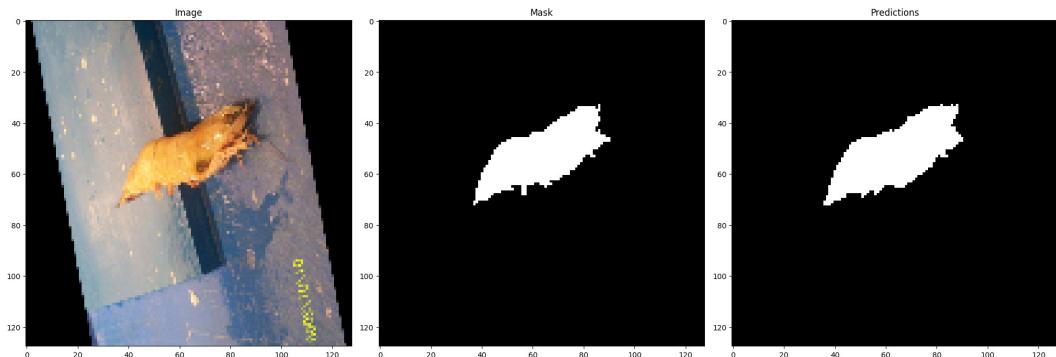
Nhóm sử dụng xây dựng mạng phân đoạn hình ảnh dựa trên cấu trúc mạng Unet:



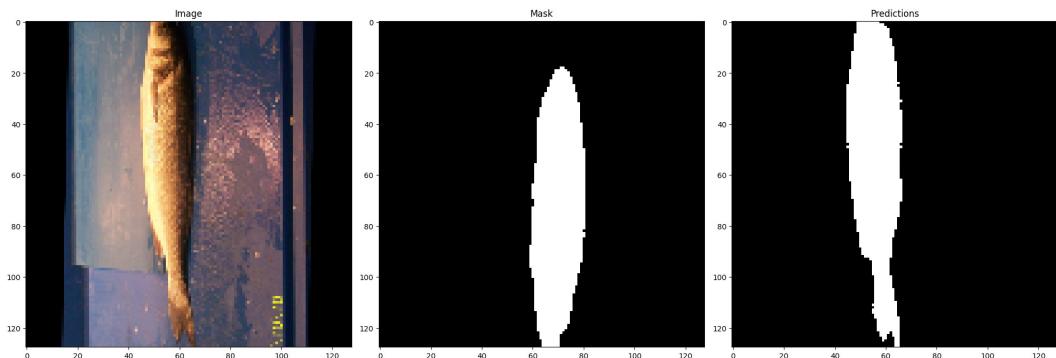
Hình 3.1: Cấu trúc mạng

Với 2 thành phần encoder (convolution network) và decoder (deconvolution network). Trong đó phần encoder dùng để giảm chiều dài và chiều rộng của ảnh bằng việc sử dụng các lớp convolutions và các lớp poolings. Trong đó phần decoder dùng để phục hồi lại kích thước ban đầu của ảnh.

Một số hình ảnh huấn luyện từ mạng sau khi huấn luyện theo thứ tự: ảnh gốc, ảnh GT có sẵn, ảnh GT mô hình nhóm xây dựng:



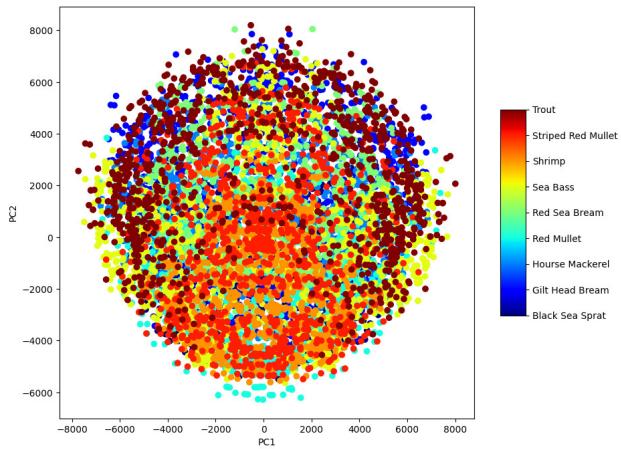
Hình 3.2: Phân đoạn hình ảnh tôm



Hình 3.3: Phân đoạn hình ảnh cá See Bass

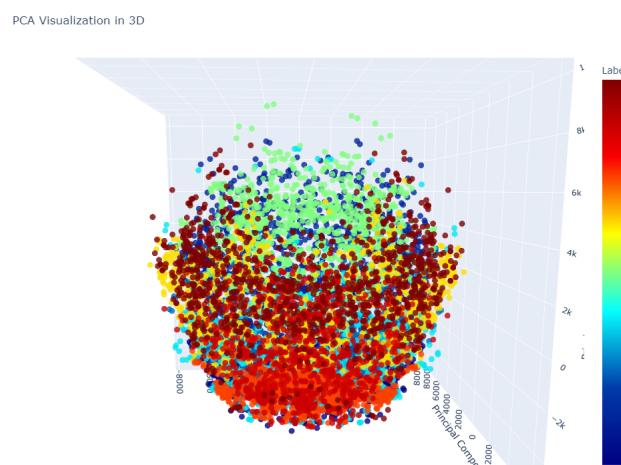
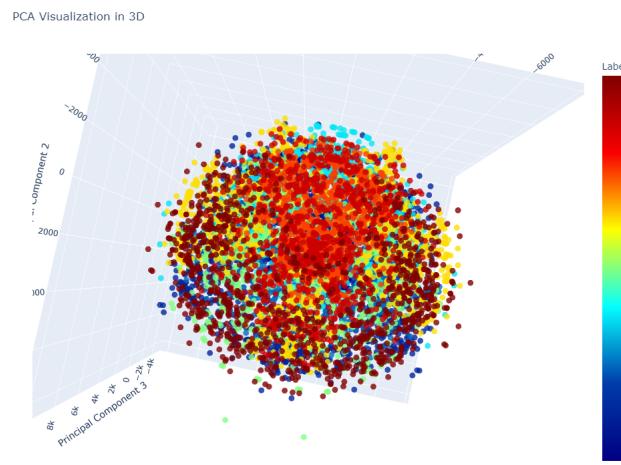
Một số hình ảnh trực quan dữ liệu sau khi thực hiện phân đoạn:

- Trực quan dữ liệu dưới dạng 2D:



Hình 3.4: Dữ liệu sau khi sử dụng PCA giảm về 2 chiều

- Trực quan dữ liệu dưới dạng 3D:



Hình 3.5: Dữ liệu sau khi sử dụng PCA giảm về 3 chiều dưới 2 góc độ khác nhau

## Chương 4

# Cơ sở lý thuyết

### 4.1 K-Means

Thuật toán K-means là một thuật toán phân cụm dữ liệu phổ biến trong lĩnh vực học không giám sát. Trong thuật toán này, chúng ta không biết nhãn của từng điểm dữ liệu. Mục đích là làm thế nào để phân dữ liệu thành các cụm khác nhau sao cho dữ liệu trong cùng một cụm có tính chất giống nhau.

#### PHÂN TÍCH TOÁN HỌC

**Phát biểu bài toán dưới ký hiệu toán học:** Cho tập dữ liệu  $X = \{x_n\}_{n=1}^N$  và số cụm cần phân chia  $0 < k < N$ . Chúng ta cần tìm các điểm trung tâm  $m_1, m_2, \dots, m_k \in R^{d \times 1}$  và nhãn của mỗi điểm dữ liệu.

Với mỗi điểm dữ liệu  $x_i$  đặt  $y_i = [y_{i1}, y_{i2}, \dots, y_{iK}]$  là one-hot vector đại diện cho nhãn của nó, trong đó nếu  $x_i$  được phân vào cụm  $k$  thì  $y_{ik} = 1$  và  $y_{ij} = 0, \forall j \neq k$ .  $Y$  là ma trận chứa vector nhãn của mỗi điểm dữ liệu  $Y = [y_1, y_2, \dots, y_N]$ .

Ràng buộc  $y_i$  có thể được viết dưới dạng toán học như sau:

$$y_{ik} \in \{0, 1\}, \quad \sum_{k=1}^K y_{ik} = 1 \quad (1)$$

#### Hàm mất mát và bài toán tối ưu:

Giả sử ước lượng một số các điểm được phân cụm vào cụm  $k$  có tâm là  $m_k$ , thì mỗi điểm dữ liệu  $x_i$  có khoảng cách tới tâm là  $(x_i - m_k)$ . Và chúng ta phải thực hiện tính sao cho khoảng cách đó có trị tuyệt đối nhỏ nhất:

$$\|x_i - m_k\|_2^2$$

Dựa vào ràng buộc (1), có thể viết lại biểu thức trên thành:

$$y_{ik} \|\mathbf{x}_i - \mathbf{m}_k\|_2^2 = \sum_{j=1}^K y_{ij} \|\mathbf{x}_i - \mathbf{m}_j\|_2^2$$

Vậy với bộ dữ liệu có N điểm, ta có tổng khoảng cách trên toàn bộ dữ liệu là:

$$\mathcal{L}(\mathbf{Y}, \mathbf{M}) = \sum_{i=1}^N \sum_{j=1}^K y_{ij} \|\mathbf{x}_i - \mathbf{m}_j\|_2^2$$

Hàm số mất mát trong bài toán K-means clustering của chúng ta là hàm  $\mathcal{L}(\mathbf{Y}, \mathbf{M})$  với ràng buộc như được nêu trong phương trình (1).

Thuật toán phân cụm cần tối ưu bài toán:

$$\mathbf{Y}, \mathbf{M} = \arg \min_{\mathbf{Y}, \mathbf{M}} \sum_{i=1}^N \sum_{j=1}^K y_{ij} \|\mathbf{x}_i - \mathbf{m}_j\|_2^2 \quad (2)$$

thỏa mãn  $y_{ij} \in \{0, 1\} \quad \forall i, j; \quad \sum_{j=1}^K y_{ij} = 1 \quad \forall i$

### **Thuật toán tối ưu hàm mất mát:**

Bài toán (2) là một bài toán thuộc loại mix-integer programming được đánh giá là khó tìm nghiệm tối ưu toàn cục. Tuy nhiên trong một số trường hợp vẫn có thể tìm được phương pháp để tìm nghiệm gần đúng.

Cách giải bài toán (2) là giải xen kẽ Y và M khi biến còn lại được cố định. Đây là một thuật toán lặp, cũng là kỹ thuật phổ biến khi giải bài toán tối ưu.

**1. Cố định M, tìm Y:** giả sử đã tìm được các tâm cụm, tìm các nhãn vector để hàm mất mát đạt giá trị nhỏ nhất.

Khi đó, bài toán trở thành:

$$\mathbf{y}_i = \arg \min_{\mathbf{y}_i} \sum_{j=1}^K y_{ij} \|\mathbf{x}_i - \mathbf{m}_j\|_2^2 \quad (3)$$

thỏa mãn subject to:  $y_{ij} \in \{0, 1\} \quad \forall j; \quad \sum_{j=1}^K y_{ij} = 1$

Do  $y_i$  chỉ có một giá trị bằng 1, nên (3) có thể viết lại:

$$j = \arg \min_j \|\mathbf{x}_i - \mathbf{m}_j\|_2^2$$

Đây chính là bình phương khoảng cách từ điểm  $x_i$  tới tâm cụm  $m_j$ . Có thể kết luận rằng mỗi điểm  $x_i$  thuộc vào các cụm có khoảng cách từ nó tới tâm cụm đó là nhỏ nhất.

**2. Cố định Y, tìm M:** giả sử đã phân cụm cho từng điểm dữ liệu, tìm tâm cụm mới sao cho hàm mất mát đạt giá trị nhỏ nhất.

Bài toán có thể được viết lại thành:

$$\mathbf{m}_j = \arg \min_{\mathbf{m}_j} \sum_{i=1}^N y_{ij} \|\mathbf{x}_i - \mathbf{m}_j\|_2^2.$$

Đặt  $l(\mathbf{m}_j)$  là hàm bên trong dấu argmin, thực hiện đạo hàm hai vế, ta có:

$$\frac{\partial l(\mathbf{m}_j)}{\partial \mathbf{m}_j} = 2 \sum_{i=1}^N y_{ij} (\mathbf{m}_j - \mathbf{x}_i)$$

Giải phương trình đạo hàm :

$$\begin{aligned} \sum_{i=1}^N y_{ij} (\mathbf{m}_j - \mathbf{x}_i) &= 0 \\ \mathbf{m}_j \sum_{i=1}^N y_{ij} &= \sum_{i=1}^N y_{ij} \mathbf{x}_i \\ \Rightarrow \mathbf{m}_j &= \frac{\sum_{i=1}^N y_{ij} \mathbf{x}_i}{\sum_{i=1}^N y_{ij}} = \frac{\sum_{i=1}^N \mathbf{x}_i}{N} \end{aligned}$$

$m_j$  chính là trung bình cộng của các điểm trong cụm thứ j. Đây cũng chính là lý do cho tên K-means.

### TÓM TẮT THUẬT TOÁN:

Đầu vào: Dữ liệu X và số lượng cluster cần tìm K.

Đầu ra: Các center M và label vector cho từng điểm dữ liệu Y.

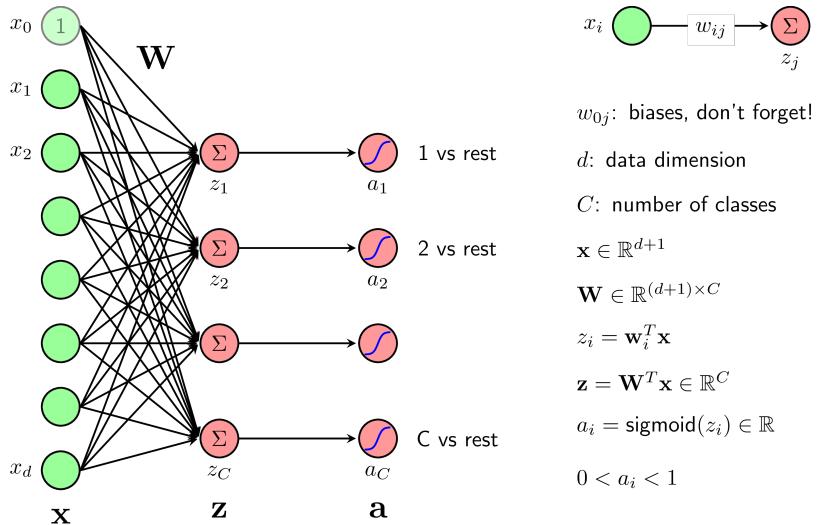
1. Chọn K điểm bất kỳ làm các center ban đầu.
2. Phân mỗi điểm dữ liệu vào cluster có center gần nó nhất.
3. Nếu việc gán dữ liệu vào từng cluster ở bước 2 không thay đổi so với vòng lặp trước nó thì ta dừng thuật toán.
4. Cập nhật center cho từng cluster bằng cách lấy trung bình cộng của tất cả các điểm dữ liệu đã được gán vào cluster đó sau bước 2.
5. Quay lại bước 2.

## 4.2 Softmax Regression

Mô hình Multinomial Logistic Regrssion (hay còn gọi là Softmax Regression) là một mô hình học máy dùng để dự đoán xác suất của các lớp dữ liệu đầu ra dựa trên đầu vào, được sử dụng phổ biến trong các bài toán phân loại như phân loại ảnh, văn bản,... . Mô hình sử dụng hàm softmax để tính toán các xác suất dữ liệu thuộc từng lớp dựa trên các đặc trưng đầu vào, chúng chuyển đổi vector số thực thành vector xác suất, trong đó mỗi giá trị đại diện cho xác suất của một lớp, tổng các giá trị này bằng 1.

### PHÂN TÍCH TOÁN HỌC

**Mô hình One-vs-Rest:** One-vs-Rest còn được gọi là one-hot coding, ove-vs-all, one-against-rest... là một kỹ thuật trong Machine Learning sử dụng để phân loại đa lớp (multi-class classification). Ta xây dựng một mô hình phân loại riêng biệt cho mỗi lớp, bao gồm một lớp và tất cả các lớp còn lại.



Hình 4.1: Phân loại đa lớp với hồi quy logistic và one-vs-rest

Dữ liệu  $\mathbf{X}$  có  $(d+1)$  chiều vì thêm phần tử 1 được thêm vào phía trước, thể hiện hệ số tự do trong hàm tuyến tính.

Tầng đầu ra có thể tách thành hai tầng con  $\mathbf{z}$  và  $\mathbf{a}$ . Mỗi thành phần của tầng con thứ hai chỉ phụ thuộc vào thành phần tương ứng ở tầng con thứ nhất  $z_i$  thông qua hàm sigmoid:  $a_i = \sigma(z_i)$ . Các giá trị đầu ra  $a_i$  đều là các số dương nhưng vì không có ràng buộc giữa chúng, tổng các xác suất này không đảm bảo bằng một.

Các mô hình hồi quy tuyến tính, PLA, và hồi quy logistic chỉ có một nút ở tầng đầu ra. Trong các trường hợp đó, tham số mô hình chỉ là một vector  $w$ . Trong trường hợp tầng đầu ra có nhiều hơn một nút, tham số mô hình sẽ là tập hợp tham số  $w_i$  ứng với từng nút. Lúc này, ta có một ma trận trọng số  $W = [w_1, w_2, \dots, w_C]$ ,

mỗi cột ứng với một nút ở tầng đầu ra

**Hàm Softmax:** Chúng ta cần một mô hình xác suất để tính xác suất để mỗi đầu vào  $x$  rơi vào lớp thứ  $i$ , được biểu diễn bởi  $a_i$ . Để đảm bảo tính chính xác của mô hình, điều kiện cần là các  $a_i$  phải là giá trị dương và tổng của chúng phải bằng một. Ngoài ra, để đạt được mục tiêu này, ta cần thêm điều kiện rằng giá trị  $z_i = x^T w_i$  càng lớn thì xác suất rơi vào lớp thứ  $i$  càng cao. Điều kiện cuối cùng này cho thấy ta cần một mối quan hệ đồng biến giữa  $z_i$  và  $a_i$ .

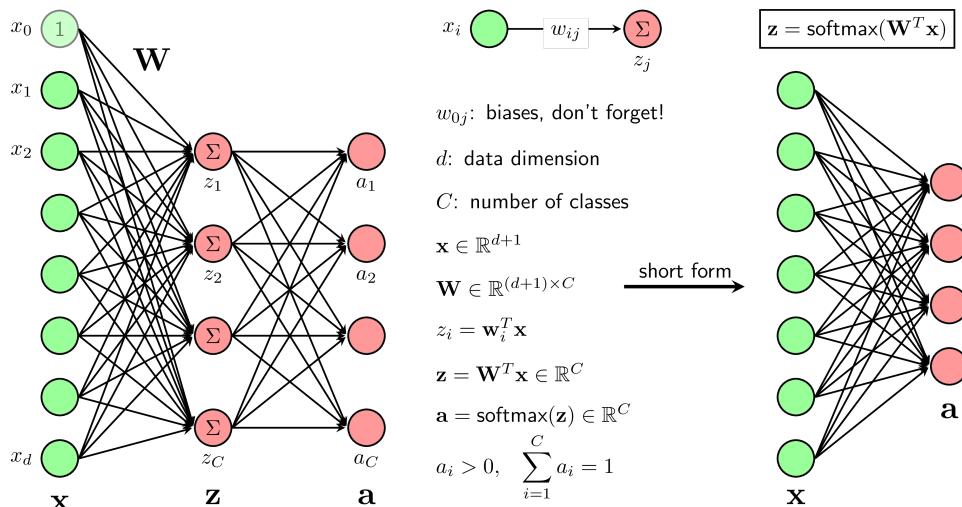
Hàm softmax:  $g(z) : R \rightarrow (0, 1)$  xây dựng trên tập  $z_1, \dots, z_C$ , xác định như sau:

$$a_i = g(z_i) = \frac{\exp(z_i)}{\sum_{j=1}^C \exp(z_j)}$$

Do đây là hàm mũ nên  $a_i$  luôn dương và có tổng bằng 1. Lúc này có thể xem:

$$P(y_k = i|x_k; W) = a_i$$

Trong đó,  $P(y = i|x, W)$  là xác suất để một điểm dữ liệu  $x$  rơi vào lớp thứ  $i$  nếu biết tham số mô hình là ma trận trọng số  $W$ . Nó khác mô hình one-vs-rest ở chỗ có các liên kết giữa mọi nút của 2 tầng  $z$  và  $a$ .



Hình 4.2: Phân loại đa lớp với hồi quy logistic và one-vs-rest

**Softmax-stable:** Khi một trong các  $z_i$  quá lớn, dẫn đến việc tính toán  $\exp(z_i)$  có thể gây ra hiện tượng tràn số ảnh hưởng đến việc tính toán hàm softmax. Có thể khắc phục nó bằng:

$$\frac{\exp(z_i)}{\sum_{j=1}^C \exp(z_j)} = \frac{\exp(-c)\exp(z_i)}{\exp(-c) \sum_{j=1}^C \exp(z_j)} = \frac{\exp(z_i - c)}{\sum_{j=1}^C \exp(z_j - c)}$$

Với  $c$  là hằng số bất kỳ (đó là 1 giá trị đủ lớn). Trong nghiên cứu thực nghiệm họ thường chọn  $c = \max z_i$

### **Hàm măt măt và bài toán tối ưu:**

Có 2 cách để đi tới hàm mục tiêu :

- Sử dụng ước lượng hợp lý cực đại - hàm log-likelihood
- Sử dụng so sánh phân phối xác suất với vector one-hot-coding qua Entropy chéo (Cross Entropy).

Ở đây nhóm chúng tôi quyết định giải bài toán hàm mục tiêu bằng cách sử dụng ước lượng hợp lý cực đại:

Từ giả thiết xác suất xảy ra của dữ liệu là độc lập, trung bình log-Likelihood của tập dữ liệu huấn luyện là:

$$\begin{aligned} L(w) &= \frac{1}{N} \sum_{n=1}^N \log P(y^{(n)}|x^{(n)}; w) = \frac{1}{N} \sum_{n=1}^N \log \left( \frac{\exp(w_{yn}^T X_n)}{\sum_{c=1}^C \exp(w_c^T x_n)} \right) \\ &= \frac{1}{N} \sum_{n=1}^N (w_{yn}^T) - \log \left[ \sum_{c=1}^C \exp(w_c^T x_n) \right] \end{aligned}$$

Theo phương pháp (Maximum Likelihood Estimation), chúng ta sẽ tìm ra tham số tối ưu  $w_c \in R^d$  ( $c = 1, \dots, C$ ) thông qua cực đại hóa hàm  $L(w)$

Trường hợp sử dụng hiệu chỉnh  $L_2$ , hàm tốn thất :

$$J(w) = -L(w) + \frac{\lambda}{2} \sum_{c=1}^C \|w_c\|_2^2 = -L(w) + \frac{\lambda}{2} \sum_{c=1}^C \sum_{j=1}^d w_{cj}^2 \quad (.)$$

### **Thuật toán tối ưu hàm măt măt:**

Để tối ưu hàm LOSS, trước hết ta nên tối ưu hàm log-Likelihood bằng cách dùng thuật toán Gradient Descent.

- **Bước 1:** Với  $k = 1, \dots, C$  ứng với các phân lớp,  $j = 1, \dots, d$  ứng với các chiều

dữ liệu, ta có đạo hàm đối với 1 cặp dữ liệu duy nhất  $(x_n, y_n)$  cho hàm log-Likelihood:

$$\begin{aligned}\frac{\partial L_n(w)}{\partial w_{kj}} &= \delta(y_n = k)(x_n)_j - \frac{1}{N} \sum_{n=1}^N P(y_n = k|x; w)(x_n)_j \\ &= \frac{1}{N} \sum_{n=1}^N \delta(y_n = k) - g(w_k^T x_n)(x_n)_j \quad (*)\end{aligned}$$

- **Bước 2:** Tính đạo hàm cho toàn bộ dữ liệu

Với mỗi nhãn  $y_n = c \in R^c$  nếu  $y_n = c$  thì thành phần thứ  $c$  của  $\bar{y}^n$  là  $\bar{y}_c^n = 1$  và  $\bar{y}_m^n = 0$  với  $m \neq c$ . Đặt  $a_{kn} := P(y_n = k|x_n; w) = g(w_k^T x_n)$

Từ (\*), ta có:

$$\frac{\partial L(w)}{\partial w_{k,j}} = \frac{1}{N} \sum_{n=1}^N (\delta(y_n = k) - g(w_k^T x_n))(x_n)_j = \frac{1}{N} \sum_{n=1}^N (\bar{y}_k^n - a_{kn})(x_n)_j = \frac{1}{N} \sum_{n=1}^N e_{k,n}(x_n)_j$$

Với  $e_{k,n} := \bar{y}_k^n$ , là sai số giữa phân lớp theo dữ liệu quan sát và xác suất phân lớp theo mô hình.

Công thức đạo hàm dưới dạng ma trận:

$$\frac{\partial L_n(w)}{\partial \mathbf{w}} = \left[ \frac{\partial \mathbf{L}_n(\mathbf{w})}{\partial w_1} \frac{\partial \mathbf{L}_n(\mathbf{w})}{\partial w_2}, \dots, \frac{\partial \mathbf{L}_n(\mathbf{w})}{\partial w_C} \right] = [e_{1,n} \mathbf{x}_n, e_{2,n} \mathbf{x}_n, \dots, e_{C,n} \mathbf{x}_n]$$

$$= \mathbf{x}_n [e_{1,n}, e_{2,n}, \dots, e_{C,n}]^T =: \mathbf{x}_n \mathbf{e}_n^T$$

Vậy dạng ma trận của đạo hàm  $L(w)$  sẽ là:

$$\frac{\partial \mathbf{L}(\mathbf{w})}{\partial \mathbf{w}} = \sum_{n=1}^N \mathbf{x}_n \mathbf{e}_n^T = \mathbf{X} \mathbf{E}^T$$

Với  $E = \bar{Y} - A$  các ma trận đã được định nghĩa  $\bar{Y} = (\bar{y}_k^n); A = (a_{kn})$

- **Bước 3:** Tính đạo hàm cho hàm tổn thất:

Ta có dạng ma trận của đạo hàm hàm tổn thất cho một cặp dữ liệu

$$\frac{\partial J_{(n)}^0(\mathbf{w})}{\partial \mathbf{w}} = -\frac{\partial \mathbf{L}_n(\mathbf{w})}{\partial \mathbf{w}} = -x_n \mathbf{e}_n^T$$

Ma trận của đạo hàm hàm tổn thất cho toàn bộ dữ liệu

$$\frac{\partial J^0(\mathbf{w})}{\partial \mathbf{w}} = - \sum_{n=1}^N \frac{\partial \mathbf{L}_n(\mathbf{w})}{\partial \mathbf{w}} = -X(A - \bar{Y})^T$$

Trường hợp sử dụng hiệu chỉnh  $L_2$ , hàm tổn thất xác định theo (.). Đạo hàm hàm tổn thất cho 1 dữ liệu và cho toàn bộ N dữ liệu tương ứng là

$$\frac{\partial J_n(w)}{\partial w} = -x_n e_n^T + \lambda w$$

và

$$\frac{\partial J(w)}{\partial w} = X(A - \bar{Y})^T + \lambda w \quad (**)$$

- **Bước 4:** Cập nhật trọng số:

$$\mathbf{w} = \mathbf{w} - \eta \cdot \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}}$$

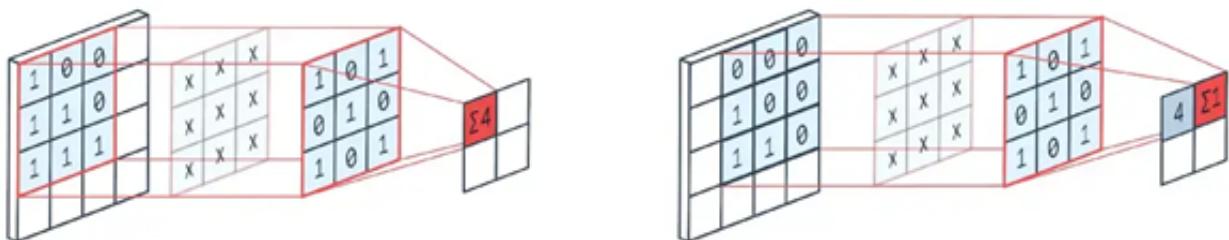
Với  $\eta$ (learning rate)  $> 0$

### 4.3 Convolutional Neural Network

CNN là một loại mạng nơ-ron nhân tạo được sử dụng rộng rãi trong xử lý ảnh và thị giác máy tính. Mạng này sử dụng các lớp tích chập (convolutional layers) để tìm hiểu và trích xuất các đặc trưng quan trọng trong ảnh thông qua việc áp dụng các bộ lọc (kernels) và thực hiện phép tích chập giữa bộ lọc và vùng nhỏ trong ảnh.

#### MỘT SỐ LỚP PHÔ BIẾN TRONG MẠNG CNN

**Lớp tích chập (Conv2D):** Tích chập sử dụng các bộ lọc để thực hiện phép tích chập khi đưa chúng đi qua đầu vào I theo các chiều của nó.



Hình 4.3: Hình ảnh minh họa cách tính tích chập

Giả sử ảnh đầu vào I có kích thước  $M \times N$  pixel và kernel W có kích thước  $K \times K$  phần tử, ảnh đầu ra F có cùng kích thước  $M \times N$  là kết quả tích chập hai chiều

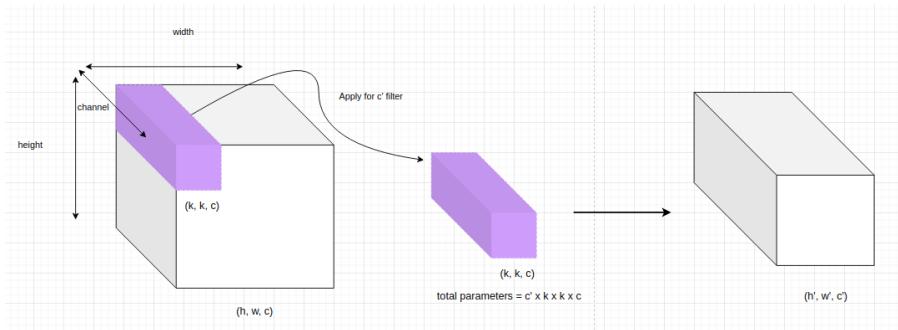
giữa ảnh đầu vào I và kernel W. Gọi C là ma trận đầu ra có giá trị các phần tử được tính theo công thức:

$$C(r, c) = \sum_{i=1}^K \sum_{j=1}^K I(r + i - \frac{K}{2} - 1, c + j - \frac{K}{2} - 1)W(i, j)$$

Với r, c là chỉ số hàng và cột của vị trí cần tính trong ma trận C; i, j là chỉ số hàng và cột của ma trận kernel W.

Có một số dạng tích chập phổ biến trong bài toán xử lý ảnh:

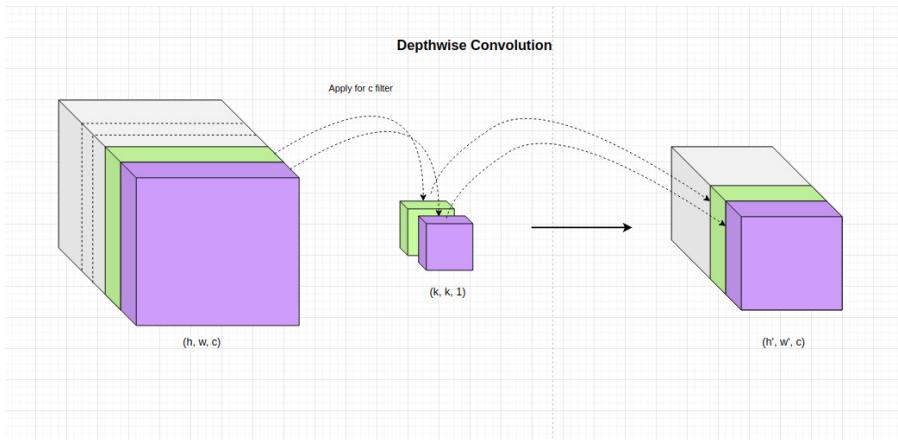
- Tích chập 2 chiều thông thường:



Hình 4.4: Hình ảnh minh họa cách tính tích chập thường

Giả sử đầu vào có kích thước  $h \times w \times c$ , tích chập thông thường sẽ cần  $k \times k \times c$  tham số để thực hiện tích chập trên toàn bộ độ sâu của layers. Mỗi bộ lọc sẽ tạo ra một ma trận có kích thước  $h' \times w' \times 1$ . Áp dụng  $c'$  bộ lọc khác nhau sẽ tạo ra đầu ra có kích thước  $h' \times w' \times c'$  và số lượng tham số cần dùng là  $c' \times k \times k \times c$ .

- Tích chập chiều sâu (Depthwise Convolution): thực hiện chia khối input 3D thành những lát cắt ma trận theo độ sâu và thực hiện tích chập trên từng lát cắt như hình minh họa.

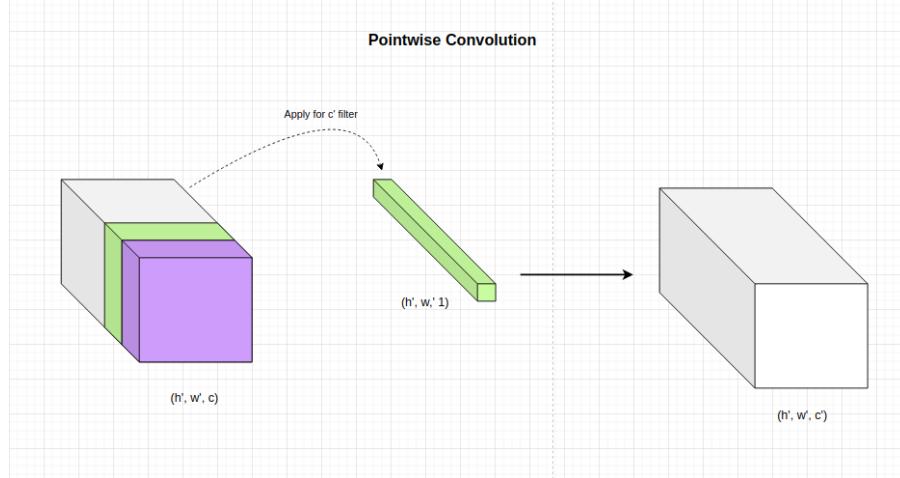


Hình 4.5: Hình ảnh minh họa cách tính tích chập chiều sâu

Tích chập này sẽ giúp quá trình học và nhận diện đặc trưng sẽ được tách biệt theo từng bộ lọc và đặc biệt giảm thiểu khối lượng tính toán và số lượng tham số

xuống còn  $c \times k \times k$ , ít hơn  $c'$  lần so với tích chập thông thường.

- Tích chập điểm (Pointwise Convolution): Có tác dụng thay đổi độ sâu của output bước trên từ  $c$  sang  $c'$  với bộ lọc kích thước  $1 \times 1 \times c$ . Như vậy các kích thước về chiều dài và chiều rộng sẽ không bị thay đổi.



Hình 4.6: Hình ảnh minh họa cách tính tích chập điểm

**Lớp BatchNormalization:** dùng để chuẩn hóa giá trị input của layer bất kì, đưa chúng về xếp xỉ phân phối chuẩn với trung bình xấp xỉ 0 và phương sai xấp xỉ 1.

Công thức tính trung bình của mini-batch:

$$\mu_B \leftarrow \frac{1}{m} \sum_{i=1}^m x_i$$

Công thức tính phương sai của mini-batch:

$$\sigma_\beta^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_B)^2$$

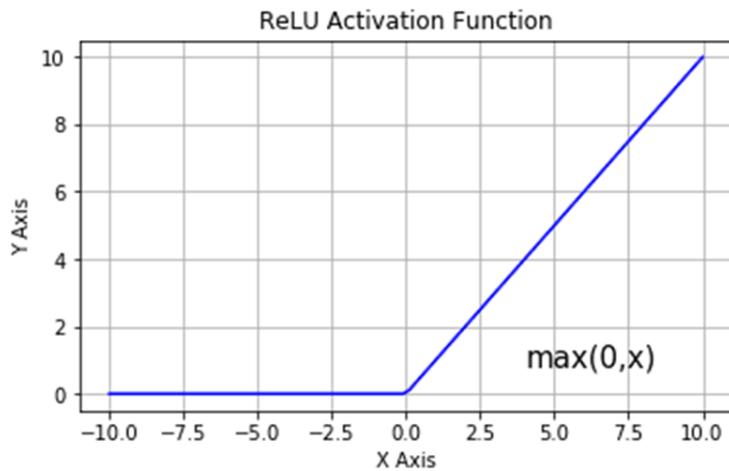
Chuẩn hóa:

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}, \quad y = \gamma \hat{x}_i + \beta$$

**Lớp Activation ReLU:** giúp mô hình học được mối quan hệ phi tuyến, có công thức:

$$f(x) = \max(0, x)$$

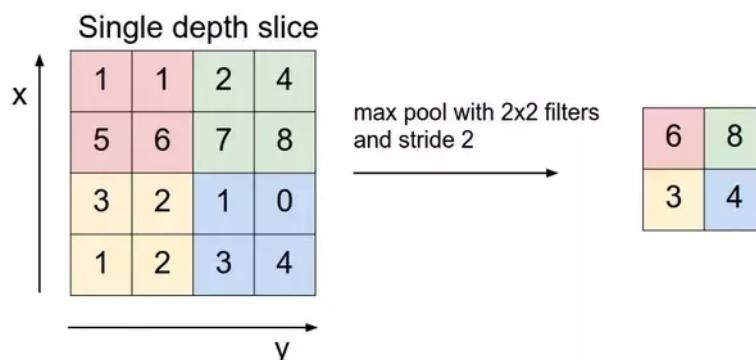
**Lớp MaxPooling:** giúp giảm kích thước của đầu vào, tăng tính đơn giản của mô hình, giúp giảm lượng tham số của mô hình và giảm khả năng overfitting. Không gian pooling còn được gọi là lấy mẫu con hoặc lấy mẫu xuyên, làm giảm



Hình 4.7: Hàm kích hoạt ReLU

kích thước của mỗi map nhưng vẫn giữ lại thông tin quan trọng. MaxPooling giảm kích thước đầu vào bằng cách chọn giá trị lớn nhất từ các vùng mà MaxPooling trượt qua. Cụ thể, lớp MaxPooling thực hiện các bước như sau:

- Chia input thành các khối có kích thước bằng nhau (thường là các khối có kích thước 2x2 hoặc 3x3).
- Chọn giá trị lớn nhất từ mỗi khối và ghi nhận giá trị đó vào output của lớp MaxPooling.
- Tiếp tục các bước trên trên toàn bộ input, cho đến khi không thể chia khối được nữa.

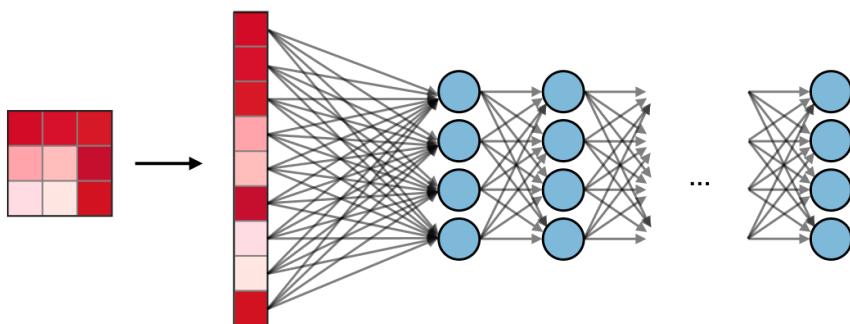


Hình 4.8: Hình ảnh minh họa cách tính MaxPooling với kích thước 2x2

**Lớp Dropout:** giúp loại bỏ một số node trong mạng neural trong quá trình huấn luyện. Cụ thể, trong mỗi lần forward hoặc backward, một số node sẽ bị tạm thời loại bỏ với xác suất dropout\_rate. Khi đó, số lượng node trong mạng được giảm xuống và các node còn lại sẽ cần phải học cách hỗ trợ các node bị loại bỏ. Điều này sẽ giúp cho mô hình tránh được trường hợp overfitting.

**Lớp Flatten:** giúp chuyển đổi đầu ra của các lớp tích chập hoặc pooling thành vecto 1 chiều để sử dụng cho các lớp kết nối đầy đủ (Fully Connect). Ví dụ, nếu đầu ra của lớp tích chập hoặc lớp pooling là một tensor có kích thước 3D (ví dụ: (batch\_size, height, width, channels)), lớp Flatten sẽ chuyển đổi đầu ra này thành một vector 1D có kích thước (batch\_size, height \* width \* channels).

**Lớp Fully Connect:** Tầng kết nối đầu đủ (FC) nhận đầu vào là các dữ liệu đã được làm phẳng, mà mỗi đầu vào đó được kết nối đến tất cả neuron. Trong mô hình mạng CNNs, các tầng kết nối đầy đủ thường được tìm thấy ở cuối mạng và được dùng để tối ưu hóa mục tiêu của mạng ví dụ như độ chính xác của lớp.



Hình 4.9: Hình ảnh minh họa lớp Fully Connect

**Lớp Activation Softmax:** đưa ra xác suất dự đoán rơi vào các class.

Ngoài ra, nhóm còn sử dụng mô hình checkpoint sẽ giúp máy theo dõi chỉ số monitor (như trên code là giá trị val\_loss) trên tập dữ liệu validation để lưu trữ mô hình có chỉ số monitor nhỏ nhất hoặc lớn nhất (tùy mode người dùng chọn).

Đối với mỗi epoch, trong quá trình huấn luyện, callback này sẽ lưu trữ trọng số của mô hình vào tệp 'model1.hdf5' nếu giá trị của hàm mất mát trên tập validation tốt hơn so với các epoch trước đó. Khi huấn luyện kết thúc, callback này sẽ trả về trọng số của mô hình tốt nhất được lưu trữ.

Việc lưu trữ trọng số của mô hình tốt nhất được sử dụng để tránh việc quá trình huấn luyện bị quá khớp (overfitting) trên dữ liệu huấn luyện, và đồng thời giúp mô hình có khả năng dự đoán tốt trên dữ liệu mới.

## THUẬT TOÁN TỐI ƯU ADAM

Adam (Adaptive Moment Estimation) là một thuật toán tối ưu hóa các hàm mục tiêu ngẫu nhiên dựa trên gradient bậc nhất và sự ước tính thích ứng của các moment bậc thấp. Phương pháp này tính toán từng tỉ lệ học thích ứng cho các tham số khác nhau. Adam sử dụng ước tính của khoảng thời gian thứ nhất và thứ hai để điều chỉnh tốc độ học cho mỗi trọng số của mạng nơ-ron.

Adam optimizer là một thuật toán kết hợp giữa thuật toán RMS prop và thuật

toán momentum. Giống như Gradient Descent thì Adam cũng cập nhật các trọng số qua từng vòng lặp, và Adam là tổ hợp của hai phương pháp Momentum và RMSProp sẽ có siêu tham số sau  $\beta_1$  (Trong Momentum),  $\beta_2$  (trong RMSProp) và  $\alpha$  (learning rate).

Dưới đây là quá trình tính toán trong quá trình lặp của thuật toán Adam:

- Khởi tạo:

$$V_{dw} = 0, s_{dw} = 0, V_{db} = 0, S_{db} = 0$$

- Trong từng vòng lặp, tính:

$$\begin{aligned} V_{dw} &= \beta_1 \cdot V_{dw} + (1 - \beta_1) \cdot dw \\ V_{db} &= \beta_1 \cdot V_{db} + (1 - \beta_1) \cdot db \\ S_{dw} &= \beta_2 \cdot S_{dw} + (1 - \beta_2) \cdot dw^2 \\ S_{db} &= \beta_2 \cdot S_{db} + (1 - \beta_2) \cdot db^2 \end{aligned}$$

- Cập nhật trong vòng lặp

$$\begin{aligned} V_{dw} &= \frac{V_{dw}}{1 - \beta_1^t}, V_{db} = \frac{V_{db}}{1 - \beta_1^t} \\ S_{dw} &= \frac{S_{dw}}{1 - \beta_2^t}, S_{db} = \frac{S_{db}}{1 - \beta_2^t} \\ w &:= w - \alpha \cdot \frac{v_{dw}}{\sqrt{S_{dw} + \epsilon}} \\ b &:= b - \alpha \cdot \frac{v_{db}}{\sqrt{S_{db} + \epsilon}} \end{aligned}$$

Để việc descent được thực hiện nhanh hơn, thuật toán đã sử dụng hai kỹ thuật:

- Tính exponential moving average của giá trị đạo hàm lưu vào biến  $m$  và sử dụng nó là tử số của việc cập nhật hướng. Với ý nghĩa là nếu  $m$  có giá trị lớn, thì việc descent đang đi đúng hướng và chúng ta cần bước nhảy lớn hơn để đi nhanh hơn. Tương tự, nếu giá trị  $m$  nhỏ, phần descent có thể không đi về hướng tối thiểu và chúng ta nên đi 1 bước nhỏ để thăm dò. Đây là phần momentum của thuật toán.

- Tính exponential moving average của bình phương giá trị đạo hàm lưu vào biến  $v$  và sử dụng nó là phần mẫu số của việc cập nhật hướng. Với ý nghĩa như sau: Giả sử gradient mang các giá trị dương, âm lẫn lộn, thì khi cộng các giá trị lại theo công thức tính  $s$  ta sẽ được giá trị  $s$  gần số 0. Do âm dương lẫn lộn nên nó bị triệt tiêu lẫn nhau. Nhưng trong trường hợp này thì  $v$  sẽ mang giá trị lớn. Do đó, trong trường hợp này, chúng ta sẽ không hướng tới cực tiểu, chúng ta sẽ

không muốn đi theo hướng đạo hàm trong trường hợp này. Chúng ta để  $v$  ở phần mẫu vì khi chia cho một giá trị cao, giá trị của các phần cập nhật sẽ nhỏ, và khi  $v$  có giá trị thấp, phần cập nhật sẽ lớn. Đây chính là phần tối ưu RMSProp của thuật toán.

Ở đây,  $s$  được xem như là moment thứ nhất,  $v$  xem như là moment thứ hai, nên thuật toán có tên là “Adaptive moment estimation”.

#### 4.4 Phương pháp Transfer Learning

Transfer learning là một kỹ thuật trong học máy sử dụng các kiến thức đã học được từ một tác vụ hoặc một tập dữ liệu cho tác vụ hoặc tập dữ liệu khác liên quan. Thay vì xây dựng và huấn luyện một mô hình từ đầu, ta sử dụng pre-trained model - mô hình đã được huấn luyện trước đó và thích ứng lại cho bài toán mới.

Các phương pháp phổ biến trong Transfer learning:

1. Feature Extraction (Trích xuất đặc trưng): Phương pháp này bao gồm sử dụng một mô hình đã được huấn luyện trước (pre-trained model) để trích xuất các đặc trưng từ dữ liệu.
2. Fine-tuning (Điều chỉnh lại): Phương pháp này liên quan đến việc tiếp tục huấn luyện một phần hoặc toàn bộ mô hình đã được huấn luyện trước trên tập dữ liệu mới.

# Chương 5

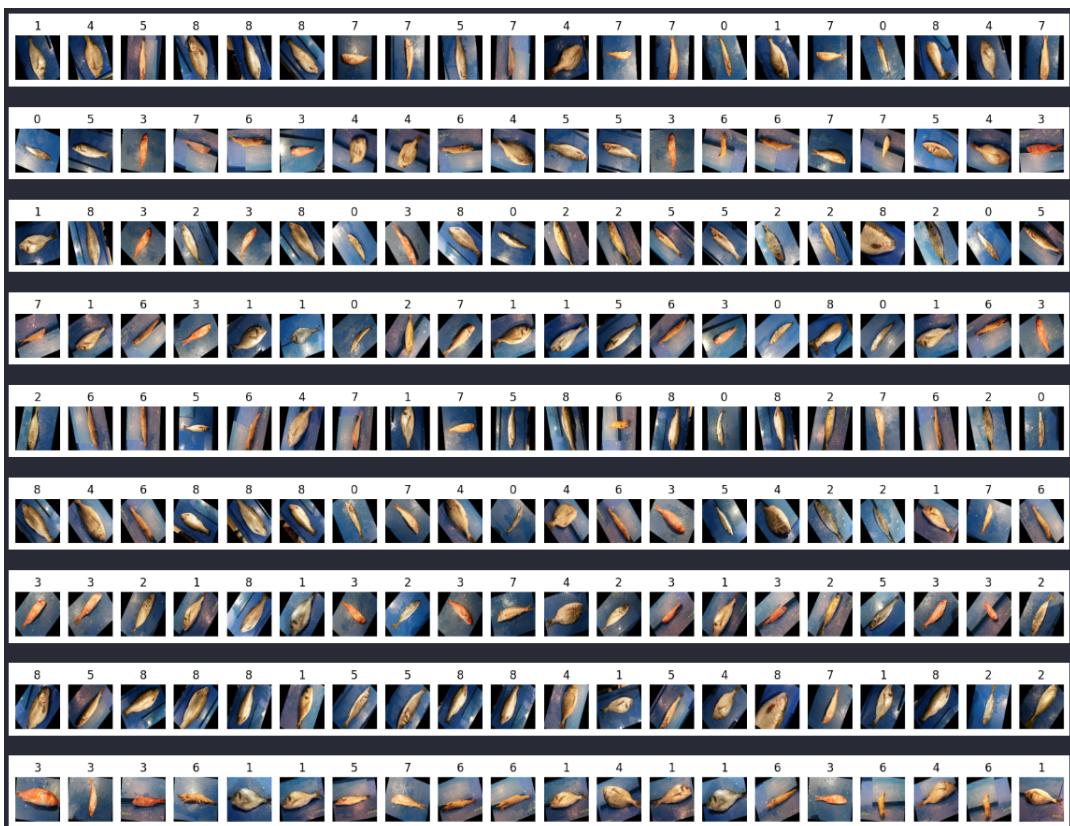
## Xây dựng mô hình

### 5.1 Phân cụm dữ liệu

Để thực hiện phân cụm dữ liệu, nhóm sử dụng thuật toán KMeans cùng với một số kịch bản kết hợp với dữ liệu đã phân và kĩ thuật Transfer Learning sử dụng mạng MobileNetV2.

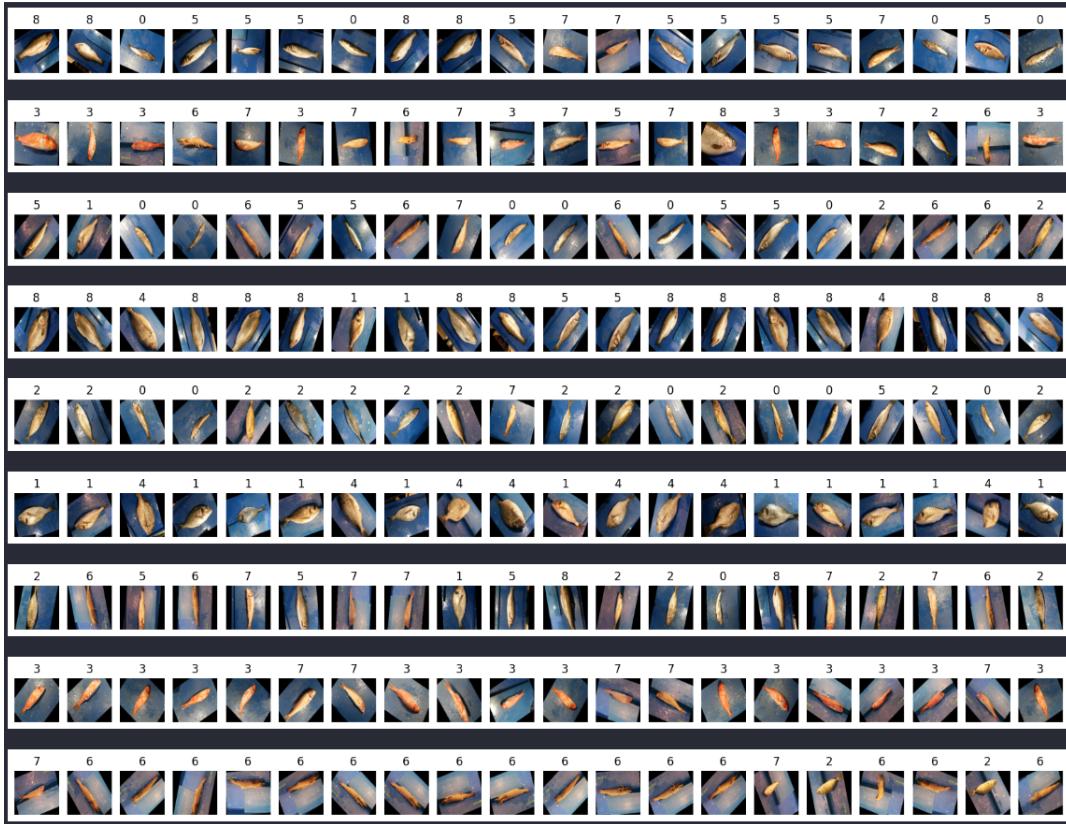
Dưới đây là kết quả thực nghiệm của các kịch bản:

**Dữ liệu gốc:**



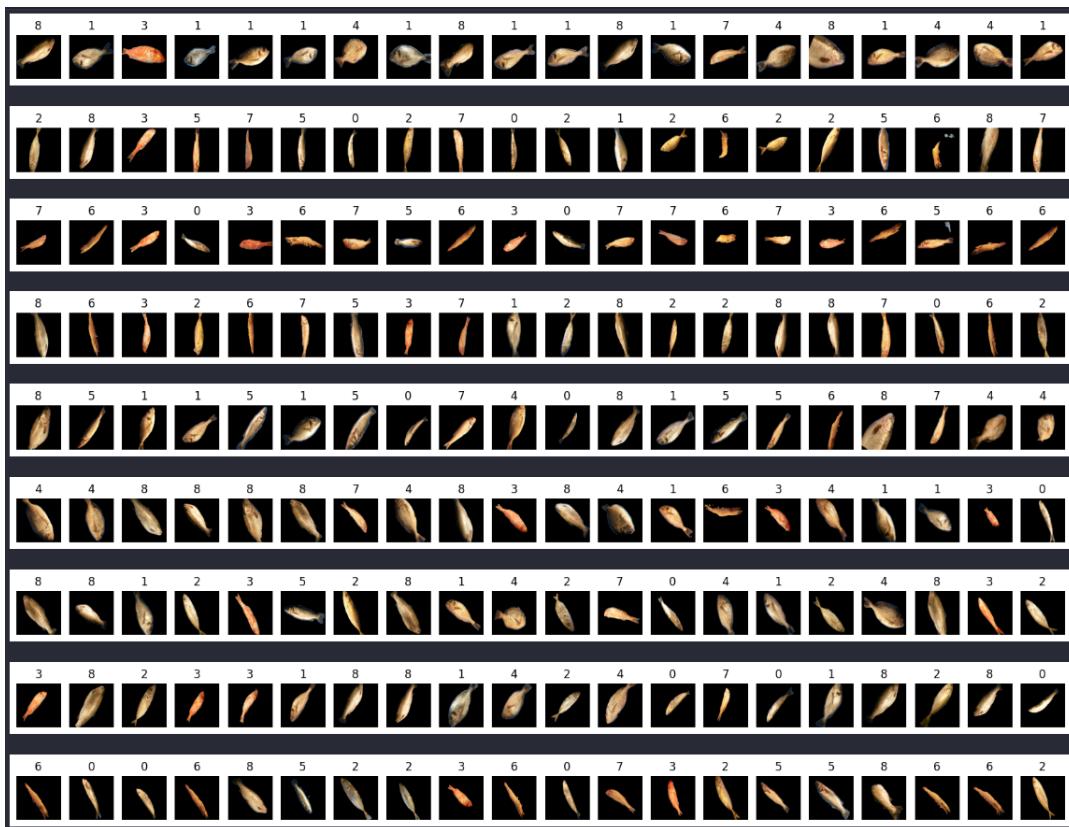
Hình 5.1: Hình ảnh tâm và 20 ảnh gần tâm cụm nhất

## Dữ liệu gốc kết hợp Transfer Learning:



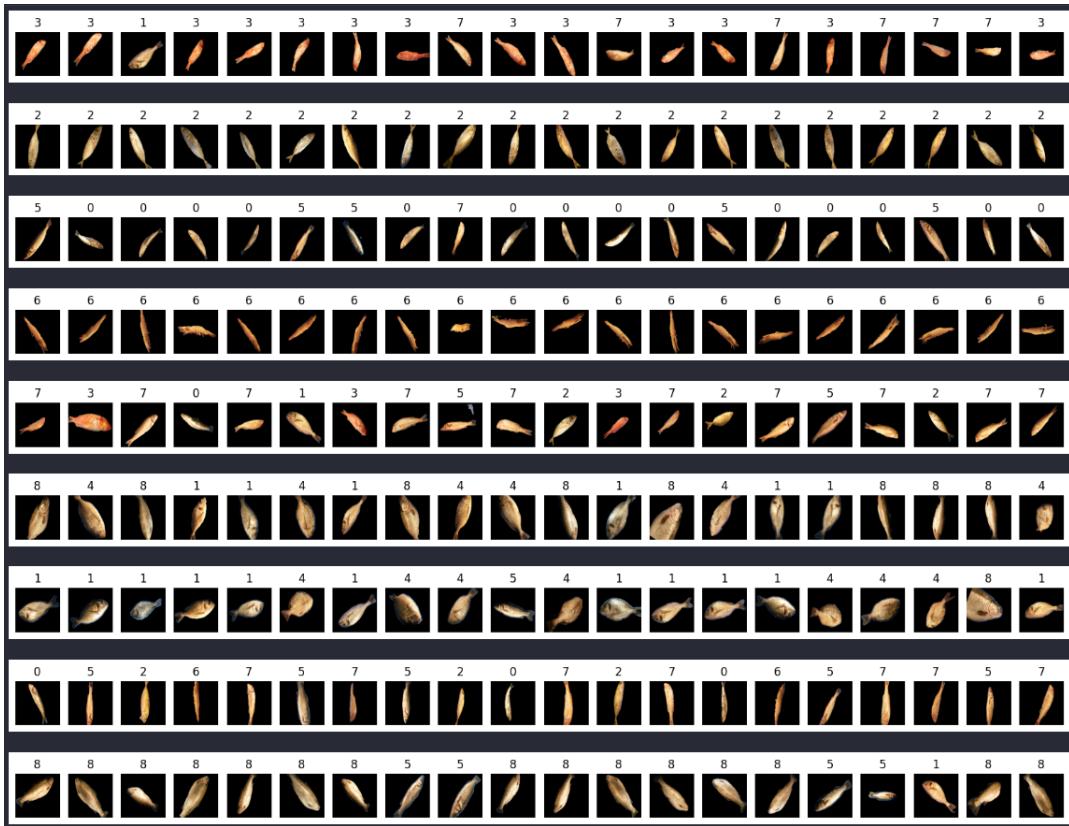
Hình 5.2: Hình ảnh tâm và 20 ảnh gần tâm cụm nhất

## Dữ liệu đã phân đoạn.



Hình 5.3: Hình ảnh tâm và 20 ảnh gần tâm cụm nhất

## Dữ liệu đã phân đoạn kết hợp với Transfer Learning.



Hình 5.4: Hình ảnh tâm và 20 ảnh gần tâm cụm nhất

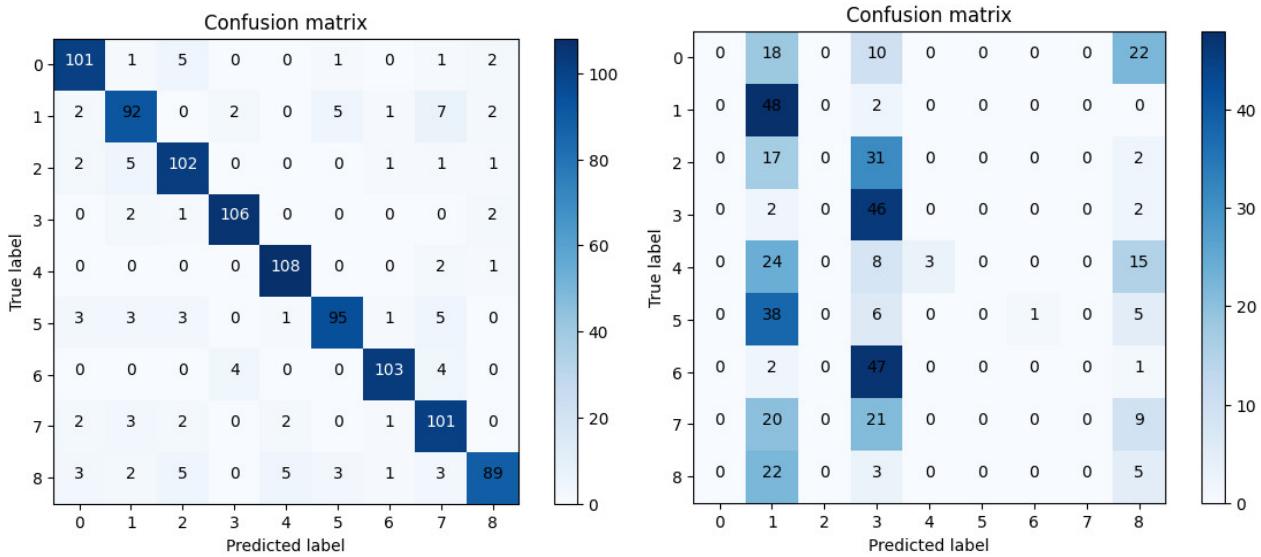
## 5.2 Phân loại dữ liệu

### 5.2.1 Softmax Regression

Nhóm thực hiện xây dựng mô hình Softmax Regression dưới hai kịch bản sử dụng với dữ liệu gốc sau khi thực hiện đọc, duỗi ảnh và sử dụng thêm một bước transfer learning với mạng MobileNetV2.

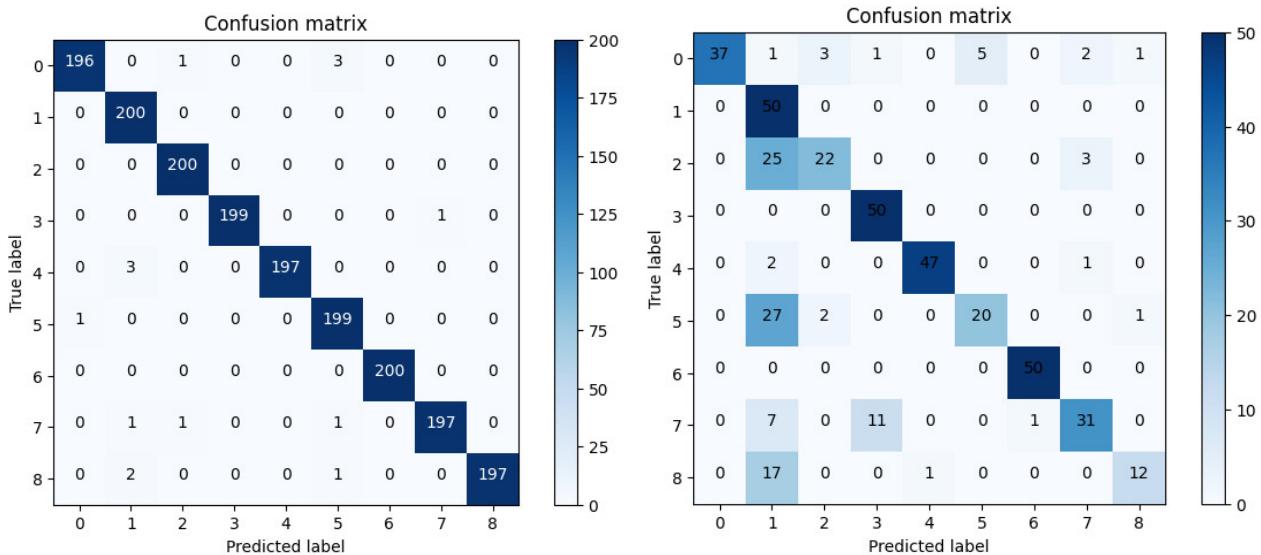
Dưới đây là kết quả thực nghiệm của các kịch bản:

**Dữ liệu gốc:** Kết quả mang lại trên tập validation cao: accuracy: 89.7%, recall: 89.7%, precision: 89.85% nhưng lại thật sự thấp trên tập test: accuracy: 23.72%, recall: 23.4%, precision: 17.75%



Hình 5.5: Ma trận Confusion trên hai tập validation(bên trái) và test(bên phải)

**Sử dụng Transfer Learning bằng mạng MobileNetV2:** Kết quả mang lại có thể khẳng định rằng nó giúp giảm thiểu vấn đề overfit: tập validation có accuracy: 99%, recall: 99%, precision: 99%, tập test: có accuracy: 74%, recall: 74%, precision: 82%



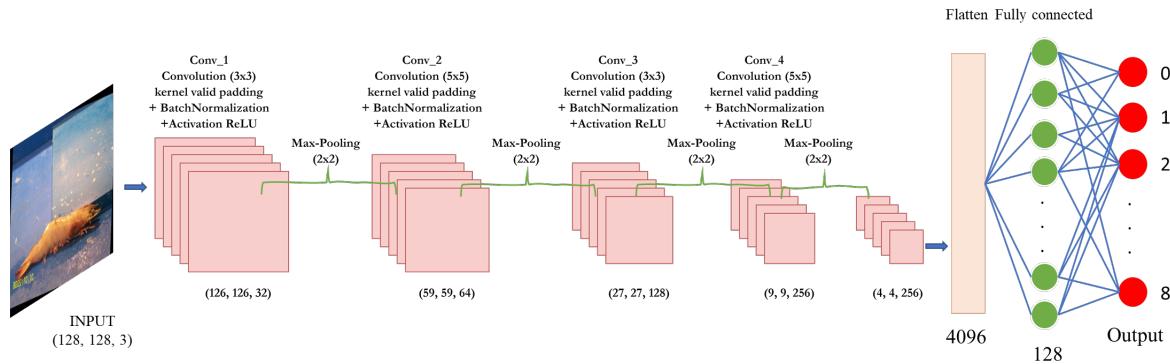
Hình 5.6: Ma trận Confusion trên hai tập validation(bên trái) và test(bên phải)

### 5.2.2 Convolutional Neural Network

#### Mô tả cấu trúc mạng sử dụng:

- Đầu vào của mạng có kích thước ( $h, w, c$ ) ứng với chiều cao, chiều rộng và số kênh màu của ảnh. Mô hình này bao gồm nhiều lớp tích chập (Conv2D) với kích thước kernel khác nhau ( $3 \times 3, 5 \times 5$ ) với số lượng filter tăng dần (32, 64, 128, 256). Bên cạnh đó, mỗi lớp tích chập được kết hợp với lớp BatchNormalization

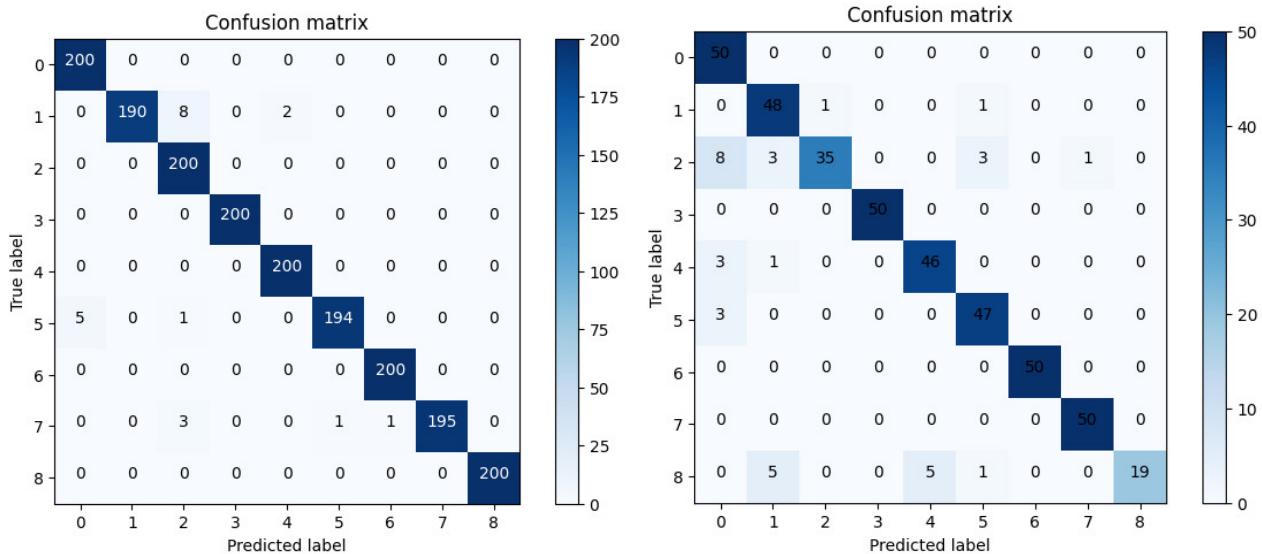
để chuẩn hóa dữ liệu đầu vào, tăng tốc độ hội tụ và giảm overfitting. Sau đó sẽ là một lớp Activation với hàm kích hoạt ReLU và một lớp MaxPool2D với kích thước pool\_size = (2,2) để giảm kích thước đầu ra và độ phức tạp của mô hình. Cuối cùng là một lớp Flatten được sử dụng để làm phẳng các đặc trưng, sau đó được đưa qua một số lớp Fully Connected (Dense) và lớp output với 9 neuron với hàm kích hoạt Softmax để phân loại ảnh vào 9 lớp.



Hình 5.7: Mô tả cấu trúc sử dụng

- Mô hình được đào tạo bằng hàm mất mát categorical\_crossentropy và tối ưu hóa bằng thuật toán Adam và sử dụng độ chính xác (accuracy) để đánh giá hiệu suất mô hình.

**Kết quả thực nghiệm:** accuracy: 98.83%, recall: 98.83%, precision: 98.88% trên tập validation và accuracy: 91.86%, recall: 91.86%, precision: 92.81% trên tập test

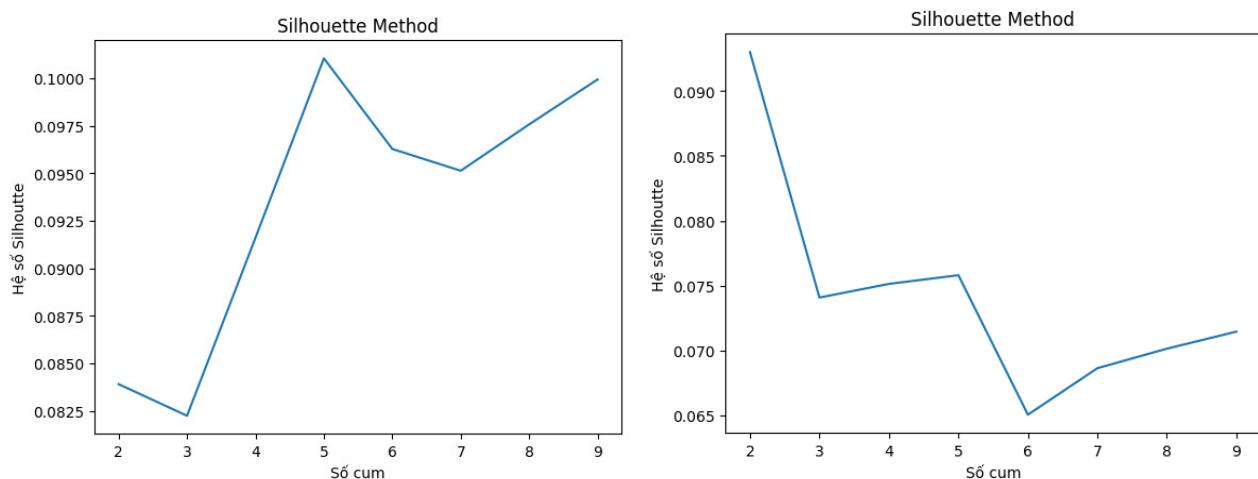


Hình 5.8: Ma trận Confusion trên hai tập validation(bên trái) và test(bên phải)

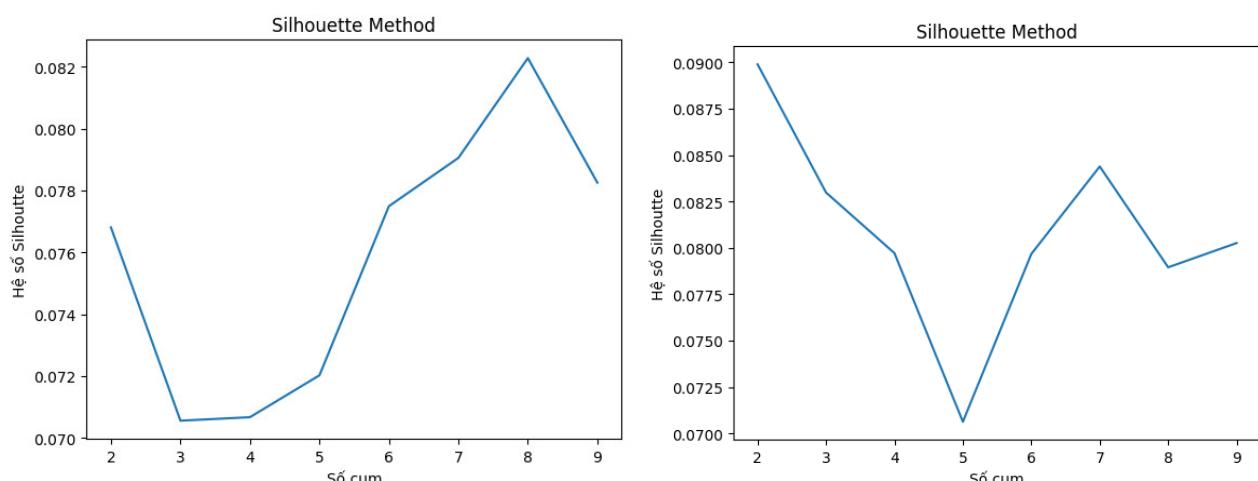
# Chương 6

## Kết quả và thảo luận

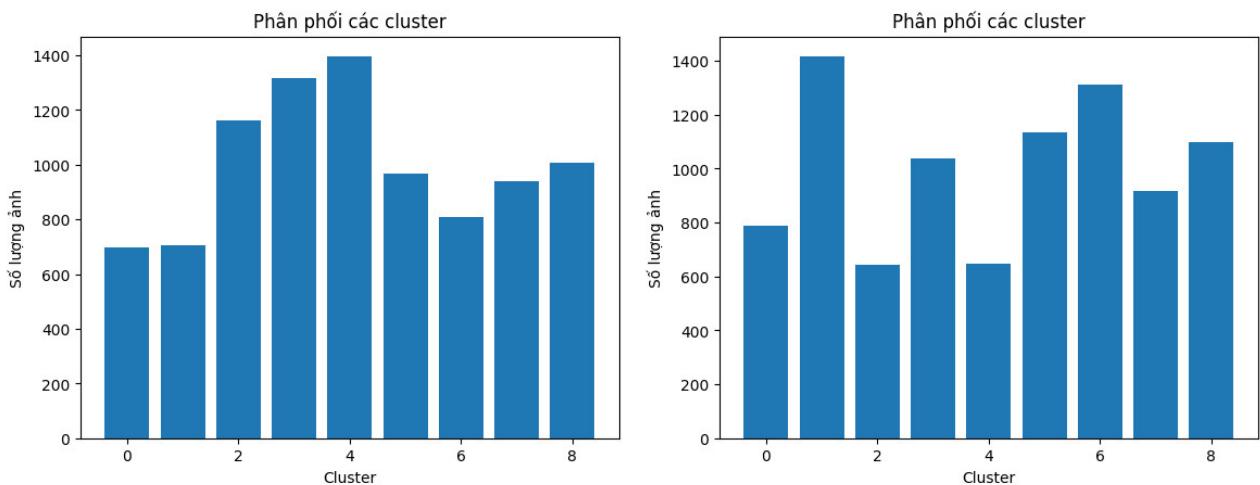
### 6.1 Phân cụm dữ liệu



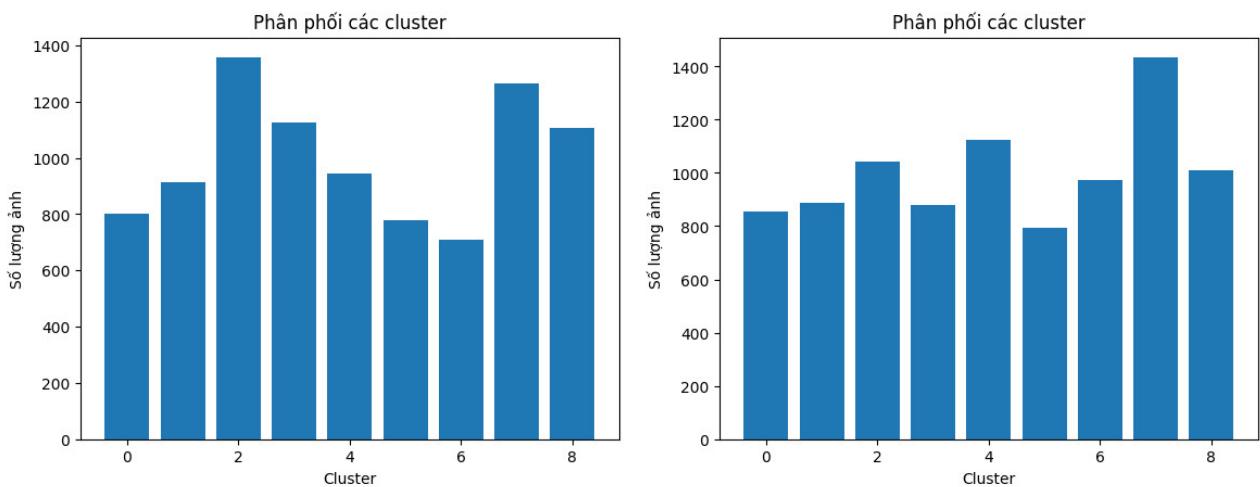
Hình 6.1: Phân bố số lượng hình ảnh gốc khi thực hiện KMeans (bên trái), KMeans kết hợp với Transfer (Bên phải)



Hình 6.2: Hệ số Silhouette trên ảnh đã phân đoạn của KMeans(bên trái) và KMeans kết hợp Transfer(bên phải)



Hình 6.3: Phân bố số lượng hình ảnh trên ảnh gốc khi thực hiện KMeans (bên trái), KMeans kết hợp với Transfer (Bên phải)



Hình 6.4: Phân bố số lượng hình ảnh trên ảnh đã phân đoạn khi thực hiện KMeans (bên trái), KMeans kết hợp với Transfer (Bên phải)

Với tập dữ liệu bao gồm 9000 ảnh tương ứng 1000 ảnh mỗi loại cá, khi nhìn vào biểu đồ cột thể hiện phân bố số lượng ảnh ảnh ở mỗi cụm khi thực hiện thuật toán, nhận thấy được việc sử dụng trên dữ liệu gốc không mang lại hiệu quả cao khi có nhiều cụm chỉ đạt tầm 700 ảnh, ngoài ra, kết hợp với đồ thị thể hiện hệ số Silhouette (tốt nhất nên để 3 cụm) và ảnh trực quan 20 ảnh gần tâm cụm nhất ở chương 5, kết quả phân cụm không tốt khi các ảnh chung 1 cụm chưa có nhiều điểm chung như kích thước, màu sắc

Việc thực hiện kết hợp với Transfer learning đã mang lại hiệu quả tốt hơn với những hình ảnh trong cụm đã có sự đồng dạng về hình dáng.

Kết quả đạt tốt nhất khi thực hiện trên dữ liệu đã được phân đoạn kết hợp với kĩ thuật transfer khi số lượng ảnh phân bố đồng đều và sắp sỉ với con số 1000 hơn, đồng thời các ảnh trong 1 cụm đã có sự giống nhau về kích thước, hình dạng, và màu sắc.

## 6.2 Phân loại dữ liệu

Hai kịch bản xây dựng mô hình Softmax Regression:

|                    | precision | recall | f1-score | support |                    | precision | recall | f1-score | support |
|--------------------|-----------|--------|----------|---------|--------------------|-----------|--------|----------|---------|
| Black Sea Sprat    | 0.00      | 0.00   | 0.00     | 50      | Black Sea Sprat    | 1.00      | 0.74   | 0.85     | 50      |
| Gilt Head Bream    | 0.23      | 0.98   | 0.38     | 50      | Gilt Head Bream    | 0.39      | 1.00   | 0.56     | 50      |
| Hourse Mackerel    | 0.00      | 0.00   | 0.00     | 50      | Hourse Mackerel    | 0.81      | 0.44   | 0.57     | 50      |
| Red Mullet         | 0.28      | 1.00   | 0.43     | 50      | Red Mullet         | 0.81      | 1.00   | 0.89     | 50      |
| Red Sea Bream      | 0.75      | 0.18   | 0.29     | 50      | Red Sea Bream      | 0.98      | 0.94   | 0.96     | 50      |
| Sea Bass           | 0.00      | 0.00   | 0.00     | 50      | Sea Bass           | 0.80      | 0.40   | 0.53     | 50      |
| Shrimp             | 0.00      | 0.00   | 0.00     | 50      | Shrimp             | 0.98      | 1.00   | 0.99     | 50      |
| Striped Red Mullet | 0.00      | 0.00   | 0.00     | 50      | Striped Red Mullet | 0.84      | 0.62   | 0.71     | 50      |
| Trout              | 0.23      | 0.20   | 0.21     | 30      | Trout              | 0.86      | 0.40   | 0.55     | 30      |
| accuracy           |           |        | 0.27     | 430     | accuracy           |           |        | 0.74     | 430     |
| macro avg          | 0.17      | 0.26   | 0.15     | 430     | macro avg          | 0.83      | 0.73   | 0.73     | 430     |
| weighted avg       | 0.16      | 0.27   | 0.14     | 430     | weighted avg       | 0.83      | 0.74   | 0.74     | 430     |

Hình 6.5: Softmax regression(bên trái) và Softmax regression kết hợp với Transfer (bên phải)

Với mô hình Softmax Regression gốc, kết quả mang lại trên tập validation cao (xấp xỉ 90%) nhưng lại thấp trên tập test khi các chỉ số nhỏ hơn 25%. Có thể dễ dàng nhận thấy mô hình đang gặp hiện tượng overfit. Điều này làm tiền đề cho nhóm thực hiện kịch bản thứ 2, sử dụng kết hợp bước transfer learning trước khi tiến hành phân loại bằng mô hình Softmax Regression, kết quả mang lại ổn định hơn khi các chỉ số rơi vào khoảng 74% đến 83%.

Để giải thích cho hiện tượng này, nhóm đưa ra ý kiến do dữ liệu vẫn còn hạn chế, các dữ liệu trên tập huấn luyện được tăng cường bằng cách xoay và lật ảnh đơn giản, dẫn đến các dữ liệu trên tập validation có thể là những hình ảnh cá đã tồn tại trong huấn luyện nên mang lại dự đoán cao, nhưng khi gặp những ảnh cá chưa từng xuất hiện (tập kiểm tra), thuật toán chưa thể nhận ra được. Điều này khẳng định lần nữa việc trích xuất đặc trưng quan trọng trong xử lý ảnh là một nhiệm vụ vô cùng cần thiết và kết quả khi sử dụng Softmax Regression kết hợp với transfer learning đã giúp làm rõ hơn phần nào khẳng định đó.

Mô hình Softmax Regression và CNN:

|                    | precision | recall | f1-score | support |                    | precision | recall | f1-score | support |
|--------------------|-----------|--------|----------|---------|--------------------|-----------|--------|----------|---------|
| Black Sea Sprat    | 1.00      | 0.74   | 0.85     | 50      | Black Sea Sprat    | 0.78      | 1.00   | 0.88     | 50      |
| Gilt Head Bream    | 0.39      | 1.00   | 0.56     | 50      | Gilt Head Bream    | 0.84      | 0.96   | 0.90     | 50      |
| Hourse Mackerel    | 0.81      | 0.44   | 0.57     | 50      | Hourse Mackerel    | 0.97      | 0.70   | 0.81     | 50      |
| Red Mullet         | 0.81      | 1.00   | 0.89     | 50      | Red Mullet         | 1.00      | 1.00   | 1.00     | 50      |
| Red Sea Bream      | 0.98      | 0.94   | 0.96     | 50      | Red Sea Bream      | 0.90      | 0.92   | 0.91     | 50      |
| Sea Bass           | 0.80      | 0.40   | 0.53     | 50      | Sea Bass           | 0.90      | 0.94   | 0.92     | 50      |
| Shrimp             | 0.98      | 1.00   | 0.99     | 50      | Shrimp             | 1.00      | 1.00   | 1.00     | 50      |
| Striped Red Mullet | 0.84      | 0.62   | 0.71     | 50      | Striped Red Mullet | 0.98      | 1.00   | 0.99     | 50      |
| Trout              | 0.86      | 0.40   | 0.55     | 30      | Trout              | 1.00      | 0.63   | 0.78     | 30      |
| accuracy           |           |        | 0.74     | 430     | accuracy           |           |        | 0.92     | 430     |
| macro avg          | 0.83      | 0.73   | 0.73     | 430     | macro avg          | 0.93      | 0.91   | 0.91     | 430     |
| weighted avg       | 0.83      | 0.74   | 0.74     | 430     | weighted avg       | 0.93      | 0.92   | 0.92     | 430     |

Hình 6.6: Softmax regression(bên trái) và Convolution neural network (bên phải)

Dễ dàng nhận thấy rằng mô hình CNN có kết quả tốt hơn, mặc dù thuật toán Softmax Regression đã được qua các bước nhân tích chập từ transfer learning để trích xuất các đặc trưng quan trọng nhưng do CNN còn sử dụng một số các lớp Fully Connect, giúp kết nối các đặc trưng ảnh sau khi được làm phẳng qua lớp Flatten() giúp mô hình học được các quy tắc phức tạp và tương quan giữa các đặc trưng trích xuất và các lớp đối tượng, tìm mối quan hệ tốt nhất giữa các đặc trưng với lớp đầu ra

Từ những phân tích trên, có thể kết luận: đối với tác vụ phân loại cá trong bộ dữ liệu A Large Scale Fish nên sử dụng mạng Convolution Neural.

---

Hết.