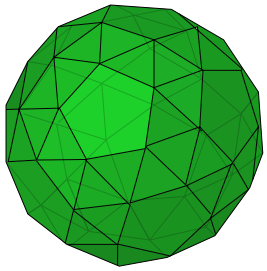
Projects in Mathematics and Applications

MẠNG TÍCH CHẬP VÀ MÔ HÌNH UNET TRONG BÀI TOÁN PHÂN MẢNG HÌNH ẢNH

Nguyễn Hoàng Khang [*] Tạ Quang Khôi [†] Hoàng Nghĩa [‡]
Phan Nguyễn Tuấn Kiệt [§]

Thành phố Hồ Chí Minh, Ngày 9 tháng 8 năm 2019



[*] Trường THPT Chuyên Lê Quý Đôn, Khánh Hòa
[†] Trường THPT Chuyên Lý Tự Trọng, Cần Thơ
[‡] Trường Phổ Thông Năng Khiếu, ĐHQG-HCM
[§] Trường Phổ Thông Năng Khiếu, ĐHQG-HCM

LỜI CẢM ƠN

Chúng tôi xin gửi lời cảm ơn trân trọng nhất đến ban tổ chức trại hè Toán học và Ứng dụng PiMA, trường Đại học Khoa học Tự nhiên - Đại học Quốc gia Thành phố Hồ Chí Minh cũng như các đơn vị tài trợ đã tạo điều kiện tốt nhất về cơ sở vật chất và tinh thần cho chúng tôi trong khuôn khổ hai tuần tại trại. Những trải nghiệm và kiến thức chúng tôi tích lũy được ở đây có ý nghĩa rất lớn đối với bản thân mỗi thành viên trong nhóm, đây chắc chắn sẽ là những hành trang quý báu mà chúng tôi có thể mang theo trên con đường theo đuổi khoa học sau này.

Với lòng biết ơn chân thành nhất, chúng tôi xin gửi lời cảm ơn đến các anh chị hướng dẫn đã nhiệt tình chỉ dạy và dẫn dắt chúng tôi từ những bước đầu tiên trong việc nghiên cứu để chúng tôi có thể có đủ kiến thức và kĩ năng hoàn thành đề tài. Chúng tôi xin đặc biệt cảm ơn anh Cấn Trần Thành Trung, anh Kèn và bạn Hà Phương Uyên vì đã theo sát đề tài cũng như giúp đỡ nhóm từ những vấn đề nhỏ nhặt nhất.

Bên cạnh đó, chúng tôi cũng xin cảm ơn toàn thể các bạn trại sinh của PiMA vì đã cho chúng tôi những kỉ niệm khó quên trong hai tuần cùng học tập và nghiên cứu.

Trong quá trình làm việc, do thời gian chỉnh sửa không nhiều cũng như trình độ của nhóm tác giả có hạn, tuy đã rất cố gắng nhưng chúng tôi hiểu rằng sai sót là rất khó tránh khỏi. Chúng tôi mong nhận được sự thông cảm, góp ý và chia sẻ từ phía bạn đọc để nhóm có thể hoàn thiện đề tài tốt hơn.

Thành phố Hồ Chí Minh, ngày 9 tháng 8 năm 2019.
Nhóm tác giả.

TÓM TẮT NỘI DUNG

Phân mảng hình ảnh là một vấn đề có tính ứng dụng cao và được nghiên cứu nhiều trong khoa học dữ liệu. Bài viết này mô tả những cấu trúc cơ bản của một mạng nơron tích chập và mô hình UNet, một mô hình học sâu với cấu trúc tích chập có khả năng phân mảng hình ảnh tương đối tốt. Đồng thời, chúng tôi cũng trình bày kết quả mà chúng tôi đạt được khi dùng mô hình UNet để phân mảng giữa xe ô tô và nền trên bộ dữ liệu Carvana Image Mask Challenge.

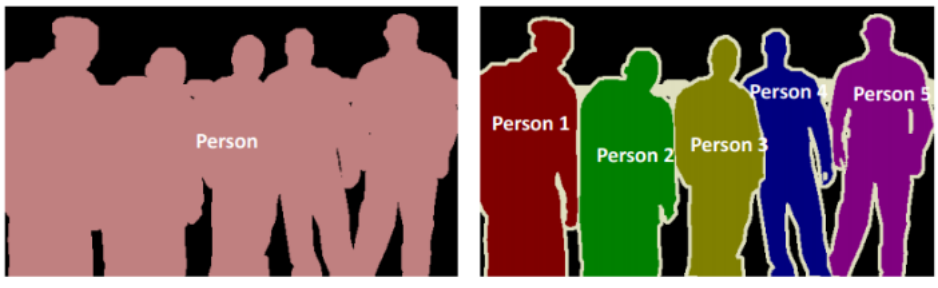
Mục lục

1 BÀI TOÁN PHÂN MẢNG HÌNH ẢNH

Phân mảng hình ảnh giữa đối tượng và nền

Phân mảng hình ảnh (Image segmentation) là bài toán phân dữ liệu trong ảnh thành nhiều mảng nhỏ, mỗi mảng chứa các dữ liệu có cùng một tính chất nào đó. Bài toán được chia làm hai bài toán nhỏ hơn, đó là:

- Phân mảng hình ảnh theo lớp (Semantic segmentation) là bài toán phân mảng hình ảnh thành những lớp (class) khác nhau với các đặc tính khác nhau (Ví dụ: lớp con người, lớp phương tiện giao thông, lớp nhà cửa).
- Phân mảng hình ảnh theo cá thể (Instance segmentation) là bài toán phân mảng từng đối tượng của từng lớp (Ví dụ: ba người khác nhau sẽ được tô màu khác nhau trong output).

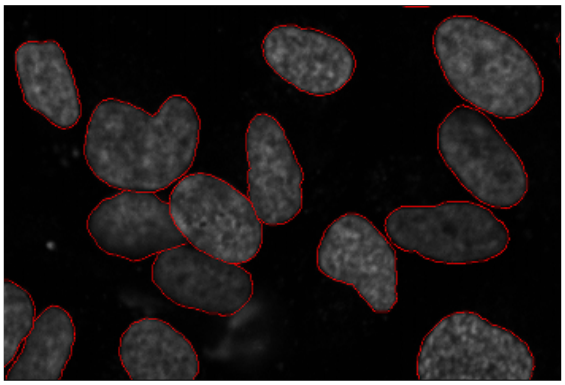


Phân mảng hình ảnh theo lớp và phân mảng hình ảnh theo cá thể

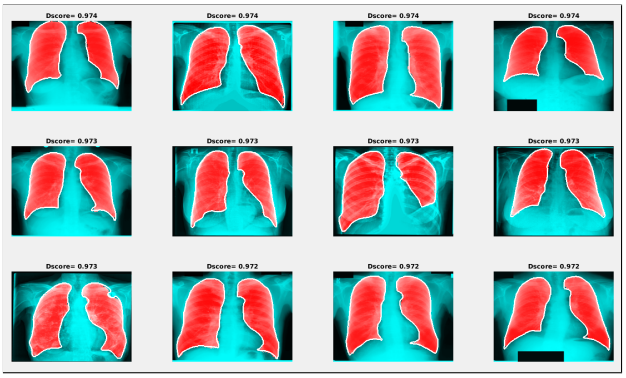
Trong thực tế, ta cần phải lựa chọn phương pháp phân mảng thích hợp cho từng bài toán. Cụ thể như nếu bạn chỉ muốn phân loại các tính chất của các lớp (như trong xe tự lái để tránh vật thể) thì bạn chỉ cần giải quyết bài toán phân mảng hình ảnh theo lớp, còn nếu bạn muốn

phân mảng độc lập các phần tử của các lớp (như trong việc giám sát người trên đường) thì bạn phải giải quyết được bài toán phân mảng hình ảnh theo cá thể.

Các mô hình giải quyết bài toán phân mảng hình ảnh được ứng dụng rất rộng rãi và có tiềm năng lớn trong nhiều ngành như: tự động hóa việc định vị các khối u trong y tế, chẩn đoán bệnh, phương tiện giao thông tự hành hay nhận diện khuôn mặt.



Phát hiện tế bào ung thư

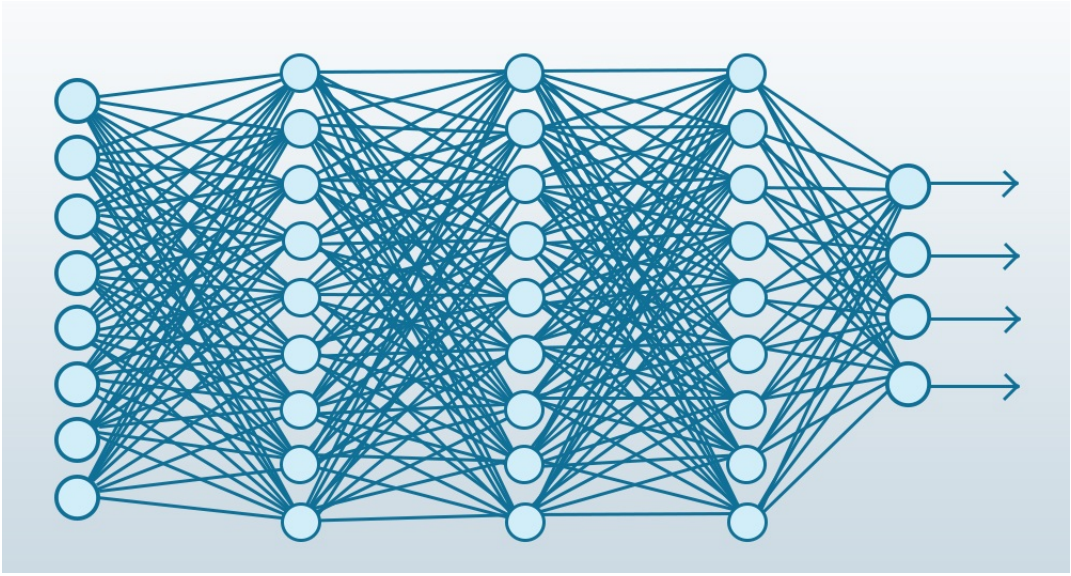


Chẩn đoán bệnh ở phổi

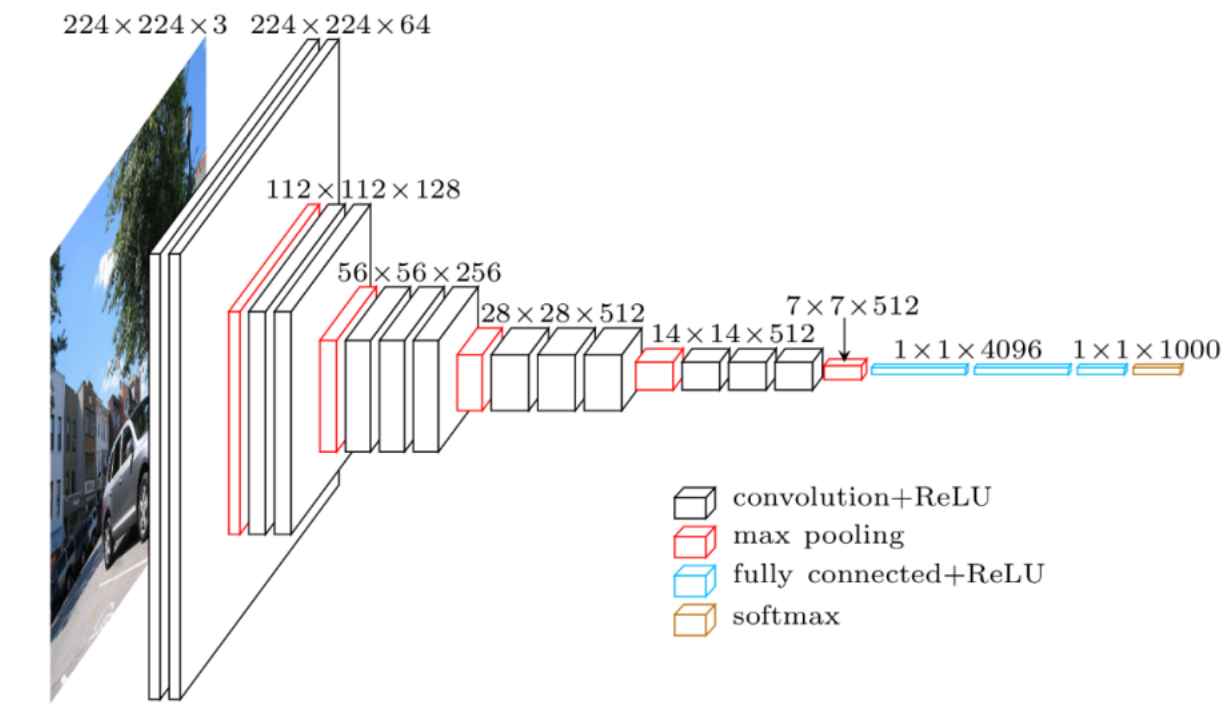
Trong khuôn khổ bài viết này, chúng tôi sẽ tập trung giải quyết bài toán phân mảng hình ảnh theo lớp. Mô hình chúng tôi trình bày để giải quyết bài toán này là mạng nơron tích chập và kiến trúc UNet.

2 MẠNG NƠRON TÍCH CHẬP

Các bài toán phân mảng hình ảnh nói riêng và các bài toán trong lớp thị giác máy tính nói chung đều gây cho ta một số khó khăn nhất định để giải quyết. Một trong những khó khăn tiêu biểu mà ta gặp phải là độ lớn kích thước của dữ liệu vào và có thể của cả dữ liệu ra. Các mạng nơron đơn giản thường có cấu trúc dựa trên việc tạo lập các lớp ẩn nối toàn bộ các nơron của lớp tính toán trước với lớp tính toán sau.



Hướng tiếp cận này thường không phù hợp với công việc xử lí hình ảnh. Ta dễ thấy nếu chỉ làm công việc như thế thì việc xử lí dữ liệu sẽ rất tốn kém tài nguyên, trong khi đó các đặc trưng của dữ liệu ảnh không được khai thác và tận dụng tốt. Một mô hình máy học có nhiều lợi thế hơn các mạng nơron đơn giản trong công việc xử lí hình ảnh mà ta thường dùng có thể kể đến là mô hình mạng nơron tích chập (CNN - Convolutional Neural Network). Ý tưởng cơ bản của mạng nơron tích chập là thay vì ta thực hiện trực tiếp các tính toán trên các nơron dựa trên mỗi một input đầu vào đơn, ta có thể liên tục nhân tích chập dữ liệu ra của các lớp trước trong cấu trúc mạng để được những ma trận dữ liệu có kích thước nhỏ hơn nhưng vẫn nắm bắt được các đặc trưng của ảnh. Mạng nơron tích chập gồm những lớp cơ bản: Lớp tích chập (Convolutional Layer), lớp pooling, lớp chuẩn hóa (Normalization Layer), lớp liên kết đầy đủ (Fully-connected Layer) và lớp soft-max. Ngoài các lớp cơ bản trên, người ta thường sử dụng thêm các hàm phi tuyến như ReLu hay Sigmoid trong mạng để tăng độ phức tạp cho mô hình, giúp nâng cao khả năng học của máy.



Ví dụ mô hình mạng nơron tích chập

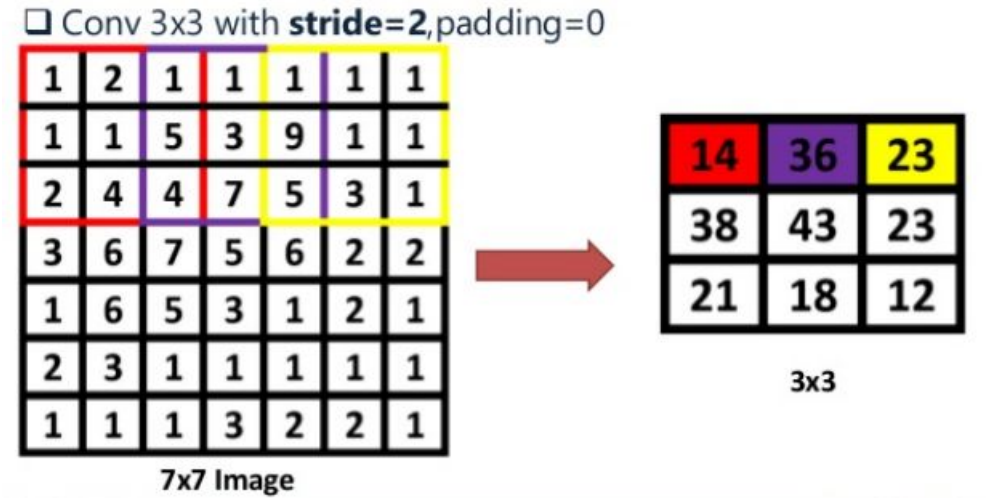
Ta sẽ đi vào việc tìm hiểu cụ thể các lớp cơ bản của mạng nơron tích chập và nhiệm vụ của chúng trong mô hình.

2.1 Lớp tích chập

Lớp tích chập là lớp đặc trưng của cấu trúc mạng CNN. Nhiệm vụ chính của lớp này là lọc các đặc trưng của dữ liệu ảnh đầu vào, những đặc trưng nào cần lọc và cách lọc những đặc trưng ấy ra sao sẽ được mô hình nơron học qua quá trình huấn luyện. Lớp này hoạt động bằng cách thực hiện các phép nhân tích chập giữa các ma trận con trên ma trận pixel của ảnh

đầu vào với các bộ lọc (filter). Sau khi thực hiện xong tất cả các phép nhân tích chập, kết quả sẽ là nhiều ma trận đặc trưng (feature map), mỗi ma trận đặc trưng ứng với một đặc trưng mà mô hình máy muốn học. Ma trận đặc trưng sẽ có kích thước nhỏ hơn ảnh gốc, điều này sẽ giúp cho việc xử lý tiếp theo trở nên dễ dàng hơn và nhanh hơn. Với mỗi lớp tích chập ta có thể quyết định được sẽ có bao nhiêu đặc trưng được tạo ra từ lớp này tùy vào số bộ lọc của lớp. Kích thước của đặc trưng được quyết định bởi một số tham số như sau:

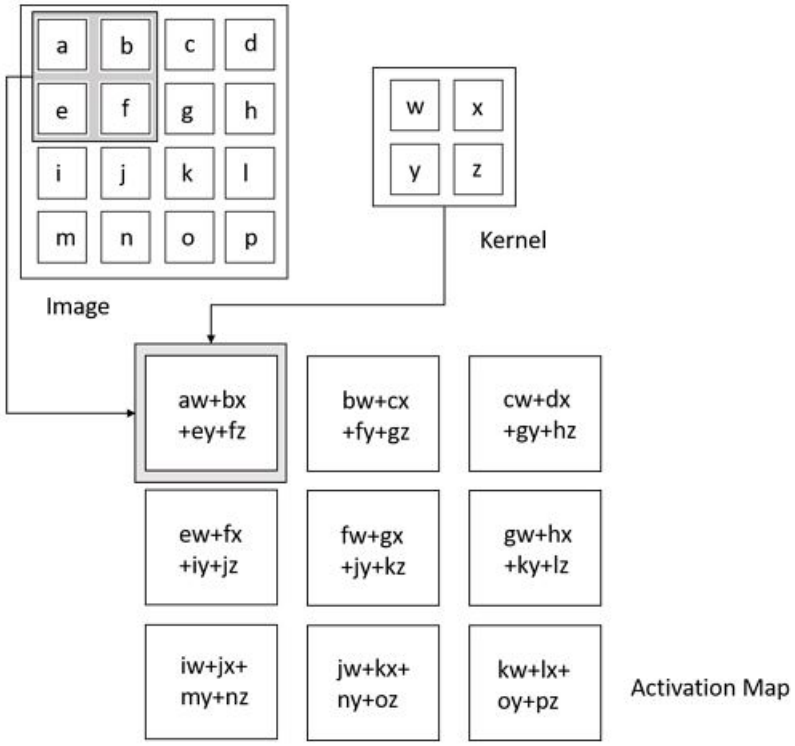
- Bước nhảy (stride): là khoảng cách bộ lọc di chuyển qua một ma trận con khác trên ảnh gốc.
- Lớp đệm (padding): là một số lớp 0 thêm vào viền ma trận ảnh gốc trước khi thực hiện việc tính tích chập.



Ví dụ về tích chập

Nói rõ hơn, lớp tích chập:

- Nhận dữ liệu đầu vào $\mathbf{W}_1 \times \mathbf{H}_1 \times \mathbf{D}_1$.
- Gồm ba tham số không thay đổi:
  - Độ dài cạnh mỗi bộ lọc $\mathbf{F}$
  - Bước nhảy $\mathbf{S}$
  - Số lớp đệm $\mathbf{P}$
- Dữ liệu đầu ra có kích thước $\mathbf{W}_2 \times \mathbf{H}_2 \times \mathbf{D}_2$, với:
  - $\mathbf{W}_2 = (\mathbf{W}_1 - \mathbf{F} + \mathbf{2} \times \mathbf{P})/\mathbf{S} + 1$
  - $\mathbf{H}_2 = (\mathbf{H}_1 - \mathbf{F} + \mathbf{2} \times \mathbf{P})/\mathbf{S} + 1$
  - $\mathbf{D}_2 = \mathbf{D}_1$



Cách tính tích chập

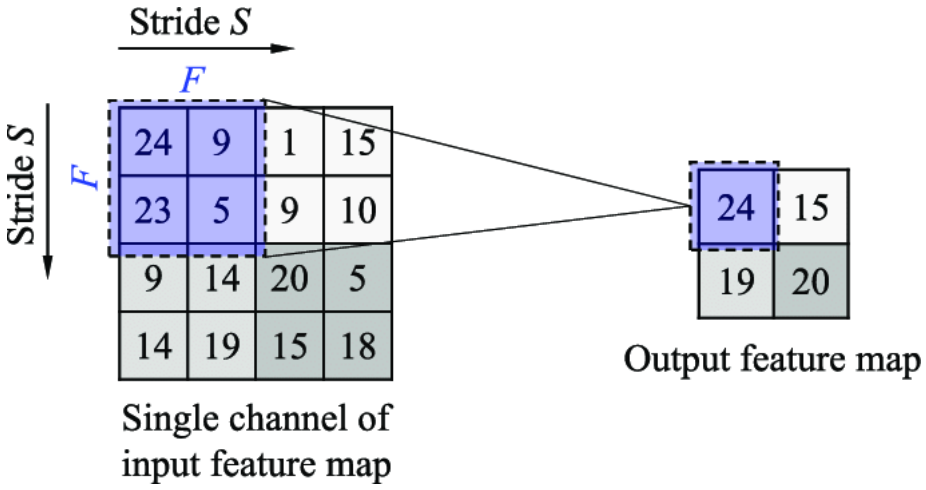
2.2 Lớp pooling

Trên thực tế, người ta thường đưa vào giữa những lớp tích chập trong một mạng nơron tích chập các lớp pooling. Ý nghĩa của việc này là để giảm lượng thông tin cần phải xử lí sau mỗi một số lượng lớp tính toán nhất định, qua đó giảm được số lượng tham số cần phải nhớ và tiết kiệm được thời gian tính toán trong một mạng CNN. Từ đó ta cũng có thể tránh được phần nào việc xảy ra overfit.

Lớp pooling được định nghĩa là một lớp:

- Nhận dữ liệu vào có kích thước $\mathbf{W}_1 \times \mathbf{H}_1 \times \mathbf{D}_1$.
- Gồm hai tham số không thay đổi (hyperparameter) của bộ lọc:
  - Độ dài cạnh mỗi bộ lọc $\mathbf{F}$,
  - Bước nhảy (stride) $\mathbf{S}$.
- Xuất dữ liệu ra có kích thước $\mathbf{W}_2 \times \mathbf{H}_2 \times \mathbf{D}_2$, trong đó:
  - $\mathbf{W}_2 = (\mathbf{W}_1 - \mathbf{F})/\mathbf{S} + 1$,
  - $\mathbf{H}_2 = (\mathbf{H}_1 - \mathbf{F})/\mathbf{S} + 1$,
  - $\mathbf{D}_2 = \mathbf{D}_1$.
- Không chứa bất kì một tham số có thể thay đổi (parameter) nào.

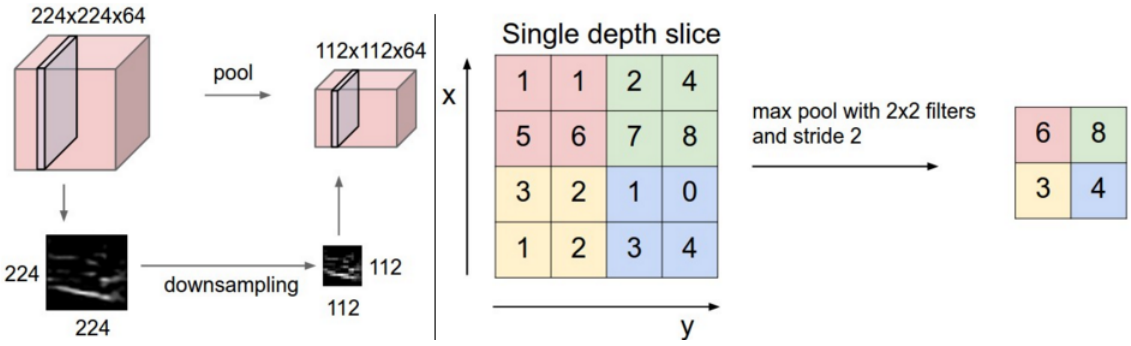
- Thường không sử dụng lớp đệm zero-padding.



Lớp pooling với **F** = 2 và **S=2**

Dạng lớp pooling mà ta thường gặp nhất là lớp pooling với bộ lọc có kích thước 2×2 và bước nhảy bằng 2, tức ta chia hình ảnh ra thành các hình vuông 2×2 và thực hiện phép pool trên mỗi hình vuông nhỏ ấy, từ đó tạo được một bộ dữ liệu mới có kích thước chỉ bằng 25% bộ dữ liệu trước khi qua lớp pooling.

Ta cần chú ý rằng những lớp pooling thường gặp nhất chỉ có bộ lọc kích thước $\mathbf{F} = 2$ và $\mathbf{S} = 2$ hoặc kích thước $\mathbf{F} = 3$ và $\mathbf{S} = 2$ (còn gọi là overlapping pooling), những kích thước khác của bộ lọc thường không hoạt động tốt trên thực tế vì chúng thay đổi quá nhiều hoặc quá ít dữ liệu đầu vào. Phép toán thường cho hiệu quả tốt nhất (và vì thế thường được dùng nhất) ở các lớp pooling là phép toán lấy max các phần tử trong các pixel của mỗi bộ lọc (phiên bản lớp pooling này được gọi là max pooling).

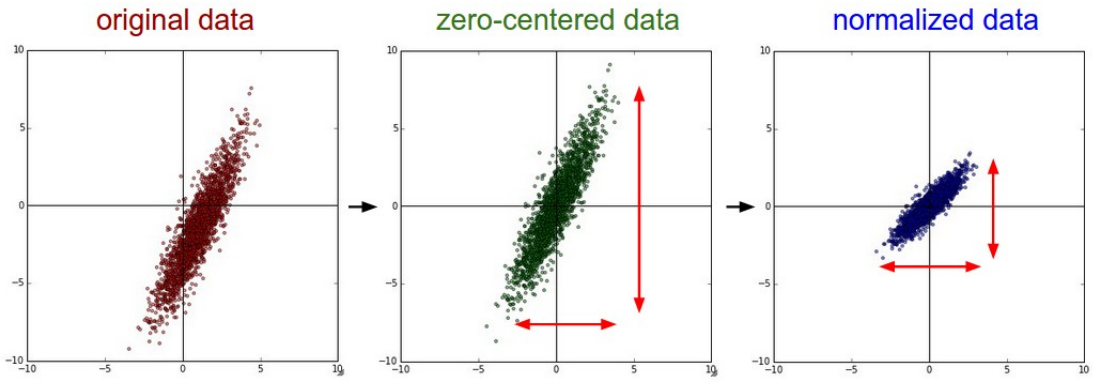


Max pooling với **F** = 2 và **S=2**

2.3 Lớp chuẩn hóa

Lớp chuẩn hóa (Normalization) được định nghĩa là một lớp nhận thông tin vào là dữ liệu thuộc một khoảng giá trị nào đó và cho thông tin ra là chính những dữ liệu đó nhưng ở một khoảng giá trị khác "đẹp"và có những tính chất mà ta mong muốn hơn, việc này được thực hiện bằng cách căn chỉnh và thu gọn hoặc phóng lớn khoảng giá trị của mỗi dữ liệu.

Lớp normalization có ý nghĩa trong việc tăng tính ổn định và tốc độ của một mô hình neural network bất kì, ta có thể hiểu tại sao việc này lại xảy ra qua một ví dụ đơn giản: giả sử ta có một số dữ liệu nằm trong khoảng $[0, 2^{16}]$, vì dữ liệu ở đây có thể nhận những giá trị rất lớn nên việc tính toán trên bộ dữ liệu này có thể sẽ mất rất nhiều thời gian và dẫn đến những sai số (do hệ thống biểu diễn số floating-point của máy tính), hoặc thậm chí không thể tính toán được khi kết quả các phép toán là quá lớn. Thay vì thực hiện trực tiếp các tính toán trên các dữ liệu này, ta có thể thực hiện một bước normalization trước khi tính toán: chia toàn bộ các dữ liệu cho 2^{16}. Khi đó, toàn bộ dữ liệu mà chúng ta có sẽ nằm trong khoảng $[0, 1]$, dễ thấy đây là một khoảng dữ liệu dễ làm việc hơn nhiều so với khoảng dữ liệu cũ.



Có nhiều phép toán ta có thể dùng để tạo ra lớp normalization với những tính chất mà ta mong muốn như: mini-batch mean, mini-batch variance, normalize, scale and shift [1].

2.4 Lớp liên kết đầy đủ (fully connected)

Lớp liên kết đầy đủ ở mạng CNN là một mạng nơron nhân tạo gồm 2 lớp, trong đó các nơron sau được liên kết toàn bộ các nơron ở lớp trước đó bằng các trọng số (được gọi là weight và kí hiệu là w). Để dễ hình dung bạn đọc có thể liên tưởng đến mạng lưới nơron thần kinh ở người, được liên kết với nhau bởi các dây thần kinh, trọng số càng lớn thì sự liên kết của các nơron càng mạnh và ngược lại.

Một mạng nơron nhân tạo gồm ba lớp chính: lớp đầu vào (Input Layer), lớp ẩn (Hidden Layer), và lớp đầu ra (Output Layer).

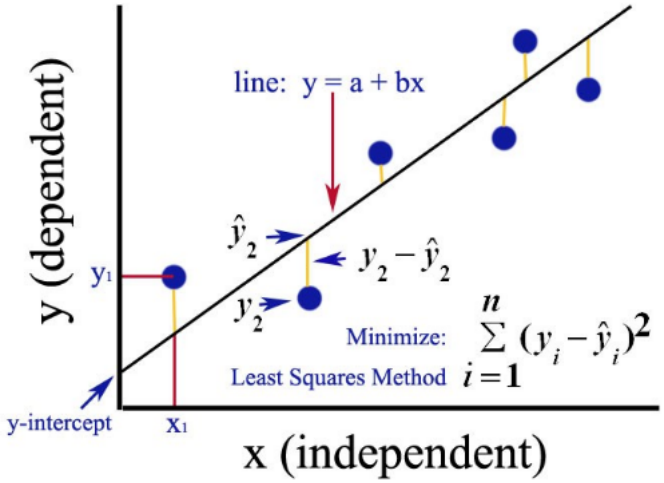
Lớp đầu vào trong mạng nơron nhân tạo là một vector cột chứa các thông số đầu vào (thường là các pixel của một ảnh) được liên kết với các nơron của lớp kế tiếp bởi các trọng số w.

Lớp ẩn là các lớp ở sau lớp input layer và trước lớp output layer. Ở các lớp này mạng nơron của chúng ta sẽ thực hiện quá trình huấn luyện (Trainning) bằng cách ước lượng các tham số W sao cho hàm mất mát (Loss Function) của chúng ta đạt được giá trị nhỏ nhất. Hàm mất mát (kí hiệu là L) là hàm số thể hiện sự chênh lệch giữa giá trị ước lượng của chúng ta (kí hiệu là $\hat{Y}$) với giá trị thật y (giá trị được gán nhãn trước trong data).

1 Xem thêm ở https://arxiv.org/abs/1502.03167

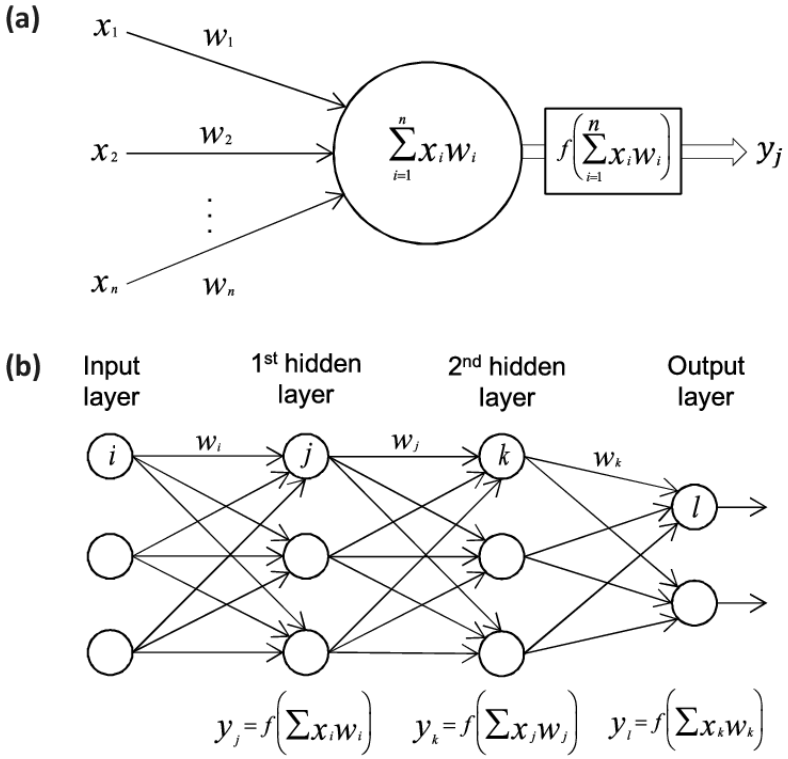
Ví dụ: Một trong những phương pháp xây dựng hàm mất mát trong mạng nơron nhân tạo là phương pháp tổng bình phương nhỏ nhất (Least Square Regression). Công thức của phương pháp tổng bình phương nhỏ nhất như sau:

$$\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = 1.$$



Phương pháp tổng bình phương nhỏ nhất

Mỗi nơron trong cùng một lớp tương ứng với một hàm kích hoạt (Activation Function). Hàm này có đầu vào là tổng có trọng số tín hiệu từ nơron ở lớp trước cộng thêm một sai số nhất định (Bias).

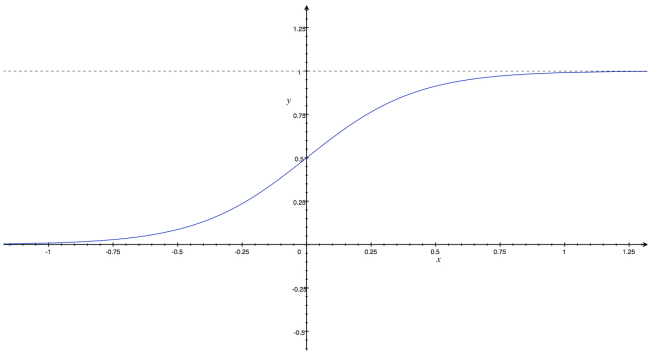


Một mạng lưới nơron đầy đủ

Khi hàm kích hoạt là hàm nhị phân (Binary-Stated), các tín hiệu đầu vào từ các nơron ở lớp trước và cho ra tín hiệu đầu ra chỉ khi đạt đến một ngưỡng (Threshold) nhất định. Ngưỡng là một giá trị quyết định xem nơron đó có được kích hoạt hay không, kí hiệu của ngưỡng là θ.

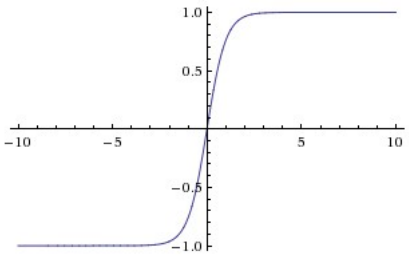
Hàm kích hoạt thông thường được chọn bởi các hàm phi tuyến như:

- Hàm sigmoid: $y = \dfrac{1}{1+e^{-x}}$



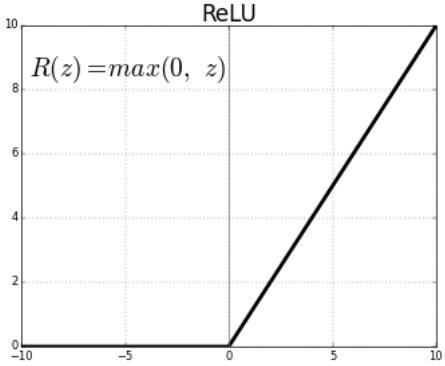
Đồ thị hàm sigmoid

- Hàm tanh: $y = \dfrac{e^{2x}-1}{e^{2x}+1}$



Đồ thị hàm tanh

- Hàm ReLU: $y = \max(0, x)$

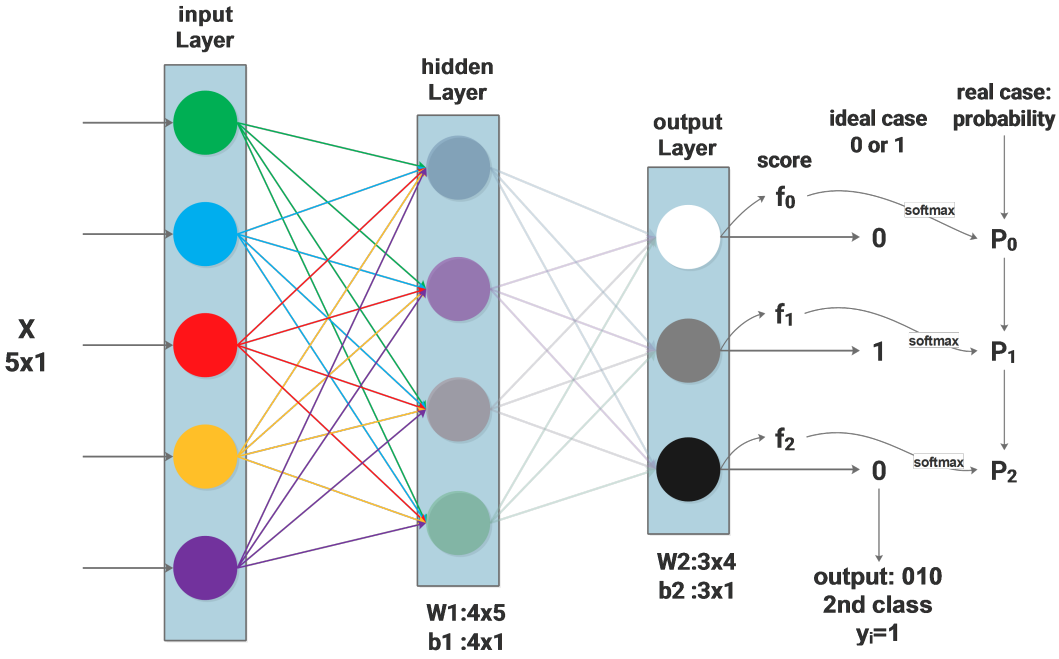


Đồ thị hàm ReLU

- Hàm Softmax là một hàm số $f : \mathbb{R}^K \to (0,1)^K$ được cho bởi công thức:

$$f(z)_j = \frac{e^{Z_j}}{\sum_{k=1}^{K} e^{z_k}}, \text{với } j = 1, 2, \ldots, K.$$

Hàm softmax có khả năng chuyển bộ giá trị số thành một bộ phân bố xác suất, do hàm softmax đồng biến nên các giá trị lớn hơn sẽ ứng với các xác suất lớn hơn. Chính vì thế hàm softmax được ứng dụng khá nhiều trong các bài toán phân loại vì ứng với mỗi xác suất sẽ cho ta biết được khả năng của ảnh hoặc pixel nào đó thuộc vào lớp nào. Chính vì đầu ra cuối cùng là một phân bố xác suất nên hàm mất mát phù hợp để có thể sử dụng với các mạng nơron nhân tạo có lớp softmax chính là hàm pixel-wise cross entropy sẽ được giới thiệu vào phần kế tiếp.



Mô hình sau khi đã qua lớp softmax

Các hàm phi tuyến này được áp dụng vì trên thực tế dữ liệu mà ta có rất ít khi được phân bố tuyến tính nên ta phải áp dụng các hàm phi tuyến này vào để ước lượng (optimize) của ta đạt được hợp lý nhất có thể. Còn một lý do khác mà các hàm kích hoạt là các hàm phi tuyến trên chính là các hàm phi tuyến của ta đã đơn giản hóa số liệu để việc tính toán được dễ dàng hơn. **Ví dụ:** hàm sigmoid đưa dữ liệu đầu ra của ta về khoảng $(0,1)$ có thể mô phỏng các giá trị sác xuất, hàm tanh thì đưa về khoảng $(-1,1)$.

Bên cạnh đó, các hàm phi tuyến trên có các công thức đạo hàm đơn giản giúp giảm độ phức tạp của thuật toán lan truyền ngược (Backpropagation). Mỗi hàm phi tuyến được sử dụng phổ biến trong các dạng bài toán và kiểu dữ liệu riêng. Cụ thể trong bài toán phân mảng hình ảnh thì hàm ReLU được sử dụng khá phổ biến.

Lớp đầu ra Tùy thuộc vào yêu cầu bài toán, lớp đầu ra sẽ có các giá trị khác nhau, nếu là bài toán phân loại nhị phân thì kết quả đầu ra sẽ là giá trị nhị phân. Nếu là bài toán phân loại hình ảnh (Classification) thì lớp đầu ra sẽ nhận các giá trị là một phân phối xác suất.

2.5 Hàm mất mát trong phân mảng hình ảnh

Hàm mất mát (Loss Function) thường được dùng nhất cho các bài toán thuộc lớp thị giác máy tính là hàm pixel-wise cross entropy. Hàm cross entropy cho ta một phương pháp đo độ chênh lệch giữa hai phân bố xác suất có cùng không gian mẫu. Công thức hàm cross entropy đối với hai phân bố p và q là

$$H(p,q) = \mathbb{E}_p[-\log(q)].$$

Nếu p và q đều có không gian mẫu $\mathcal{X}$ rời rạc,, ta sẽ có công thức tính cross entropy là

$$H(p,q) = -\sum_{x\in\mathcal{X}} p(x)\log(q(x)).$$

Xét riêng trong bài toán phân mảng hình ảnh thành hai lớp, ta áp dụng cross entropy đối với mỗi pixel trên từng ảnh khi huấn luyện rồi lấy trung bình các cross entropy ấy làm hàm mất mát cho mô hình. Kí hiệu $p(1,y)$ là xác suất pixel y thuộc đối tượng trên thực tế và $q(1,y)$ là xác suất pixel y thuộc đối tượng do mô hình máy dự đoán.

$$p(1,y) = \begin{cases} 1 & \text{nếu pixel } y \text{ thuộc đối tượng ,} \\ 0 & \text{nếu pixel } y \text{ không thuộc đối tượng.} \end{cases}$$

Gọi $\mathcal{Y}_1$ là tập các pixel thuộc đối tượng cần phân loại, $\mathcal{Y}_2$ là tập các pixel không thuộc đối tượng và $||\mathcal{Y}_1|| + ||\mathcal{Y}_2|| = n$ (n là số lượng pixel cần phân lớp của ảnh). Ta có hàm loss pixel-wise cross entropy

$$L = -\frac{1}{n}\left(\sum_{y\in\mathcal{Y}_1}\log(q(1,y)) + \sum_{y\in\mathcal{Y}_2}\log(1-q(1,y))\right).$$

Sau đó, ta có thể dùng thuật toán lan truyền ngược để tìm cực trị của hàm mất mát này và tối ưu hóa mô hình phân mảng của chúng ta.
Lí giải cho việc tại sao pixel-wise cross entropy lại phù hợp để làm hàm loss trong bài toán này, ta có thể thấy hàm cross entropy có hai tính chất rất thuận lợi:

- Hàm trả về giá trị nhỏ khi hai phân bố giống nhau và có thể trả về giá trị rất lớn nếu hai phân bố rất khác nhau, điều này giúp ta có thể nhanh chóng tối ưu hóa mô hình máy.
- Đạo hàm của hàm cross entropy khá đơn giản và dễ tính, điều này giúp ta tiện lợi trong việc tìm thuật toán tối ưu hóa.

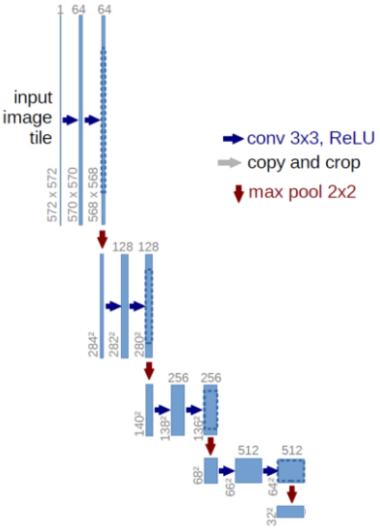
3 MÔ HÌNH UNET

3.1 Đặc điểm của bài toán phân mảng hình ảnh

Bài toán phân mảng hình ảnh khác với các bài toán trong lớp thị giác máy tính khác như phân loại hình ảnh hay nhận diện hình ảnh ở việc output của bài toán không đơn giản chỉ trả lời câu hỏi đối tượng trong ảnh là gì mà còn phải trả lời câu hỏi đối tượng đó nằm chính xác ở vùng nào của hình ảnh. Để có được output trả lời được câu hỏi này, dữ liệu ra của chúng ta phải là một ảnh có kích thước bằng hoặc tương đối với kích thước ảnh cũ. Một mô hình deep learning lí tưởng để giải quyết bài toán phân mảng hình ảnh sẽ là một mô hình neural network nào đó có khả năng sử dụng mạng CNN để học các đặc trưng của hình ảnh và sau đó tái tạo lại ảnh về kích thước ban đầu.
Ta có thể xây dựng một mô hình mạng CNN giải quyết được bài toán phân mảng hình ảnh được từ hai bộ phận: bộ phiên mã (CNN Encoder) và bộ dịch mã (CNN Decoder). Phiên mã có thể được hiểu như bộ phận làm nhiệm vụ nhận thông tin hình ảnh vào và tách chiết các đặc trưng của hình ảnh như hình dáng người, vật rồi phân lớp chúng thành các đối tượng khác nhau. Sau khi các đặc tính của bức ảnh đã được xác định bởi bộ phiên mã, dữ liệu ra của phiên mã sẽ được dịch bởi decoder để tạo ngược lại hình ảnh có kích thước lớn hoàn chỉnh (đã được phân mảng). Mô hình UNet là một mô hình tiêu biểu cho cấu trúc như vậy.

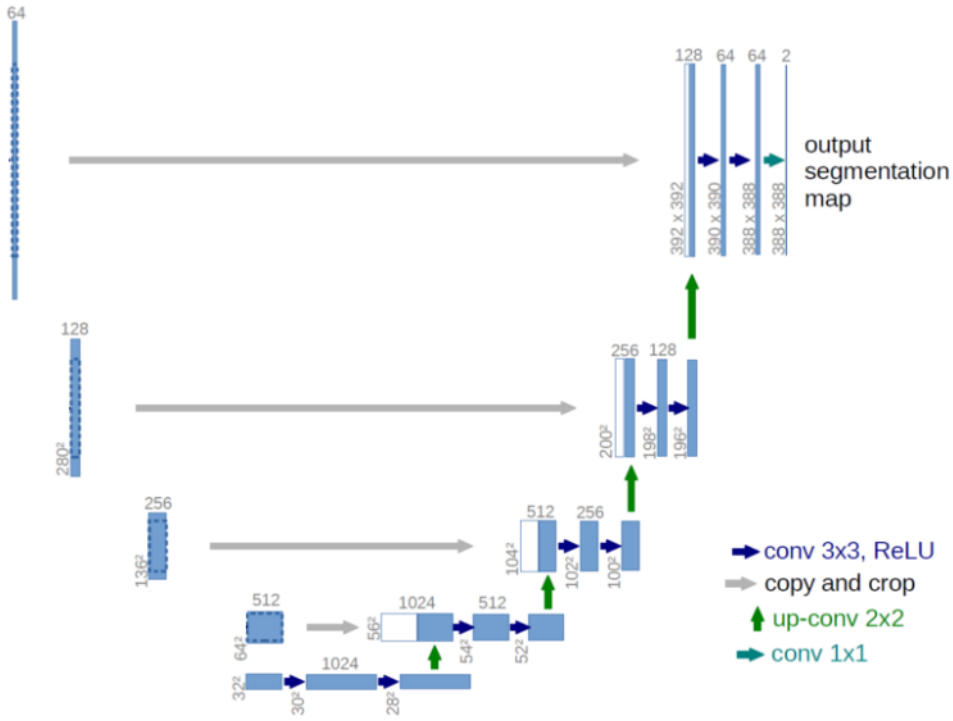
3.2 Cấu trúc mô hình UNet

3.2.1 *Bộ phiên mã*



Bộ phiên mã trong mô hình UNet chứa các lớp cơ bản của một mô hình CNN bình thường, nghĩa là phiên mã cũng sẽ có các lớp tích chập và lớp pooling xen kẽ nhau. Qua mỗi một lớp thì ta sẽ thu được nhiều ma trận đặc trưng có chiều dài và rộng nhỏ hơn chồng lên nhau (các lớp này còn có thể gọi là các channel). Còn sau mỗi lớp pooling thì ma trận đặc trưng sẽ nhỏ lại và chỉ còn chứa những ô có giá trị lớn nhất thể hiện rõ nhất các đặc trưng riêng của ảnh. Sau khi ảnh đã được phần phiên mã xử lý xong thì ta sẽ thu được rất nhiều các lớp ma trận đặc trưng chứa các đặc điểm riêng của ảnh làm cho việc phân loại các pixel tiện lợi và nhanh hơn. Hiểu một cách đơn giản, phiên mã khởi tạo và chắt lọc các đặc trưng của hình ảnh ban đầu. Các đặc trưng này sẽ là cơ sở để ta phân lớp hình ảnh về sau.

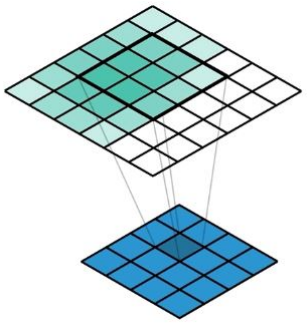
3.2.2 *Bộ dịch mã*



Bộ dịch mã của mô hình Unet

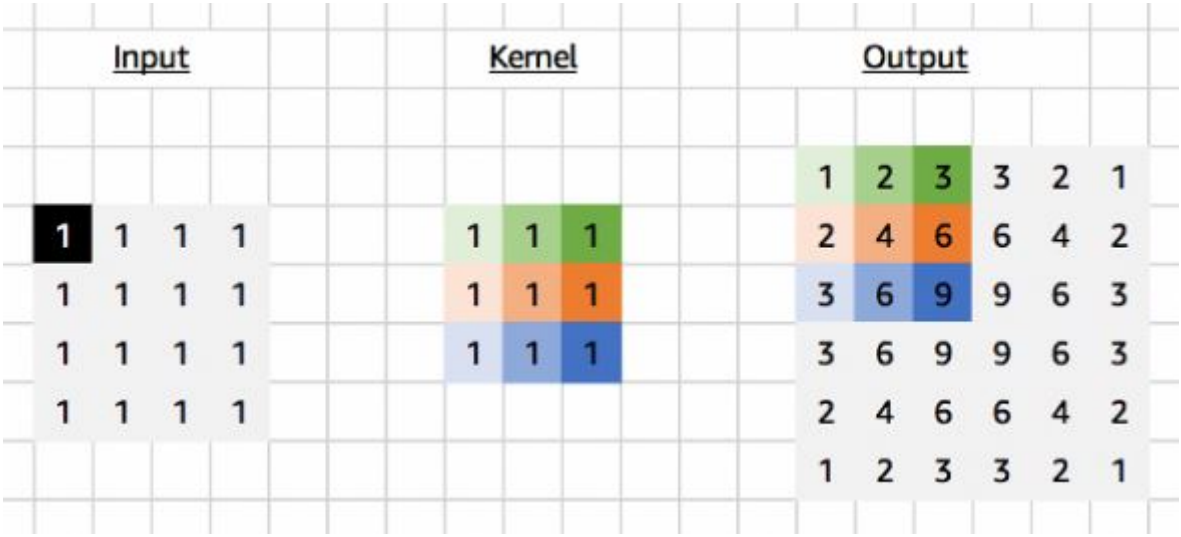
Bộ dịch mã trong mô hình UNet là bộ phận thực hiện quá trình phục hồi lại kích thước của ảnh đã qua bộ phiên mã. Quá trình dịch mã gồm ba giai đoạn cơ bản:

1. Thực hiện phép tích chập chuyển vị (Transposed Convolution) trên ma trận đặc trưng để được một ma trận đặc trưng mới có kích thước lớn hơn.
   Ta có thể nhớ được rằng ở lớp tích chập, ta đã thực hiện việc nhân tích chập giữa ma trận con của ảnh và ma trận của tham số, thì ở quá trình này, việc ngược lại sẽ được thực hiện bằng cách lấy mỗi một ô trên ma trận đặc trưng nhân lần lượt với các ô của một ma trận tham số khác . Kết quả thu được là các ma trận đặc trưng ở kích thước lớn hơn.



Mô tả phép tích chập chuyển vị

   Một ví dụ cụ thể như ma trận đặc trưng có kích thước là 4×4 và ma trận tham số 3×3 thì kết quả sẽ là một ma trận chứa đặc trưng mới kích thước 6×6.



ví dụ cụ thể cho phép tích chập chuyển vị

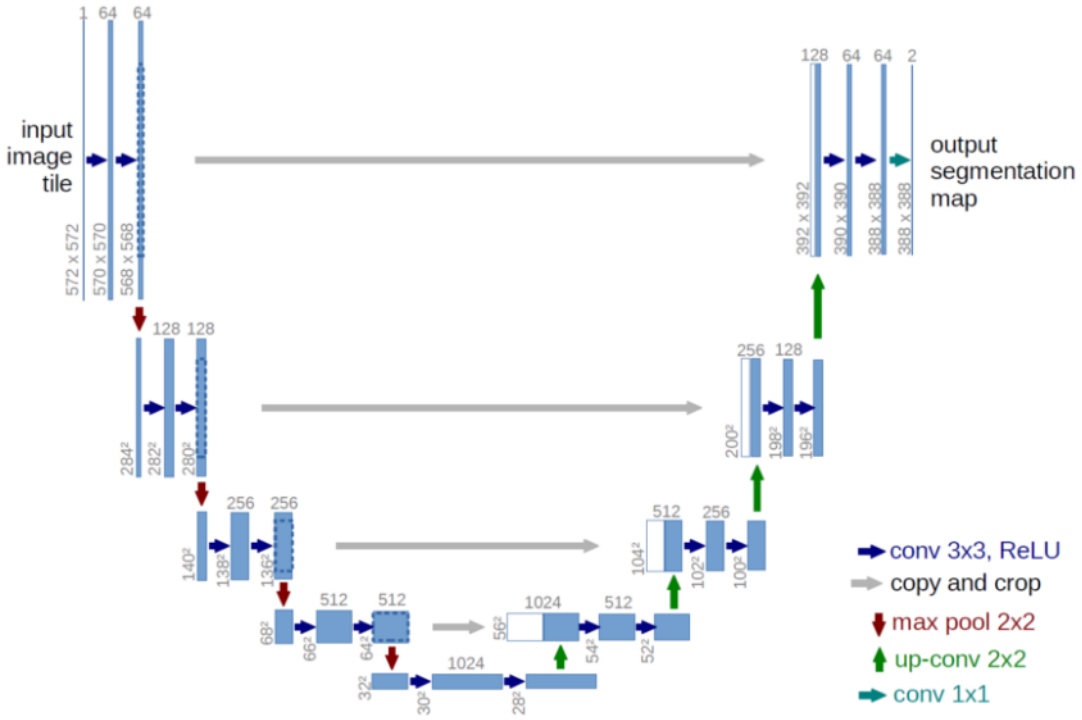
Ở ma trận output thì ô $(0,0)$ sẽ là 1×1, ô $(0,1)$ sẽ là $1 \times 1 + 1 \times 1$, tương tự như vậy với các ô còn lại.

2. Thực hiện bước skip connection.
   Bước này được thực hiện bằng cách ghép ma trận vừa nhận được sau bước 1 với ma trận cùng "cấp"với nó ở bộ phiên mã.

3. Nhân tích chập ma trận đặc trưng nhận được sau bước 2.

bộ lọc ở bước nhân tích chập 1 và 3 sẽ được máy học sao cho phù hợp nhất trong quá trình rèn luyện.
Ta có thể hiểu một cách đơn giản rằng ở bước 1, ta đang phóng lớn ma trận đặc trưng mà ta đã lọc được để được một ma trận có kích thước gần với kích thước ảnh output hơn. Bên cạnh đó, trong quá trình phiên mã, một lượng lớn thông tin tổng quát và vị trí tương đối của hình ảnh đã bị mất đi do các phép tích chập, vì thế ta thực hiện bước 2 để lấy lại các thông tin đã bị mất đi này của hình ảnh. Cuối cùng ở bước 3, ta cố gắng tìm các tích chập phù hợp nhất để kết hợp dữ liệu hai ma trận ở bước 2, sao cho ma trận mới vừa nắm được thông tin về đặc trưng ảnh, vừa khởi tạo lại được vị trí của đặc trưng đó trên ma trận có kích thước gần hơn với dữ liệu ra mong muốn.

Lặp đi lặp lại ba bước này để dần phóng lớn kích cỡ cũng như giảm số lượng của các ma trận đặc trưng, ta thu được các lớp đặc trưng vừa nắm được các chi tiết của hình ảnh, vừa lưu giữ được vị trí thực của các chi tiết ấy trên ảnh. Ở mô hình UNet cụ thể trên, lớp dữ liệu cuối của chúng ta gồm hai ma trận đặc trưng, mỗi ma trận có kích thước $388 \times 388 \times 1$ và tương ứng với hai đặc trưng: "thuộc đối tượng"và "không thuộc đối tượng".



Mô hình UNet đầy đủ

Tới đây, ta dùng hàm soft-max để xác định xác suất mỗi pixel thuộc đối tượng là bao nhiêu. Áp dụng thuật toán lan truyền ngược dựa trên hàm mất mát pixel-wise entropy, ta tìm được mô hình CNN có khả năng phân mảng hình ảnh mà ta mong muốn.

4 THỰC NGHIỆM

4.1 Lập trình mô hình UNet

4.1.1 *Khối tích chập*

```
def conv_block(inp,filters,kernel=(3,3)):

    db = Conv2D(filters, kernel, padding='same')(inp)
    db = BatchNormalization()(db)
    db = Activation('relu')(db)
    db = Conv2D(filters, kernel, padding='same')(inp)
    db = Activation('relu')(db)
    db = BatchNormalization()(db)

    return db
```

Như đã đề cập phía trên, mô hình UNet gồm nhiều cấp, mỗi cấp gồm nhiều lớp. Hình ảnh phía trên biểu diễn một cấp thuộc mô hình UNet.
Dựa vào các dòng code phía trên, ta thấy rằng mô hình của chúng ta có hai lớp tích chập với hàm kích hoạt ReLU.

4.1.2 *Quá trình phiên mã*

Theo như sơ đồ mô hình UNet đã được đề cập phía trên, quá trình "đi xuống"trong mô hình được biểu diễn bằng code như sau:

```
inp=Input(shape=input_shape)

dn0a = conv_block(inp,16,(3,3))
dn0a_pool = MaxPooling2D((2, 2), strides=(2, 2))(dn0a)

dn0 = conv_block(dn0a_pool,32,(3,3))
dn0_pool = MaxPooling2D((2, 2), strides=(2, 2))(dn0)

dn1 = conv_block(dn0_pool,64,(3,3))
dn1_pool = MaxPooling2D((2, 2), strides=(2, 2))(dn1)

dn2 = conv_block(dn1_pool,128,(3,3))
dn2_pool = MaxPooling2D((2, 2), strides=(2, 2))(dn2)

dn3 = conv_block(dn2_pool,256,(3,3))
dn3_pool = MaxPooling2D((2, 2), strides=(2, 2))(dn3)

dn4 = conv_block(dn3_pool,512,(3,3))
dn4_pool = MaxPooling2D((2, 2), strides=(2, 2))(dn4)

cn =conv_block(dn4_pool,1024,(3,3))
```

Quá trình phiên mã

Theo trên, mỗi cấp sẽ gồm một khối tích chập có bộ lọc kích thước 3x3 và một lớp pooling kích thước 2x2.

4.1.3 *Quá trình dịch mã*

Quá trình dịch mã, như phía trên đã trình bày rõ ràng, là quá trình giúp mô hình tái thiết lập lại một ma trận pixel có kích thước tương đương với hình ảnh đưa vào ban đầu.

```
up4 = UpSampling2D((2, 2))(cn)
up4 = concatenate([dn4, up4], axis=3)
up4 = conv_block(up4,512,(3,3))

up3 = UpSampling2D((2, 2))(up4)
up3 = concatenate([dn3, up3], axis=3)
up3 = conv_block(up3,256,(3,3))

up2 = UpSampling2D((2, 2))(up3)
up2 = concatenate([dn2, up2], axis=3)
up2 = conv_block(up2,128,(3,3))

up1 = UpSampling2D((2, 2))(up2)
up1 = concatenate([dn1, up1], axis=3)
up1 = conv_block(up1,64,(3,3))

up0 = UpSampling2D((2, 2))(up1)
up0 = concatenate([dn0, up0], axis=3)
up0 = conv_block(up0,32,(3,3))

up0a = UpSampling2D((2, 2))(up0)
up0a = concatenate([dn0a, up0a], axis=3)
up0a = conv_block(up0a,16,(3,3))

up0a = Conv2D(16, (3, 3), padding='same')(up0a)
up0a = BatchNormalization()(up0a)
up0a = Activation('relu')(up0a)

output = Conv2D(num_class, (1, 1), activation='sigmoid')(up0a)
```

Quá trình dịch mã

Ta dễ dàng thấy rằng, quá trình dịch mã dài hơn và phức tạp hơn quá trình phiên mã. Đó chính là bởi mô hình lúc này bao gồm quá trình skip connection để tái sử dụng hình ảnh tổng quan đã mất qua các lớp tích chập và pooling.

4.2 Kết quả mô hình

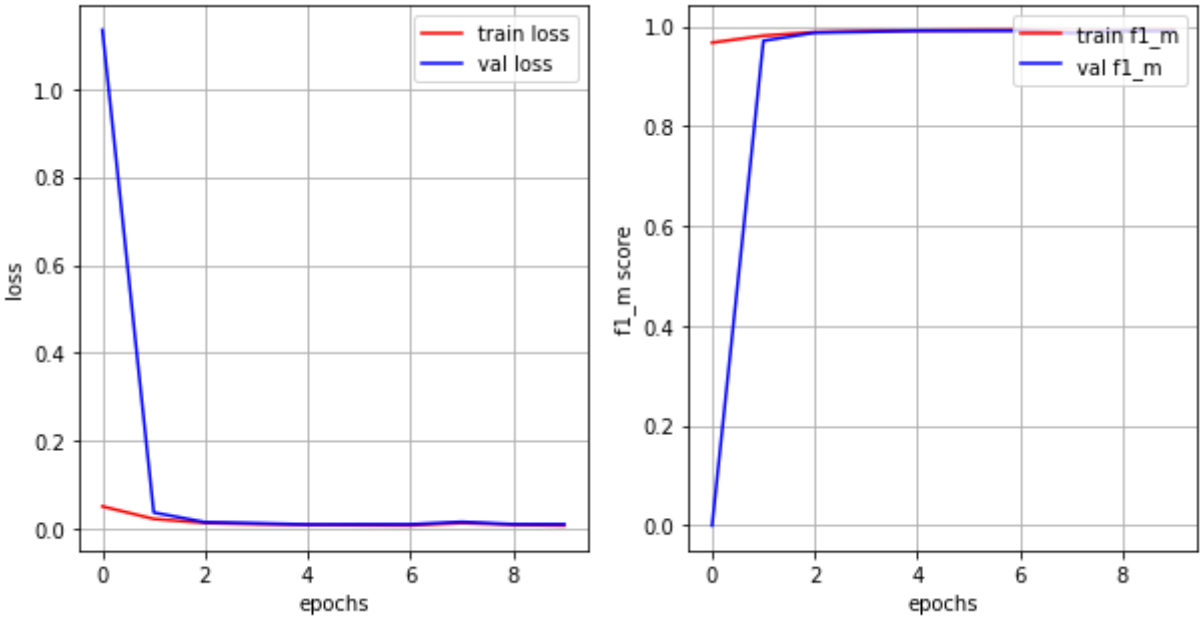
4.2.1 *Số lượng tập dữ liệu huấn luyện và tập đánh giá*

Trong tập dùng để huấn luyện của Kaggle cho ta tổng cộng **5088 bộ input-output**. Để mô hình không bị quá chủ quan, nên đưa vào bộ dữ liệu đánh giá (validation data) để đánh giá các bước đi của mô hình. Chính vì vậy, ta có thể chia bộ dữ liệu làm 2 phần (tập dữ liệu train nhiều hơn tập dữ liệu đánh giá) như sau:

```
Train set: 4284
Validation set: 804
```

4.2.2 *Đồ thị biểu diễn hàm mất mát và độ chính xác theo thời gian*

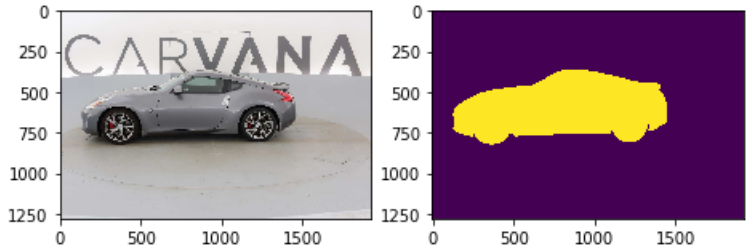
Ta có thể dùng nhiều loại hàm đánh giá cho mô hình này nhưng trong dự án này, chúng tôi sử dụng hàm mất mát (loss function) và thêm một hàm tính toán điểm số cho mô hình là f1 score.



Đồ thị của loss function và f1 score

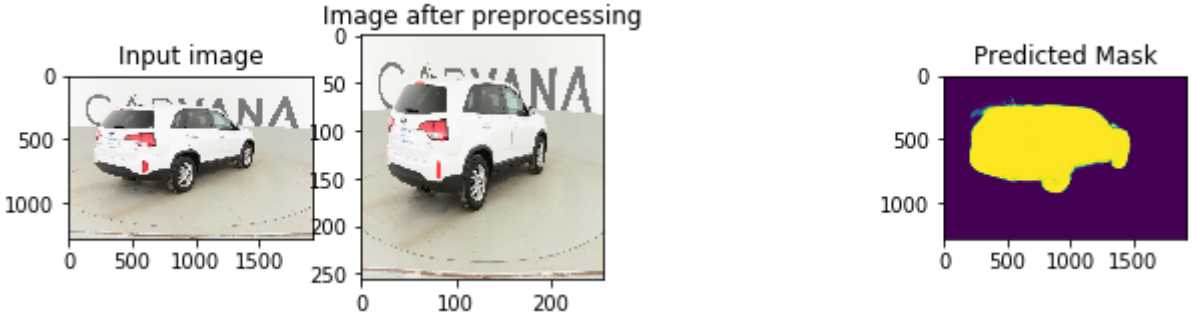
4.2.3 *Đánh giá độ chính xác của mô hình*

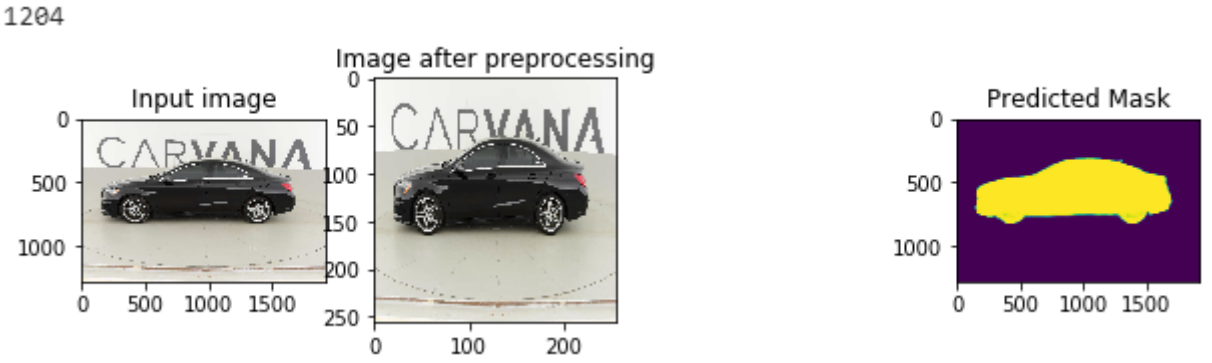
Đây là kết quả mong muốn của một mô hình Image Segmentation



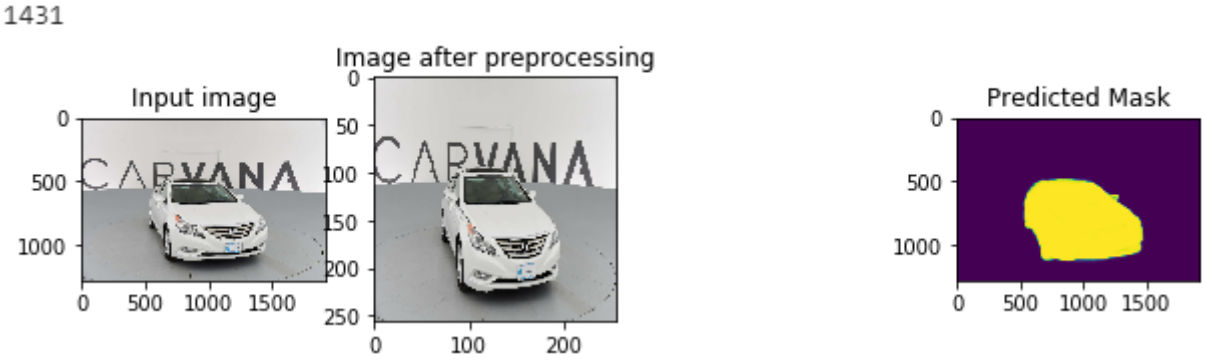
Expected result

Để so sánh, chúng tôi xin phép được trình bày 3 dự đoán mà mô hình của chúng tôi đưa ra. Những hình ảnh máy đọc vào được cam đoan không nằm trong tập huấn luyện.





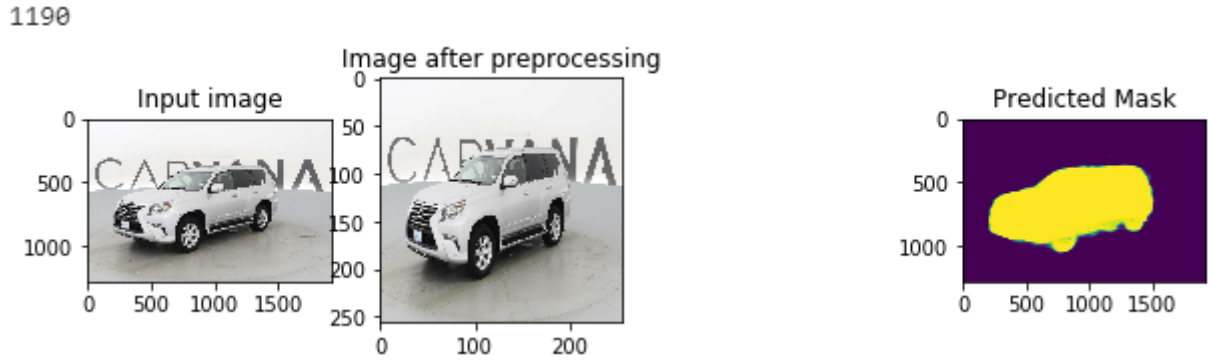
Hình ảnh thứ 1204 trong tập kiểm tra



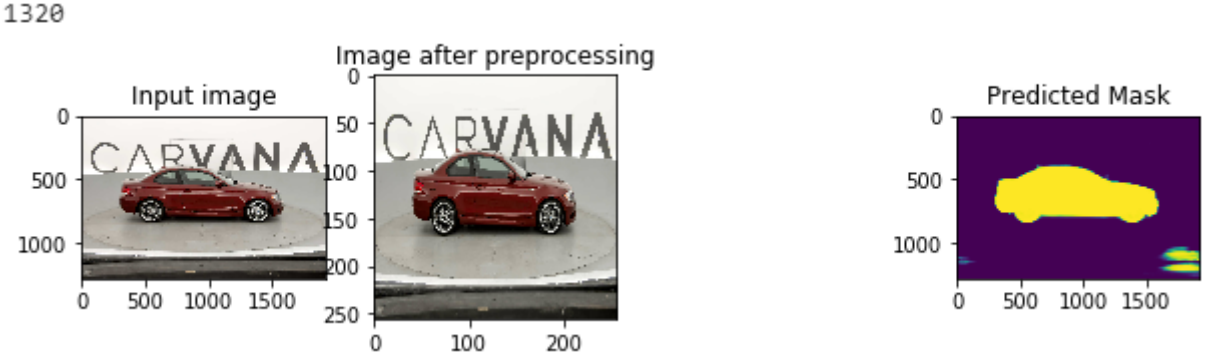
Hình ảnh thứ 1431 trong tập kiểm tra

4.2.4 *Điểm hạn chế của mô hình*

Do có độ chính xác tương đối, đôi khi dự đoán của máy sẽ không được chính xác như những hình ảnh khác. Chúng tôi sẽ trình bày một số ví dụ.

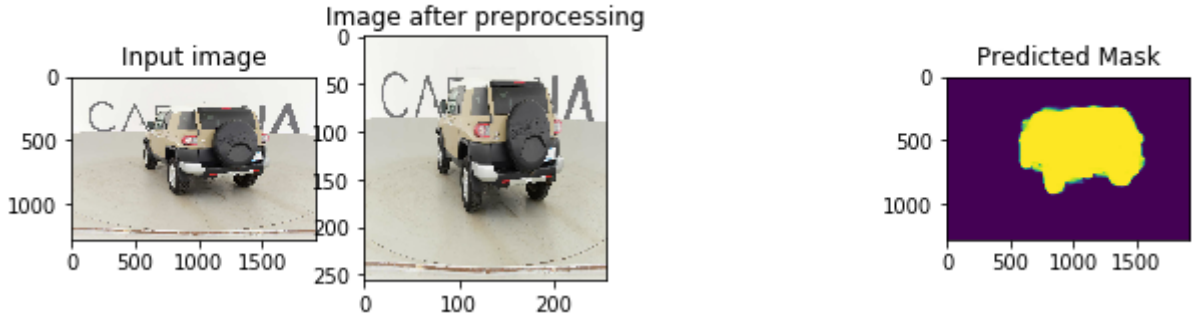


Đường viền của chiếc xe vẫn còn chưa chính xác



Mô hình bị nhầm lẫn nền và xe

1348



Hình ảnh của chiếc xe không còn rõ ràng

TÀI LIỆU

[1] Cs231n: Convolutional neural networks for visual recognition. http://cs231n.github.io/convolutional-networks/.

[2] Otavio Good. Convolutional neural network visualization. https://www.youtube.com/watch?v=f0t-OCG79-U. Accessed on Youtube.

[3] Jeremy Jordan. An overview of semantic image segmentation. https://www.jeremyjordan.me/semantic-segmentation/, May 2018.

[4] Ujjwal Karn. An intuitive explanation of convolutional neural networks. https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/, Aug 2016.

[5] Harshall Lamba. Understanding semantic segmentation with unet. https://towardsdatascience.com/understanding-semantic-segmentation-with-unet-6be4f42d4b47, Feb 2019.

[6] Thom Lane. Transposed convolutions explained with... ms excel! https://medium.com/apache-mxnet/transposed-convolutions-explained-with-ms-excel-52d13030c7e8, Nov 2018.

[7] Mayank Mishra. Convolutional neural networks, explained. https://www.datascience.com/blog/convolutional-neural-network, Mar 2019.

[8] Augustus Odena, Vincent Dumoulin, and Chris Olah. Deconvolution and checkerboard artifacts. https://distill.pub/2016/deconv-checkerboard/, Oct 2016.

[9] Heet Sankesara. Unet. https://towardsdatascience.com/u-net-b229b32b4a71, Jan 2019.

[10] Mathew Stewart. Simple introduction to convolutional neural networks. https://towardsdatascience.com/simple-introduction-to-convolutional-neural-networks-cdf8d3077bac, Feb 2019.

[11] SuperDataScience Team. Convolutional neural networks (cnn): Softmax cross-entropy. https://www.superdatascience.com/blogs/convolutional-neural-networks-cnn-softmax-crossentropy, Aug 2018.

[12] Song Yuheng and Yan Hao. Image segmentation algorithms overview. *CoRR*, abs/1707.02051, 2017.