TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA TOÁN CƠ TIN HỌC

BÁO CÁO CUỐI KỲ

HỌC MÁY

Đề tài:

ỨNG DỤNG MỘT SỐ MÔ HÌNH HỌC SÂU TRONG PHÁT HIỆN VÀ NHẬN DẠNG MỘT SỐ LOÀI ĐỘNG VẬT

Giảng viên:	CAO VĂN CHUNG	
Sinh viên thực hiện:	TRỊNH THỊ NGỌC MAI	20001948
	VƯƠNG THÙY DƯƠNG	20002042
	NGUYỄN HOÀNG HƯNG	20002058

Hà Nội, 05-2023

Mục lục

LỜI NÓI ĐẦU

Nhận dạng động vật là một vấn đề quan trọng trong bảo tồn và quản lý động vật hoang dã. Tuy nhiên, việc nhận dạng động vật bằng phương pháp truyền thống yêu cầu sự can thiệp của các chuyên gia và tốn nhiều thời gian. Vì vậy, việc sử dụng học sâu để nhận dạng động vật là một hướng đi mới tiềm năng để giải quyết vấn đề này.

Trong báo cáo của mình, tôi sẽ trình bày các khái niệm cơ bản về học sâu và các phương pháp sử dụng học sâu để nhận dạng động vật. Tôi cũng sẽ trình bày về các thách thức và cơ hội trong việc sử dụng học sâu để nhận dạng động vật và đưa ra các hướng phát triển trong tương lai.

Báo cáo này nhằm mục đích giúp quý vị hiểu rõ hơn về việc sử dụng học sâu để nhận dạng động vật và đóng góp vào việc bảo tồn và quản lý động vật hoang dã. Chúng tôi hy vọng rằng báo cáo này sẽ giúp quý vị có cái nhìn tổng quan về đề tài này và đưa ra các quan điểm và đánh giá cho phương pháp sử dụng học sâu trong nhận dạng động vật.

Trong bài báo cáo, chúng tôi sẽ trình bày cách nhận diện động với mô hình Mask R-CNN sử dụng backbone model là ResNet-50 và ResNet-101. Nội dung của bài báo cáo như sau:

Chương I: Đặt vấn đề
Chương II: Cơ sở lý thuyết
Chương III: Thiết kế và xây dựng mô hình
Chương IV: Kết quả nghiên cứu.
Chương V: Thảo luận và đánh giá kết quả.
Chương VI: Kết luận
Chương VII: Tài liệu tham khảo

1. Đặt vấn đề

Nhận diện các loài động vật là một ứng dụng quan trọng trong lĩnh vực sinh học và bảo tồn động vật hoang dã. Tuy nhiên, việc nhận diện động vật theo cách truyền thống đòi hỏi sự can thiệp của các chuyên gia và tốn nhiều thời gian. Vì vậy, sự phát triển của mô hình học sâu đã mở ra cơ hội cho việc phát triển các phương pháp tự động hóa nhận diện các loài động vật. Mô hình học sâu có thể học và nhận biết các đặc trưng phức tạp của hình ảnh động vật, từ đó giúp xác định chính xác loài động vật một cách nhanh chóng và hiệu quả. Với sự phát triển nhanh chóng của công nghệ và kho dữ liệu ngày càng lớn, việc sử dụng mô hình học sâu trong nhận diện các loài động vật là một hướng đi tiềm năng trong tương lai.

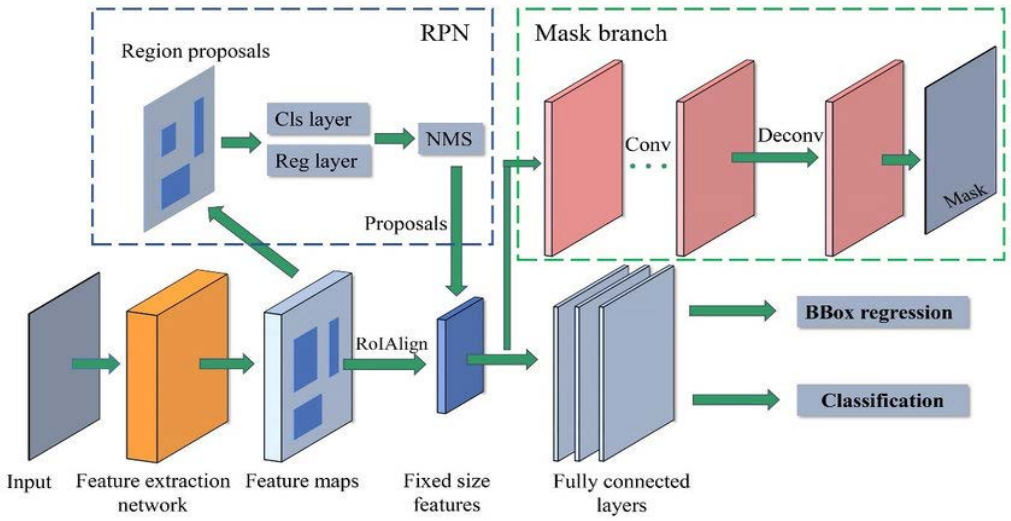
2. Cơ sở lý thuyết

2.1 Mask RCNN

Mask R-CNN (Mask Region-based Convolutional Neural Network) là một mô hình nhận dạng đối tượng và phân đoạn đối tượng tiên tiến. Được giới thiệu bởi Kaiming He, Georgia Gkioxari, Piotr Dollár và Ross Girshick vào năm 2017, Mask R-CNN là một phần mở rộng của Faster R-CNN (Region-based Convolutional Neural Network). Mô hình này kết hợp các yếu tố từ các nhiệm vụ cổ điển của thị giác máy tính như object detection, mục tiêu là phân loại các đối tượng riêng lẻ và xác định vị trí mỗi đối tượng bằng cách sử dụng hộp giới hạn (bounding box), và semantic segmentation, mục tiêu là phân loại mỗi pixel vào một tập hợp cố định các danh mục mà không phân biệt các trường hợp của đối tượng.

Mô hình Mask R-CNN kết hợp giữa ba tác vụ chính: phát hiện đối tượng (object detection), phân loại đối tượng (object classification) và phân đoạn đối tượng (object segmentation). Nó cho phép nhận dạng và phân loại các đối tượng trong ảnh cùng với việc tạo ra mặt nạ (mask) chi tiết cho từng đối tượng.

Cấu trúc của mô hình Mask R-CNN[2]:



Mô hình cấu trúc Mask R-CNN

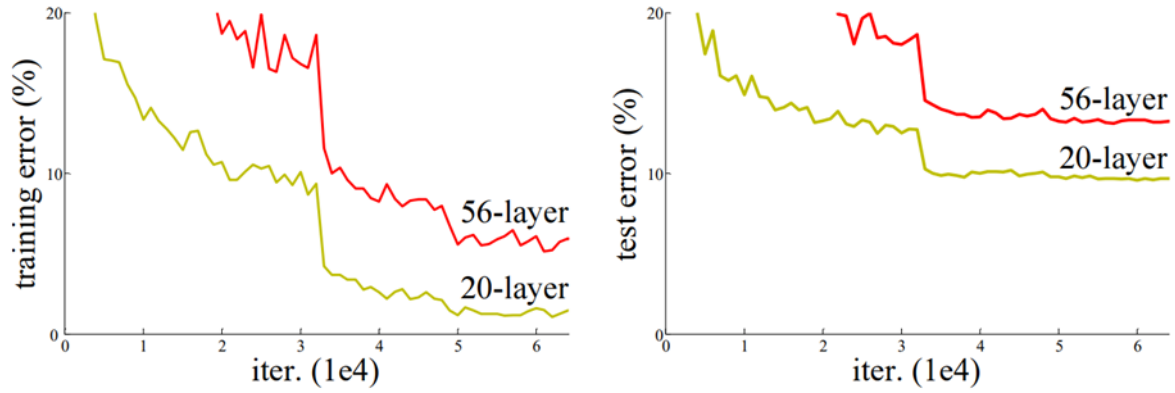
2.2 Backbone model – ResNet

2.2.1. Giới thiệu về ResNet

ResNet được phát triển bởi Kaiming He và cộng sự [1] và đã giành chiến thắng trong cuộc thi nhận dạng ảnh quy mô lớn vào năm 2015, cũng là một trong những mạng huấn luyện CNN nổi tiếng nhờ thành tích ấn tượng mà nó đạt được trong cuộc thi ILSVRC – ImageNet Large Scale Visual Recognition Challenge, được ImageNet – một hãng cở sở dữ liệu ảnh – tổ chức thường niên và được coi là cuộc thi Olympics quy mô thế giới trong lĩnh vực thị giác máy tính. ResNet giải quyết được vấn đề của học sâu truyền thống như khi mạng càng sâu (tăng số lượng lớp) thì đạo hàm sẽ bị vanishing (biến mất) hoặc explodes (bùng nổ). Vấn đề này có thể giải quyết bằng cách thêm Batch Normalization nó giúp chuẩn hóa đầu ra, giúp các hệ số trở nên cân bằng hơn, không quá nhỏ hoặc quá lớn nên sẽ giúp model dễ hội tụ hơn. Vấn đề thứ 2, do sự suy thoái (degradation), khi model càng sâu thì độ chính xác của model (accuracy) bắt đầu bão hòa, thậm chí là giảm.

Như hình vẽ bên dưới, khi mô hình càng nhiều lớp xếp chồng thì độ lỗi trong quá trình học (training error) lại cao hơn mô hình có ít lớp hơn. Như vậy, vấn đề ở đây không phải là do mô hình học quá khớp (overfitting) mà vấn đề giống như chúng ta thêm nhiều lớp vào mạng học sâu truyền thống, các lớp sau khi thêm vào sẽ không học thêm được gì nên

độ chính xác sẽ tương tự như mạng học sâu truyền thống mà không tăng. ResNet được ra đời nhằm giải quyết vấn đề này.



Training error (trái) và test error (phải) trên tập dữ liệu CIFAR-10 với mạng tiêu chuẩn 20 lớp và 56 lớp. Các mạng sâu hơn có training error và test error cao hơn.

ResNet có kiến trúc gồm nhiều residual block. Ý tưởng chính là bỏ qua một hoặc nhiều lớp bằng cách thêm kết nối với layer trước. Ý tưởng của residual block là khi đầu vào x đi qua một số layer, ta thu được F(x). Sau đó, cộng thêm x vào H(x) = F(x) + x. Mô hình sẽ dễ học hơn khi thêm đặc trưng (fearture) từ lớp trước vào.

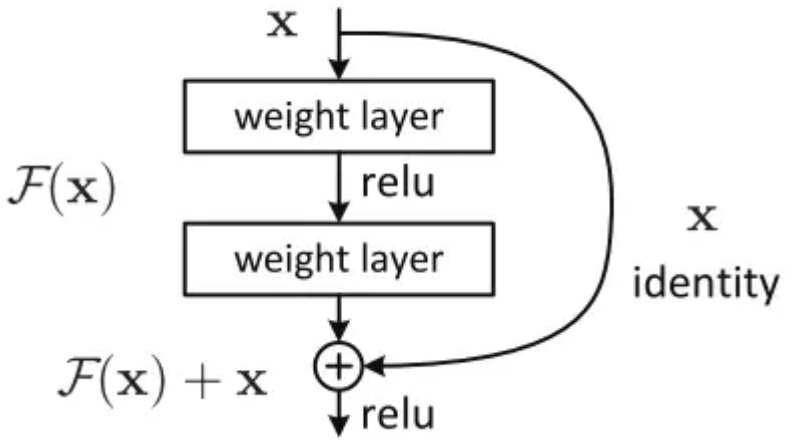
Mạng ResNet bao gồm một số khối ResNet block được xếp chồng lên nhau để tạo thành một mạng nơ-ron sâu. Mỗi khối ResNet block chứa các lớp convolution, batch normalization và activation (thường là ReLU), cùng với một shortcut connection. Dưới đây là cấu trúc chung của mạng ResNet:

1. **Convolutional layer**: Một lớp convolutional với kích thước kernel là 7x7 và stride là 2, được áp dụng cho đầu vào ban đầu.
2. **Max pooling layer:** Một lớp max pooling với kích thước kernel là 3x3 và stride là 2, để giảm kích thước của đầu vào.
3. **Các khối Residual Block**: Gồm nhiều khối Residual Block liên tiếp nhau. Mỗi khối Residual Block có thể chứa một hoặc nhiều lớp convolution và shortcut connections.
4. **Average pooling layer**: Một lớp average pooling với kích thước kernel là 7x7, được áp dụng sau khi các khối Residual Block đã được xử lý.
5. **Fully connected layer**: Một tầng fully connected với số lượng node tương ứng với số lớp đầu ra của bài toán, thường là 1000 node trong trường hợp mạng được huấn luyện trên tập dữ liệu ImageNet.
6. **Softmax layer**: Một lớp softmax được áp dụng để chuyển đổi các giá trị đầu ra thành xác suất dự đoán cho các lớp.

❖ **Residual Blocks**

Khối Residual (Residual Block)[1] là một thành phần quan trọng trong mạng ResNet (Residual Network). Nó được sử dụng để xây dựng kiến trúc mạng sâu bằng cách thêm

các kết nối shortcut (kết nối tắt) để tránh vấn đề gradient biến mất trong quá trình huấn luyện.



Cấu trúc của khối dư (Residual Block)

Mỗi Residual Block bao gồm các bước sau:

- Convolutional Layer đầu tiên: Convolutional Layer với kernel size 3x3 và số lượng filter được xác định trước. Sau đó sử dụng hàm kích hoạt ReLU để tăng tính phi tuyến và giảm số lượng tham số cần học.
- Convolutional Layer thứ hai: Convolutional Layer với kernel size 3x3 và số lượng filter được xác định trước. Sau đó tiếp tục sử dụng hàm kích hoạt ReLU.
- Shortcut Connection (đường đi tắt): Tạo một đường đi trực tiếp từ đầu vào đến đầu ra, cho phép thông tin truyền qua một số lớp một cách nhanh chóng. Điều này giúp mạng nơ-ron có khả năng học các đặc trưng cục bộ và toàn cục một cách hiệu quả, đồng thời cải thiện khả năng truyền thông tin và tăng cường khả năng học của mô hình.
- Kết hợp đầu vào và đầu ra: Đầu vào của residual block (đi qua Convolutional Layer thứ hai) được cộng với đầu ra của shortcut connection. Kết quả là đầu ra của residual block.

Điều quan trọng trong cấu trúc của residual block là sự kết hợp giữa Convolutional Layers và shortcut connection. Điều này cho phép mạng học được sự khác biệt (residual) của đặc trưng và tạo ra các đặc trưng phức tạp hơn, đồng thời tránh vấn đề mất gradient và giúp mạng ResNet có khả năng học sâu và hiệu quả

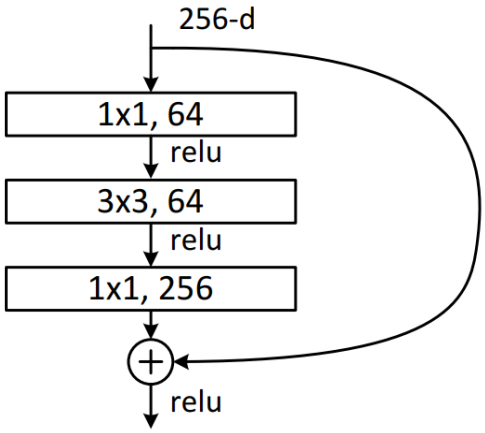
❖ **Khối Bottleneck**

Khối Bottleneck là một dạng đặc biệt của Residual Block trong mạng ResNet. Khối Bottleneck được sử dụng trong các phiên bản ResNet sâu hơn như ResNet-50, ResNet-101, và ResNet-152 để giảm độ phức tạp tính toán trong quá trình huấn luyện.

Khối Bottleneck thay thế cấu trúc đơn giản của Residual Block bằng cấu trúc Bottleneck, bao gồm các lớp convolution với các kích thước nhỏ hơn. Thay vì sử dụng hai lớp convolution 3x3 như trong Residual Block, khối Bottleneck sử dụng ba lớp convolution với kích thước 1x1, 3x3 và 1x1. Lớp convolution 1x1 ở giữa giúp giảm số chiều của đầu

vào, trong khi lớp convolution 3x3 tiếp tục thực hiện phép tích chập chính xác. Cuối cùng, lớp convolution 1x1 cuối cùng sẽ tăng số chiều trở lại cho đầu ra của khối.

Khối Bottleneck giúp giảm số lượng tham số và tính toán so với Residual Block thông thường, giúp tăng tốc quá trình huấn luyện của mạng ResNet và cho phép xây dựng các mô hình sâu hơn và phức tạp hơn.



Cấu trúc khối Bottleneck

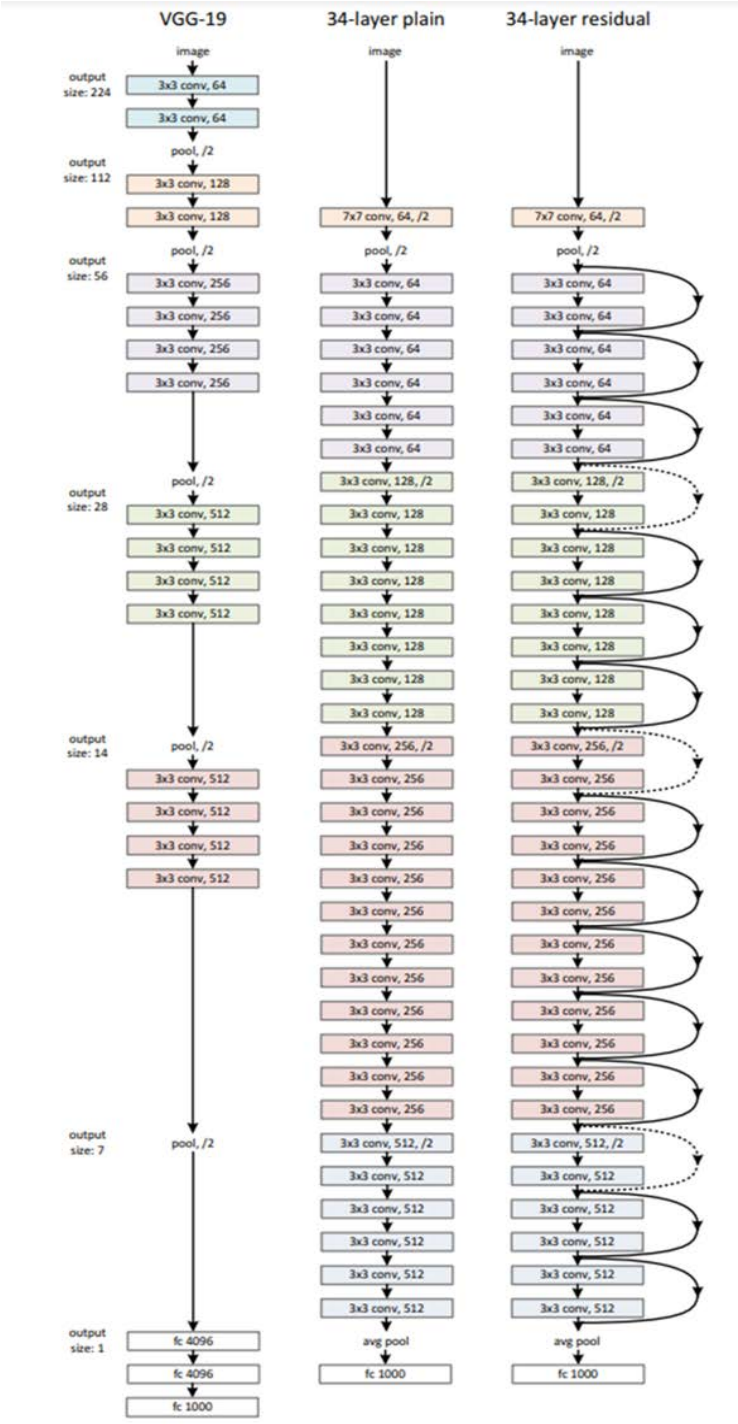
2.2.2. ResNet-50[1]

ResNet-50 có cấu trúc dựa trên mô hình ResNet-34 (gồm 34 lớp trọng số), nhưng có một điểm khác biệt quan trọng. ResNet-50 sử dụng một thiết kế bottleneck cho khối xây dựng. Một khối residual bottleneck sử dụng các convolution 1×1, được gọi là "bottleneck", giúp giảm số lượng tham số và phép nhân ma trận. Điều này cho phép huấn luyện mỗi lớp nhanh hơn đáng kể. Nó sử dụng một stack gồm ba lớp thay vì hai lớp như trong kiến trúc gốc.

Kiến trúc ResNet-50 gồm các thành phần sau:

- Một lớp tích chập 7×7 với kernel size là 64 và stride là 2.
- Một lớp max pooling với stride là 2.
- 9 lớp nữa gồm tích chập 3×3, kernel size là 64, lớp tiếp theo với kernel size là 1×1 và 64 kernel, và lớp thứ ba với kernel size là 1×1 và 256 kernel. Ba lớp này được lặp lại 3 lần.
- 12 lớp nữa với kernel size là 1×1 và 128 kernel, tích chập 3×3 với kernel size là 128, và kernel size là 1×1 và 512 kernel. Ba lớp này được lặp lại 4 lần.
- 18 lớp nữa với kernel size là 1×1 và 256 kernel, tích chập 3×3 với kernel size là 256, và kernel size là 1×1 và 1024 kernel. Ba lớp này được lặp lại 6 lần.
- 9 lớp nữa với kernel size là 1×1 và 512 kernel, tích chập 3×3 với kernel size là 512, và kernel size là 1×1 và 2048 kernel. Ba lớp này được lặp lại 3 lần. (Đến đây, mạng đã có 50 lớp).

- Lớp average pooling, sau đó là một fully connected layer với 1000 node, sử dụng hàm kích hoạt softmax.



Bên trái: mô hình VGG-19 (19.6 tỷ FLOP) để tham chiếu. Ở giữa: một mạng thông thường với 34 lớp tham số (3.6 tỷ FLOP). Bên phải: một mạng residual với 34 lớp tham số (3.6 tỷ FLOP)

2.2.3. ResNet-101

Kiến trúc ResNet-101 gồm các thành phần sau:

- Một lớp tích chập 7×7 với kernel size là 64 và stride là 2.
- Một lớp max pooling với stride là 2.

- 9 lớp nữa gồm tích chập 3×3, kernel size là 64, lớp tiếp theo với kernel size là 1×1 và 64 kernel, và lớp thứ ba với kernel size là 1×1 và 256 kernel. Ba lớp này được lặp lại 3 lần.
- 12 lớp nữa với kernel size là 1×1 và 128 kernel, tích chập 3×3 với kernel size là 128, và kernel size là 1×1 và 512 kernel. Ba lớp này được lặp lại 4 lần.
- 69 lớp nữa với kernel size là 1×1 và 256 kernel, tích chập 3×3 với kernel size là 256, và kernel size là 1×1 và 1024 kernel. Ba lớp này được lặp lại 23 lần.
- 9 lớp nữa với kernel size là 1×1 và 512 kernel, tích chập 3×3 với kernel size là 512, và kernel size là 1×1 và 2048 kernel. Ba lớp này được lặp lại 3 lần. (Đến đây, mạng đã có 50 lớp).
- Lớp average pooling, sau đó là một fully connected layer với 1000 node, sử dụng hàm kích hoạt softmax

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Kiến trúc mạng ResNet101

2.2.4. So sánh ResNet-50 với ResNet-101

ResNet-50 và ResNet-101 đều là các phiên bản của mạng ResNet và chia sẻ cấu trúc chung dựa trên việc sử dụng các khối bottleneck để giảm số lượng tham số và tăng tốc tính toán. Tuy nhiên, chúng vẫn có một số sự khác biệt:

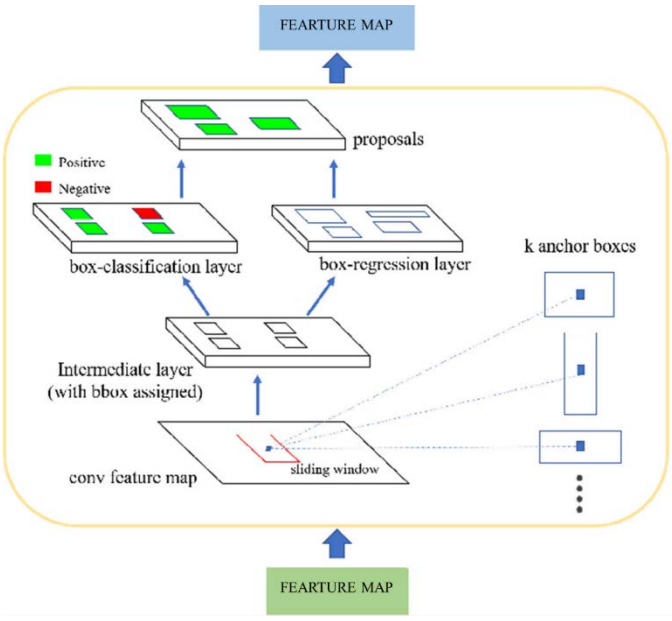
- ***Độ sâu***: ResNet-101 có độ sâu lớn hơn so với ResNet-50. Với 101 lớp, ResNet-101 có khả năng học và biểu diễn các đặc trưng phức tạp hơn.
- ***Hiệu suất chính xác***: Tính chính xác của mạng thường tăng theo độ sâu. Do đó, ResNet-101 có xu hướng có hiệu suất chính xác cao hơn so với ResNet-50. Tuy nhiên, sự khác biệt này không lớn và phụ thuộc vào tập dữ liệu và yêu cầu của nhiệm vụ cụ thể.
- ***Độ phức tạp tính toán***: ResNet-101 có số lượng lớp và tham số lớn hơn so với ResNet-50, dẫn đến độ phức tạp tính toán cao hơn. Việc huấn luyện và triển khai ResNet-101 có thể yêu cầu nhiều tài nguyên tính toán và bộ nhớ hơn so với ResNet-50.

- ***Độ tin cậy và khả năng học***: Độ sâu lớn của ResNet-101 có thể giúp nâng cao khả năng học và khả năng biểu diễn của mạng. Tuy nhiên, nếu không có đủ dữ liệu huấn luyện và tài nguyên tính toán, việc sử dụng một mạng sâu như ResNet-101 có thể dẫn đến hiện tượng overfitting và khó khăn trong huấn luyện.
- ***Thời gian huấn luyện***: Do độ sâu và độ phức tạp tính toán cao hơn, ResNet-101 có thể yêu cầu thời gian huấn luyện lâu hơn so với ResNet-50. Việc lựa chọn giữa hai kiến trúc phụ thuộc vào sự cân nhắc giữa hiệu suất và tài nguyên tính toán.

2.3 Region Proposal Network (RPN)

Region Proposal Network (RPN) là một phần quan trọng trong mô hình Mask R-CNN. Nó được sử dụng để tạo ra các vùng đề xuất (region proposals) có khả năng chứa đối tượng trong ảnh đầu vào.

RPN học cách tạo ra các vùng đề xuất từ các feature map. Một mạng RPN điển hình có thể được thể hiện như hình dưới đây:



Cấu trúc mạng RPN

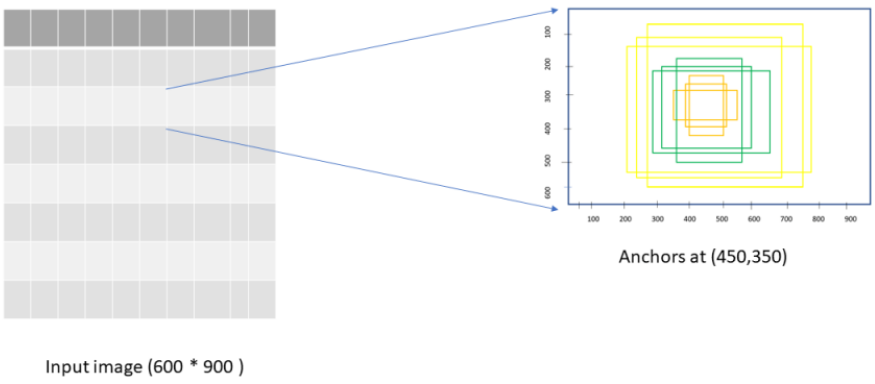
Các bước thực hiện của mạng RPN như sau:

Bước 1: Ảnh đầu vào được đưa qua Mạng Neural tích chập (Convolutional Neural Network) và ở lớp cuối cùng của nó cho ra các feature map như đầu ra.

Bước 2: Trong bước này, một cửa sổ trượt được chạy qua các feature map thu được từ bước trước đó. Kích thước của cửa sổ trượt là n*n (ở đây là 3x3). Đối với mỗi cửa sổ trượt, một tập hợp các anchor được tạo ra với 3 tỷ lệ khác nhau (1:1, 1:2, 2:1) và 3 tỷ lệ khác nhau (128, 256 và 512) như được hiển thị dưới đây.

Với 3 tỷ lệ khác nhau và 3 tỷ lệ khác nhau, tổng cộng có 9 đề xuất có thể cho mỗi pixel. Và tổng số Anchor Boxes với feature map có kích thước $W \times H$ và k anchor cho mỗi vị trí của feature map, có thể được biểu diễn là $W \times H \times k$.

Đồ thị sau đây cho thấy 9 anchors tại vị trí (450, 350) của một ảnh với kích thước (600, 900).



Trong hình trên, ba màu sắc biểu thị ba tỷ lệ hoặc kích thước: 128×128, 256×256, 512×512. Hãy tập trung vào các hộp/anchors màu nâu (các hộp nằm ở giữa trong hình trên). Ba hộp có tỷ lệ chiều cao và chiều rộng lần lượt là 1:1, 1:2 và 2:1.

Bây giờ chúng ta có 9 Anchor boxes cho mỗi vị trí trên feature map. Nhưng có thể có nhiều hộp không chứa bất kỳ đối tượng nào. Vì vậy, mô hình cần học cách xác định hộp anchor nào có thể chứa đối tượng của chúng ta. Hộp anchor chứa đối tượng của chúng ta có thể được phân loại là foreground và phần còn lại sẽ là background. Đồng thời, mô hình cần học các giá trị điều chỉnh cho các hộp foreground để điều chỉnh phù hợp với các đối tượng. Và điều này đưa chúng ta đến bước tiếp theo.

Bước 3: Thực hiện việc xác định vị trí và phân loại hộp anchor bởi lớp Bounding Box Regressor và lớp Bounding Box Classifier.

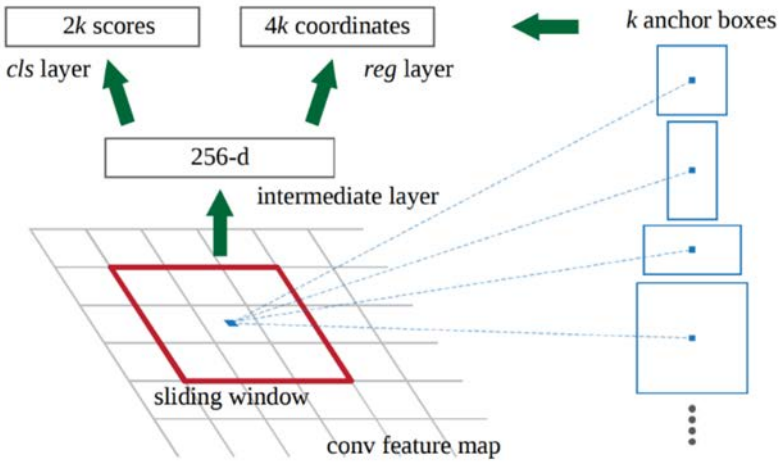
Lớp Bounding Box Classifier tính điểm IoU (Intersection over Union) của Ground Truth Box với các hộp anchor và phân loại hộp anchor là Foreground hoặc Background với xác suất nhất định, còn được gọi là điểm đáng tin cậy (objectness score).

Lớp Bounding Box Regressor học các giá trị điều chỉnh (hoặc sai khác) cho các giá trị x, y, w, h so với Ground Truth Box cho các hộp anchor đã được phân loại là Foreground, trong đó (x, y) là tâm của hộp, w và h là chiều rộng và chiều cao.

- Anchor:

Tại mỗi vị trí cửa sổ trượt, chúng tôi đồng thời dự đoán nhiều đề xuất khu vực, với số lượng đề xuất tối đa có thể cho mỗi vị trí được đặt tên là k. Vì vậy, lớp reg có 4k đầu ra mã hóa các tọa độ của k hộp, và lớp cls có 2k điểm số ước tính xác suất của vật thể hoặc không phải vật thể cho mỗi đề xuất. K các đề xuất được tham số hóa liên quan đến k hộp tham chiếu, mà chúng tôi gọi là "anchor". Một anchor được căn giữa tại vị trí trượt cửa sổ cần xét, và được liên kết với một tỷ lệ và tỷ lệ khung hình, tạo thành các hộp neo (anchor box). Theo mặc định, chúng tôi sử dụng 3 tỷ lệ và 3 khung hình, cho k = 9 anchors tại mỗi vị trí

trượt. Đối với một feature map tích chập có kích thước $W \times H$ (thông thường khoảng 2.400), tổng số anchor là $W \times H \times k$.



Region Proposal Network (RPN)

- Lớp trung gian:

Lớp trung gian của RPN là một lớp tích chập n × n nhận các feature map từ lớp tích chập chung cuối cùng của backbone làm đầu vào. Mục đích của lớp này là tạo ra một tập hợp các feature map, được sử dụng để đề xuất các bounding box cho đối tượng.

Lớp trung gian có một số lượng kênh đầu ra cố định. Mỗi vị trí trên feature map đầu ra tương ứng với một vị trí cụ thể trên feature map đầu vào. Tại mỗi vị trí, lớp trung gian tạo ra một tập hợp gồm k anchor box, với các kích thước và tỷ lệ khác nhau.

Đối với mỗi anchor box, lớp trung gian (intermediate layer) tạo ra một vector đặc trưng mã hóa vị trí không gian và diện mạo của đề xuất vùng chứa đối tượng. Vector đặc trưng này sau đó được sử dụng bởi các lớp hồi quy hộp (box-regression) và phân loại hộp (box-classification) để dự đoán tập hợp cuối cùng của các đề xuất vùng chứa đối tượng.

- Box-regression layer (reg):

Lớp hồi quy hộp (box-regression) của RPN nhận đầu vào là feature map đầu ra từ lớp trung gian và dự đoán các giá trị thay đổi (offsets) cho mỗi anchor box. Cụ thể, đối với mỗi anchor box, lớp hồi quy (reg layer) cho đầu ra là bốn số xác định sự chênh lệch (offset) giữa anchor box và bounding box thực tế (ground-truth bounding box) của đối tượng.[5]

Đối với việc dự đoán khung giới hạn (bounding box regression), chúng ta sử dụng các tham số hóa cho 4 tọa độ như sau:

$$t_x = (x - x_a)/w_a, \quad t_y = (y - y_a)/h_{a,}$$
$$t_w = log(w/w_a), \quad t_h = log\big(h/h_{a,}\big),$$
$$t_x^* = (x^* - x_a)/w_a, \quad t_y^* = (y^* - y_a)/h_{a,}$$
$$t_w^* = log(w^*/w_a), \quad t_h^* = log\big(h^*/h_{a,}\big)$$

Trong đó x, y, w và h đều là tọa độ trung tâm và chiều rộng, chiều cao của khung giới hạn. Biến x, xa và x^* lần lượt là tọa độ của khung dự đoán, khung neo và khung thực tế (tương tự cho y, w, h). [5]

- Box-classification layer (cls):

Lớp dự đoán lớp hộp (box-classification layer) của RPN sử dụng các feature map đầu ra từ lớp trung gian và dự đoán xác suất của mỗi anchor box chứa một đối tượng quan tâm hay là nền.

Đối với mỗi anchor box, lớp cls sẽ đưa ra hai số thể hiện xác suất của anchor box chứa một đối tượng và xác suất của anchor box là nền. Các xác suất này được sử dụng để phân loại các anchor box thành đề xuất đối tượng hoặc đề xuất nền và loại bỏ các đề xuất có điểm số thấp.

Chúng ta gán một nhãn lớp nhị phân (có chứa đối tượng hoặc không) cho mỗi anchor box. Chúng ta gán nhãn dương cho hai loại anchor box: (i) anchor box/ các anchor box có độ chồng chéo (IoU) cao nhất với một hộp thực tế, hoặc (ii) một anchor box có độ chồng chéo lớn hơn 0,7 với bất kỳ hộp thực tế nào. Chúng ta gán nhãn tiêu cực cho một anchor box không phải là anchor box dương nếu tỷ lệ IoU của nó thấp hơn 0,3 đối với tất cả các hộp thực tế. Các anchor box không phải là anchor box dương hoặc tiêu cực sẽ không đóng góp vào mục tiêu huấn luyện. [5]

Để tính toán cho các kích thước khác nhau, chúng ta sử dụng một tập hợp k bộ phân loại khung giới hạn (bounding-box regressors). Mỗi bộ phân loại đảm nhiệm một tỷ lệ và một khía cạnh, và k bộ phân loại không chia sẻ trọng số.

Hàm mất mát của chúng ta cho một ảnh được định nghĩa như sau:

$$L(\{p_i\},\{t_i\}) = \left(\frac{1}{N_{cls}}\right)\sum_i L_{cls}(p_i, p_i^*) + \lambda\left(\frac{1}{N_{reg}}\right)\sum_i p_i^* L_{reg}(t_i, t_i^*)$$

Trong đó:

i là chỉ số của các anchor box trong ảnh

p_i là xác suất của anchor box i chứa một đối tượng quan tâm

p_i^* là nhãn lớp thực tế của anchor box i (1 cho các anchor box tích cực và 0 cho các anchor box tiêu cực)

N_{cls} là số lượng anchor box được lấy mẫu cho phân loại

N_{reg} là số lượng anchor box được lấy mẫu cho hồi quy khung giới hạn

$t_i$là một vectơ đại diện cho dự đoán hồi quy của anchor box i

t_i^* là một vectơ đại diện cho hồi quy khung giới hạn thực tế của anchor box i

L_{cls} là hàm mất mát phân loại (chẵn cho anchor box tích cực và lẻ cho anchor box tiêu cực)

L_{reg} là hàm mất mát hồi quy khung giới hạn (chẵn cho hệ số điều chỉnh dịch và lẻ cho hệ số điều chỉnh tỷ lệ)

λ = 10 (mặc định) là một hệ số cân bằng giữa hai thành phần của hàm mất mát.

2.4 RoI Align

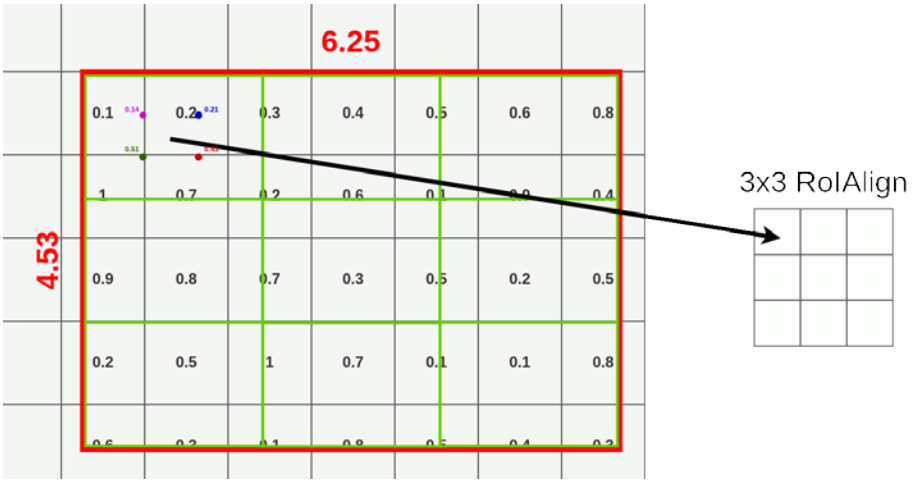
Lớp ROI Align nhận đầu vào là các feature map các đề xuất đối tượng được tạo bởi RPN, và tạo ra các feature map có kích thước cố định cho mỗi đề xuất đối tượng.

RoIAlign loại bỏ harsh quantization của RoIPool[4], đảm bảo các đặc trưng được trích xuất phù hợp với việc tạo mặt nạ (mask) bằng cách đảm bảo độ chính xác đến sub-pixel.

Để chuyển đổi đầu vào thành một feature map có kích thước cố định, RoIAlign trước tiên chọn một số lượng cố định các bin trên các chiều rộng và chiều cao của mỗi vùng quan tâm (Region of Interest - RoI). Kích thước của các bin này được chọn dựa trên kích thước đầu ra mong muốn của feature map và độ phân giải của feature map đầu vào.

Tiếp theo, đối với mỗi bin, RoIAlign tính toán một phép nội suy tuyến tính của các giá trị feature map tại bốn vị trí không gian gần nhất. Những giá trị nội suy này sau đó được tổng hợp bằng cách sử dụng max pooling để tạo ra một feature map đầu ra có kích thước cố định cho mỗi RoI.

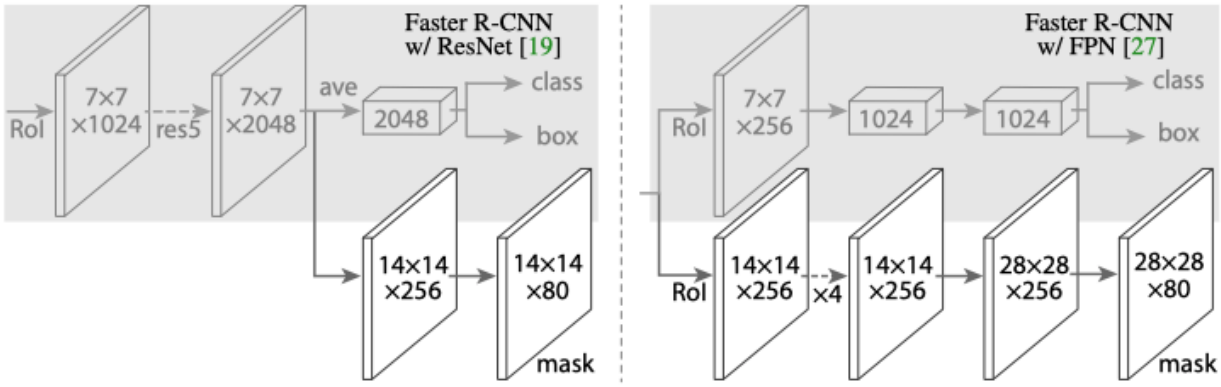
Cuối cùng, feature map đầu ra cho mỗi RoI được đưa qua một mạng fully connected network (FCN) để tạo ra một vector đặc trưng có kích thước cố định.



Kết quả sau quá trình tổng hợp RoIAlign

2.5 Head Branch

Cấu trúc Head Branch trong Mask R-CNN bao gồm hai nhánh chính: một nhánh để phân loại các đối tượng và một nhánh để dự đoán các mặt nạ cho các đối tượng đã được phân loại.



Cấu trúc Head Branch [2]

2.5.1. Object detection Branch

Object detection Branch được sử dụng để xử lý và dự đoán các thông tin liên quan đến bounding box, bao gồm phân loại lớp và điều chỉnh tọa độ.

Object detection Branch thực hiện hai nhiệm vụ chính:

- Box Classification: Tầng phân loại bounding box. Nhiệm vụ của tầng này là phân loại lớp (class) của các bounding box đề xuất. Tầng này thường sử dụng các lớp Fully Connected (FC) và các lớp kích hoạt softmax để tính toán xác suất cho mỗi lớp có thể có. Kết quả là một vector xác suất với số chiều tương ứng với số lớp cần phân loại.
- Bounding box: Tầng hồi quy bounding box. Nhiệm vụ của tầng này là điều chỉnh tọa độ, kích thước và hướng của các bounding box đề xuất để chính xác khớp với các đối tượng trong ảnh. Tầng này thường sử dụng các lớp Fully Connected (FC) để dự đoán các thông số tương ứng với tọa độ, kích thước và hướng của bounding box.

Object detection Branch cung cấp thông tin về phân loại lớp và điều chỉnh tọa độ của bounding box trong quá trình phát hiện và nhận dạng đối tượng trong mô hình Mask R-CNN.

2.5.2. Mask Branch

Mask Branch là một phần của mô hình Mask R-CNN, được sử dụng để dự đoán và tạo ra các mặt nạ theo từng pixel (pixel-wise) cho các đề xuất bounding box trong ảnh. Mask Branch giúp định vị và phân đoạn các đối tượng trong ảnh.

Cấu trúc chi tiết của Mask Branch trong mô hình Mask R-CNN bao gồm một số lớp quan trọng sau:

- Convolutional Layers: Feature map đầu ra từ tầng RoI Align được truyền qua một chuỗi các lớp convolutional. Số lượng và kích thước của các lớp convolutional có thể thay đổi tùy thuộc vào cấu trúc cụ thể của mô hình. Mỗi lớp convolutional áp dụng một bộ lọc (filter) để trích xuất đặc trưng từ feature map. Bộ lọc này có thể có kích thước kernel nhỏ như 3x3 hoặc 5x5.

- Activation Function: Sau mỗi lớp convolutional, một hàm kích hoạt (activation function) thường được áp dụng để tạo ra đầu ra phi tuyến tính từ đầu vào của lớp. Hàm kích hoạt phổ biến như ReLU (Rectified Linear Unit) thường được sử dụng.
- Upsampling: Sau khi qua chuỗi các lớp convolutional, kích thước của feature map thường được tăng lên bằng cách sử dụng lớp Upsampling. Lớp Upsampling thực hiện việc mở rộng feature map bằng cách sao chép các giá trị từ các vị trí gần nhau. Điều này giúp khôi phục kích thước ban đầu của bounding box.
- Convolutional Layer cuối cùng và Sigmoid Activation: Cuối cùng, một lớp convolutional cuối cùng được sử dụng để dự đoán mặt nạ pixel-wise cho mỗi pixel trong bounding box. Lớp này thường có kích thước kernel nhỏ và số lượng filter tương ứng với số lớp cần phân loại (ví dụ: số lượng lớp đối tượng). Hàm kích hoạt sigmoid được áp dụng lên đầu ra của lớp convolutional cuối cùng để đưa ra dự đoán xác suất cho mỗi pixel. Giá trị gần 1 thể hiện mặt nạ của đối tượng, trong khi giá trị gần 0 thể hiện mặt nạ của nền hoặc không có đối tượng.

Với k class sẽ có k mặt nạ được tạo ra, việc lựa chọn mặt nạ phù hợp được phụ thuộc vào Classification branch.

Cấu trúc chi tiết của Mask Branch có thể thay đổi tùy thuộc vào mô hình cụ thể và yêu cầu của bài toán. Các lớp và hàm kích hoạt được sử dụng có thể thay đổi và tinh chỉnh để đạt được hiệu suất tốt cho phân đoạn mặt nạ trong mô hình Mask R-CNN.

2.6 Loss function[3]

Loss function của Mask R-CNN là sự kết hợp của 3 loss: classification, localization và segmentation mask:

$$L_{all} = L_{cls} + L_{box} + L_{mask} \quad (1)$$

Trong đó, L_{all} đại diện cho hàm mất tổng thể của mô hình; L_{cls} là mất mát phân loại của hôm dự đoán; L_{box} là mất mát hồi quy của hộp dự đoán; L_{mask} là mất mát trung bình của hàm nhị phân entropy chéo.

$$L_{cls}(p_i, p_i^*) = -\text{lb}[p_i p_i^* + (1 - p_i)(1 - p_i^*)] \quad (2)$$

Trong đó p_i là xác suất dự đoán rằng điểm neo i là mục tiêu; p_i^* là giá trị dự đoán của nhãn diện tích thực tế tương ứng; lb đại diện cho hàm mất mát log.

$$L_{box}(t_i, t_i^*) = R(t_i - t_i^*) \quad (3)$$

Trong đó R đại diện cho hàm mất mát robust, t_i đại diện cho 4 vector tọa độ được tham số hóa của khung dự đoán; t_i^*đại diện cho vector tọa độ tương ứng với biên của khu vực thực tế.

$$L_{mask} = -\frac{1}{x}\sum_i \{[x_i^* lgp(x_i) - (1 - x_i^*) lg(1 - p(x_i))]\} \quad (4)$$

Trong đó, x biểu thị số lượng pixel, x_i^* biểu thị nhãn danh mục chứa pixel và $p(x_i)$ biểu thị xác suất của danh mục dự đoán x_i.

3. Thiết kế, xây dựng mô hình

3.1 Bộ dữ liệu

Bộ dữ liệu trong được sử dụng trong nghiên cứu khoa học này được chúng tôi tự sưu tầm và vẽ annotation cho mỗi ảnh. Bộ dữ liệu bao gồm 7 loài động vật: hổ(Panthera tigris), voi (Elephas maximus), gấu (Ursidae), rùa biển (Chelonioidea), khỉ vàng (Macaca mulatta), rái cá thường (Lutra lutra) và hươu sao (Cervus nippon). Tất cả các ảnh đều được chụp trong môi trường tự nhiên.

Một số ảnh trong bộ dữ liệu

Bộ dữ liệu này được chia thành ba phần: 700 ảnh để huấn luyện, 210 ảnh để đánh giá và 70 ảnh để kiểm tra. Việc chia bộ dữ liệu thành các tập này giúp đảm bảo rằng mô hình học sâu được huấn luyện trên đủ dữ liệu để đạt được độ chính xác tốt và đánh giá được hiệu quả của mô hình trên các dữ liệu mới.

Mỗi ảnh trong bộ dữ liệu được cung cấp với thông tin chi tiết về loài động vật trong ảnh, kích thước của ảnh và vị trí của các đối tượng động vật trên ảnh. Thông tin này được lưu trữ trong tập tin annotation và được đánh dấu bằng ID tương ứng với mỗi ảnh. Annotation được tạo cho mỗi ảnh bằng tool Roboflow. Tập tin annotation này được lưu dưới dạng COCO-style json.

Cách vẽ annotation

3.2 Mô hình

Mô hình trong nghiên cứu này được xây dựng với thư viện Tensorflow 1.14.0 và Keras 2.2.4.

Mô hình được xây dựng dựa theo bài báo của Mask RCNN, nhưng có vài trường hợp chúng tôi đã đơn giản hóa mã và tổng quát hóa. Đây là một số khác biệt mà chúng tôi nhận thấy:

- ***Thay đổi kích thước hình ảnh***: Để hỗ trợ huấn luyện nhiều ảnh trong cùng một batch, chúng tôi thay đổi kích thước tất cả các hình ảnh sao cho chúng có cùng kích thước. Trong đó, tỷ lệ khung hình được giữ nguyên, vì vậy nếu một ảnh không phải là hình vuông, chúng tôi sẽ lấp đầy nó bằng các số 0. Trong bài báo, việc thay đổi kích thước được thực hiện sao cho cạnh nhỏ nhất là 800px và cạnh lớn nhất bị cắt bớt thành 1000px.
- ***Bounding box***: Một số bộ dữ liệu cung cấp các bounding box và một số chỉ cung cấp các mask. Để hỗ trợ việc huấn luyện trên nhiều bộ dữ liệu, chúng tôi đã chọn bỏ qua các bounding box được cung cấp bởi bộ dữ liệu và thay vào đó tạo ra chúng theo cách linh hoạt hơn. Chúng tôi chọn hộp nhỏ nhất bao quanh tất cả các điểm ảnh của mask làm bounding box. Điều này đơn giản hóa quá trình cài đặt và cũng làm việc áp dụng các augmentation mà nếu không sẽ khó áp dụng cho các bounding box, chẳng hạn như xoay hình ảnh, trở nên dễ dàng hơn.
- ***Learning rate***: Trong bài báo, learning rate là 0,02, nhưng chúng tôi thấy rằng nó quá cao, đặc biệt là khi sử dụng kích thước batch nhỏ. Điều này có thể liên quan đến sự khác biệt giữa cách tính gradient của Caffe và TensorFlow. Chúng tôi thấy rằng learning rate nhỏ hơn sẽ hội tụ nhanh hơn, vì vậy chúng tôi sử dụng learning rate nhỏ hơn.

4. Kết quả nghiên cứu

Hai model được huấn luyện và thực nghiệm với GTX 1650. Cấu hình huấn luyện được dùng chung như sau:

- Batch size: 7
- Kích thước ảnh: tất cả được thay đổi về 256x256
- Tổng số bước/ epoch: 100
- Tổng số bước validation/ epoch: 30
- Learning rate như trong bảng dưới.

Tuy nhiên, với việc huấn luyện cho model với backbone ResNet-50, chúng tôi sử dụng thêm image augmentation. Image augmentation là quá trình thay đổi, biến đổi ảnh gốc để tạo ra các phiên bản mới với mục đích làm tăng lượng dữ liệu huấn luyện và cải thiện khả năng tổng quát hóa của mô hình học máy. Cụ thể, chúng tôi sử dụng hai phép biến đổi đơn giản là lật ngang hoặc lật dọc ảnh với xác suất 50% sử dụng thư viện imgaug.

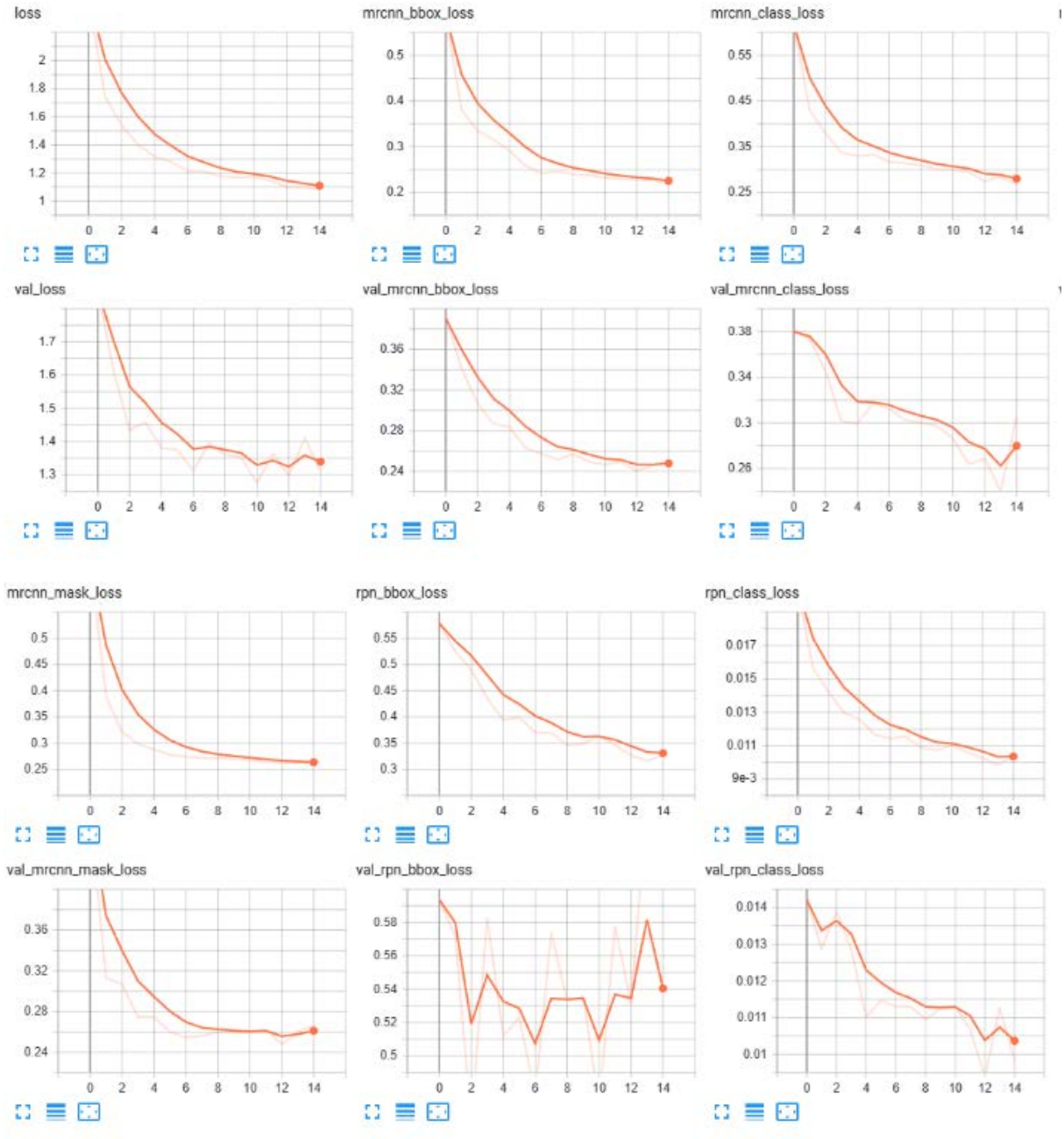
4.1. Sử dụng Backbone model – ResNet-50

Quá trình huấn luyện

	Thời gian huấn luyện	**Learning rate**
Epoch 1-5	16m 20.8s	0.001
Epoch 6-15	36m 2.7s	0.0001

Các kết quả sau khi chạy mô hình:

Kết quả sau khi chạy mô hình



Loss trên tập training và validation qua các epoch

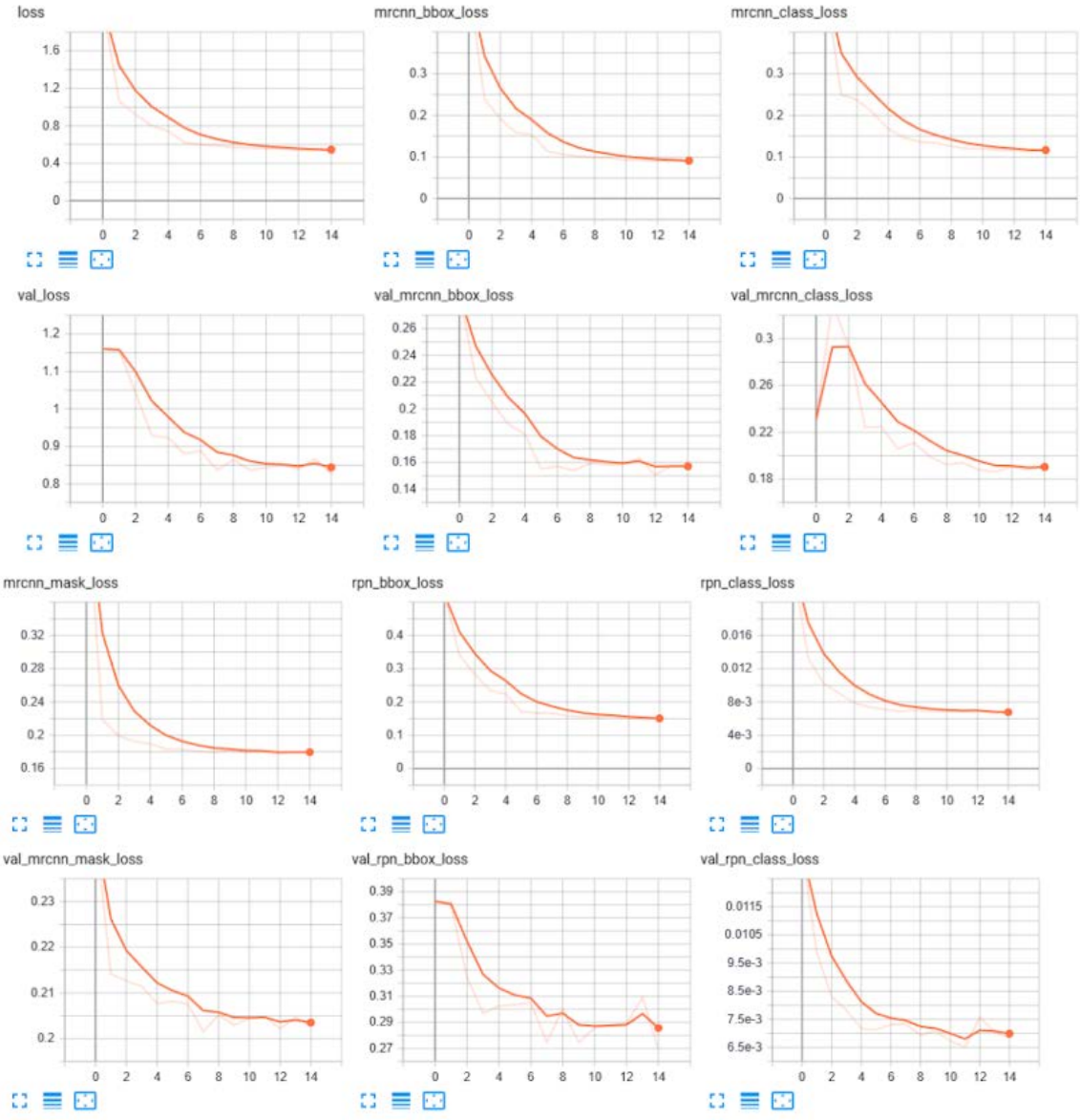
4.2. Sử dụng Backbone model – ResNet-101

Quá trình huấn luyện

	Thời gian huấn luyện	Learning rate
Epoch 1-5	32m 15.8s	0.001
Epoch 6-15	1h 18m 14.5s	0.0001

Một số kết quả dự đoán:

Kết quả sau khi chạy mô hình



Loss trên tập training và validation qua các epoch

5. Thảo luận và đánh giá kết quả

5.1 Các độ đo sử dụng

- **mAP50**

mAP50 (mean Average Precision at IoU=0.5) là chỉ số đánh giá độ chính xác của mô hình trong việc nhận dạng đối tượng. Chỉ số này tính toán độ chính xác trung bình của mô hình trên toàn bộ các lớp đối tượng và các giá trị ngưỡng IoU (Intersection over Union) từ 0.5 đến 1.0.

Trong quá trình đánh giá, mô hình sẽ sử dụng trên các ảnh kiểm tra và tìm kiếm các đối tượng trong ảnh. Sau đó, các đối tượng được đánh dấu bằng cách so sánh vị trí của chúng với các đối tượng đúng trong tập dữ liệu. Nếu giá trị IoU của các đối tượng tìm được vượt qua ngưỡng 0.5, đối tượng được xem là được tìm thấy đúng. Chỉ số mAP50 tính toán độ chính xác của mô hình dựa trên số lượng đối tượng được tìm thấy đúng và số lượng đối tượng thực sự trong tập dữ liệu.

- **DICE**

Dice (Sørensen–Dice index) là chỉ số đánh giá độ chính xác của mô hình trong việc phân đoạn ảnh và tạo ra mask (mặt nạ) cho từng đối tượng.

Chỉ số Dice được tính bằng tỉ lệ giữa diện tích phần chung của mask dự đoán và mask thực tế đối với diện tích tổng của hai mask. Công thức tính chỉ số Dice được thể hiện như sau:

$$Dice = 2 * \frac{|mask\ dự\ đoán \cap mask\ thực\ tế|}{|mask\ dự\ đoán| + |mask\ thực\ tế|}$$

Với một giá trị Dice bằng 1, tức là mask dự đoán trùng hoàn toàn với mask thực tế. Khi giá trị Dice càng gần 0 thì mô hình phân đoạn dự đoán càng kém chất lượng. Chỉ số Dice trong Mask R-CNN được tính trên từng đối tượng và được sử dụng để đánh giá hiệu suất của mô hình trong việc phân đoạn các đối tượng trong ảnh. Đối với mỗi đối tượng, mô hình sẽ tính toán chỉ số Dice và sau đó lấy giá trị trung bình của các chỉ số Dice này để tính toán tổng thể chỉ số Dice của mô hình.

5.2 Đánh giá kết quả

Kết quả dưới đây được thực nghiệm trên GTX 1650 với 1 ảnh/ 1GPU.

Bảng kết quả thực nghiệm với ResNet-50

	Training dataset	Validation dataset
Thời gian chạy	3m 31.4s	1m 6.2s
mAP	77.90%	67.61%
DICE	77.72%	68.40%

Bảng kết quả thực nghiệm với Resnet-101

	Training dataset	Validation dataset
Thời gian chạy	3m 55.1s	1m 11.7s
mAP	90.03%	81.30%
DICE	87.14%	77.83%

Trong quá trình huấn luyện, ResNet-101 đạt được mAP50 là 90.03% và DICE là 87.14%, trong khi ResNet-50 chỉ đạt được mAP50 là 77.90% và DICE là 77.71%. Điều này cho thấy ResNet-101 đạt được kết quả chính xác hơn và tốt hơn trong việc phân đoạn hình ảnh. Trên tập dữ liệu validation, ResNet-101 tiếp tục cho thấy hiệu suất tốt hơn với mAP50 là 81.30% và DICE là 77.83%, so với ResNet-50 với mAP50 là 67.61% và DICE là 68.40%. Điều này chỉ ra rằng ResNet-101 có khả năng tổng quát hóa tốt hơn và không chỉ là overfitting trên tập huấn luyện.

Tuy nhiên, ResNet-101 mất thời gian lâu hơn để huấn luyện và đánh giá so với ResNet-50. Điều này cho thấy ResNet-101 yêu cầu nhiều tài nguyên tính toán hơn, bao gồm cả thời gian và công suất tính toán, so với ResNet-50. Do đó, việc sử dụng ResNet-101 có thể gặp khó khăn trong việc triển khai thực tế, đặc biệt là trên các thiết bị có tài nguyên tính toán hạn chế.

6. Kết luận

Tổng kết lại, nhận diện các loài động vật bằng mô hình học sâu là một lĩnh vực nghiên cứu đầy tiềm năng và có ứng dụng rộng rãi trong bảo tồn động vật hoang dã và các lĩnh vực liên quan đến động vật. Tuy nhiên, như đã đề cập, hiện tại còn tồn tại một số thách thức trong việc sử dụng mô hình học sâu để nhận diện các loài động vật, bao gồm việc thiếu dữ liệu và mô hình không thực sự hiệu quả.

Để giải quyết những thách thức này, chúng ta cần tăng cường dữ liệu và cải tiến mô hình học sâu. Việc sử dụng các kỹ thuật tăng cường dữ liệu như tạo ảnh mới từ ảnh cũ hoặc tăng cường độ tương phản sẽ giúp tăng độ đa dạng và số lượng dữ liệu, từ đó cải thiện hiệu suất của mô hình. Ngoài ra, cần áp dụng các kỹ thuật huấn luyện mô hình hiệu quả hơn để giảm thiểu hiện tượng overfitting và tăng tốc quá trình huấn luyện.

Tổng quan, sự phát triển của mô hình học sâu trong nhận diện các loài động vật là một hướng đi tiềm năng và nó sẽ tiếp tục được nghiên cứu và phát triển trong tương lai để đáp ứng nhu cầu bảo tồn và quản lý động vật hoang dã.

7. Tài liệu tham khảo

[1]. Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, " Deep Residual Learning for Image Recognition", arXiv:1512.03385v1 [cs.CV] 10 Dec 2015.
[2]. Kaiming He, Georgia Gkioxari, Piotr Dollar, Ross Girshick, “Mask R-CNN”, arXiv:1703.06870v3 [cs.CV] 24 Jan 2018.
[3]. Tao Wang, Kunming Zhang, Wu Zhang, Ruiqing Wang, Shengmin Wan, Yuan Rao, Zhaohui Jiang, Lichuan Gu, “Tea picking point detection and location based on Mask-RCNN”, Information Processing In Agriculture 10 (2023) 267– 275.
[4]. Ross Girshick, " Fast R-CNN", arXiv:1504.08083v2 [cs.CV] 27 Sep 2015.
[5]. Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, arXiv:1506.01497v3 [cs.CV] 6 Jan 2016.
[6]. Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation”, arXiv:1311.2524v5 [cs.CV] 22 Oct 2014.
[7]. Alex Berg, Genevieve Patterson and Matteo Ruggero Ronchi: ILSVRC & COCO object detection/segmentation, 2016