

# NÂNG CAO ĐỘ PHÂN GIẢI HÌNH ẢNH BẰNG CÁC MÔ HÌNH HỌC SÂU

Võ Thành Hoàng Sơn, Trần Nguyễn Quỳnh Trâm

Khoa Công nghệ Thông tin, Trường Đại học Ngoại ngữ - Tin học Thành phố Hồ Chí Minh  
Tác giả liên hệ: Email: Tác giả liên hệ: Email: 19dh110660@st.huflit.edu.vn | Điện thoại: 038.405.3455

## Tóm tắt

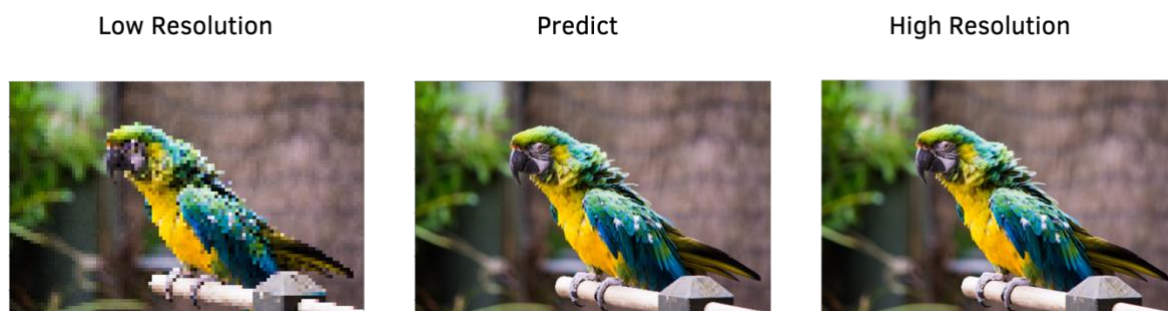
Như chúng ta đã biết hiện nay các thiết bị hiển thị đã đạt đến những mức độ phân giải rất cao 2K, 4K ... thậm chí 16K. Tuy nhiên những thông tin chúng ta muốn hiển thị phần lớn không thể sử dụng được (vì được chụp bằng công nghệ cũ, độ phân giải thấp). Vì thế bài toán Single Image Super-Resolution (SISR) đã được tạo ra khắc phục những hạn chế trên. SISR giúp nâng cao độ phân giải của hình ảnh từ một hình ảnh có độ phân giải thấp (LR) đến cao (HR). Bài toán này được ứng dụng rất nhiều trong thực tế và đặc biệt tại những lĩnh vực như: An ninh, Y tế, Thiên văn ... Trong bài báo này tôi sẽ trình bày về những khó khăn của các phương pháp truyền thống và các phương pháp học sâu đã được sử dụng đối với bài toán SISR, tiến hành thực nghiệm một số mô hình như SRCNN, EDSR cũng như thay thế lớp Upsampling với Bicubic và Pixel Shuffle, so sánh độ tối ưu của chúng trên 3 tập dữ liệu Set5, Urban100 và Div2k và phát triển một ứng dụng thử nghiệm.

**Từ khoá:** Super-Resolution; Single Image Super-Resolution; Enhance image

## 1. GIỚI THIỆU

Những tấm ảnh là nơi lưu trữ nhiều kỷ niệm trong cuộc sống của bạn, với sự phát triển nhanh chóng của khoa học kỹ thuật, bạn có thể tạo ra những tấm ảnh lung linh bất cứ khi nào và bất cứ ở đâu chỉ bằng một chiếc điện thoại cầm tay. Tuy nhiên, trong thực tế, một bức ảnh có thể đẹp và rõ trên điện thoại nhưng khi nó được tải và mở trên máy tính thì bị mờ hay gọi là bể hình. Nguyên nhân chính làm tấm ảnh bị mờ là vì bạn có một tấm ảnh có độ phân giải thấp ít điểm ảnh (pixel) trên một màn hình có kích thước độ phân giải quá lớn. Để tạo ra một hình ảnh có độ phân giải cao đồng nghĩa với việc yêu cầu nhiều điểm ảnh (pixel) hơn để thể hiện các chi tiết một cách sắc nét hơn nhưng nó sẽ đòi hỏi máy tính có cấu hình cao cũng như ổ cứng lớn để xử lý và lưu trữ, điều này gây ra tốn kém chi phí. Trong thực tế, một số lĩnh vực luôn luôn yêu cầu hình ảnh có độ phân giải cao như hình ảnh y tế (MRI, CT ...), hình ảnh trong thiên văn học [1]. Vì thế, làm sao giảm dung lượng của hình ảnh nhưng vẫn làm cho ảnh rõ và đẹp đã và đang là thử thách lớn cho các nhà khoa học trên thế giới.

Để tạo ra hình ảnh độ phân giải cao, yêu cầu số lượng pixel lớn vì thế một cách đơn giản để giải quyết bài toán Single Image Super Resolution là tăng số lượng pixel sau đó tô màu những pixel mới dựa trên những pixel ban đầu. Và trên thực tế những phương pháp ban đầu đã được ra đời như vậy có thể kể đến như Nearest-neighbor interpolation, Bilinear interpolation và Bicubic interpolation. Tuy nhiên hình ảnh được tạo ra từ những phương pháp nội suy kể trên cho ra kết quả bị răng cưa (pixelated) hoặc là bị mờ.



**Hình 1.** Giới thiệu về bài toán SISR

Để tô màu vào những pixel mới chúng ta cần một phương pháp có chiến lược hơn thông qua việc quan sát và thu thập thông tin của các đặc điểm chung của một kiểu hình ảnh riêng biệt. Và đó chính xác những gì một mô hình học sâu (Deep learning) thực hiện. Với sự phát triển mạnh mẽ của Deep Learning và đặc biệt là mô hình GAN [2] (Generative adversarial networks) hiện nay, bài toán SISR đã đạt những kết quả rất tiềm năng và có thể ứng dụng vào một số trường hợp thực tế. Tuy nhiên, để tái tạo hình ảnh một cách hoàn hảo nhất có lẽ vẫn còn một chặng đường dài vì kết quả hiện nay vẫn khó khăn trong việc tái tạo khuôn mặt người cũng như hiệu suất còn phụ thuộc nhiều vào các yếu tố như độ sáng, màu sắc ...

Trong bài báo này tôi sẽ trình bày về các phương pháp dùng để giải quyết bài toán SISR và sử dụng một số thuật toán tiêu biểu của các phương pháp học sâu chúng ta sẽ tìm hiểu như: SRCNN [3], EDSR [4]. Tiếp theo tôi sẽ tiến hành thực nghiệm trên tập dữ liệu DIV2K. Sau đó đánh giá và so sánh trên các tập dữ liệu DIV2K(Validation) [5], Set5 [6], Urban100[7]. Cuối cùng tôi sẽ thử nghiệm thay đổi hàm loss và thay lớp Upsampling (Thuật ngữ chỉ quá trình tăng số lượng điểm ảnh lên so với hình ảnh ban đầu ) vào nhằm kiểm tra chúng có lại kết quả vượt trội hơn so với ban đầu hay không.

Cấu trúc của những phần tiếp theo của bài báo sẽ bao gồm: Phần 2 công trình liên quan, ở phần này tôi sẽ trình bày về những phương pháp cổ điển cũng như các hướng tiếp cận của những mô hình học sâu cho bài toán này. Phần 3 tôi sẽ đề xuất phương pháp sử dụng trong bài toán này. Sau đó tôi sẽ tiến hành thực nghiệm để nhận xét kết quả của phương pháp đề xuất và phát triển một ứng dụng thử nghiệm. Cuối cùng là kết luận và đánh giá tổng quan về bài báo này.

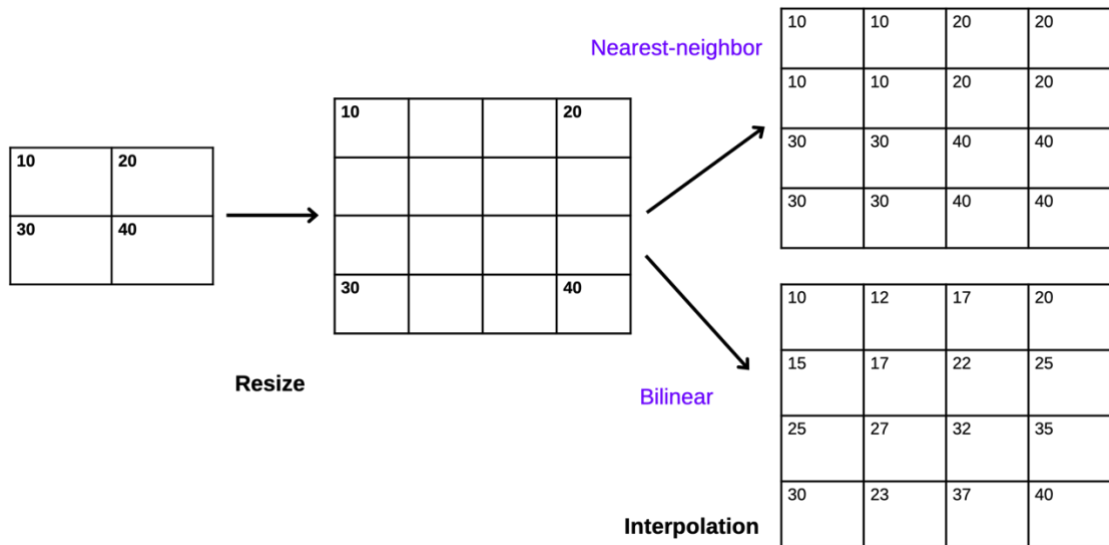
## 2. CÔNG TRÌNH LIÊN QUAN

### 2.1. Super Resolution dựa trên nội suy

Như đã đề cập ở bên trên những phương pháp sơ khởi được biết đến là nội suy (interpolation) Những phương pháp này sử dụng giá trị màu của các pixel lân cận cũng như mối liên hệ giữa các pixel để suy ra giá trị cho điểm cần tìm. Có rất nhiều phương pháp nội suy phổ biến như nội suy lân cận, nội suy song tuyến, nội suy khối như trong Hình 2.

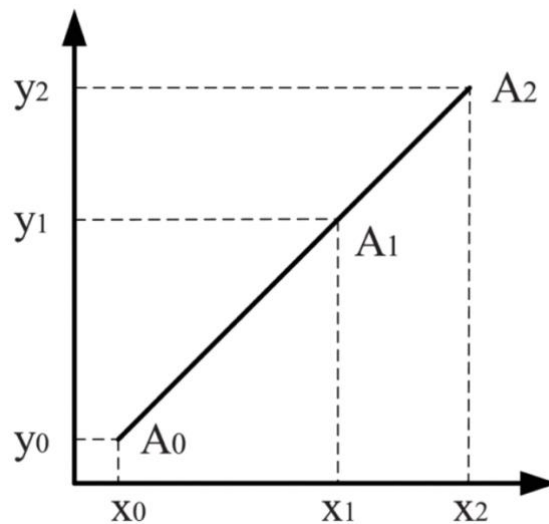
**Nội suy lân cận** hay còn được gọi là nội suy gần (**proximal interpolation**) là phương pháp đơn giản nhất trong các phương pháp nội suy. Sau khi được resize hình ảnh mới sẽ chứa những pixel trống xung quanh các pixel ban đầu tùy thuộc vào scale khi Upsampling . Phương pháp này sẽ tô vào những pixel trống giá trị bằng với giá trị gần nhất của nó. Vì thế

hình ảnh sau khi sử dụng phương pháp trên thường xảy ra hiện tượng pixelated (răng cưa) và trông hình ảnh không cải thiện quá nhiều so với ảnh ban đầu.



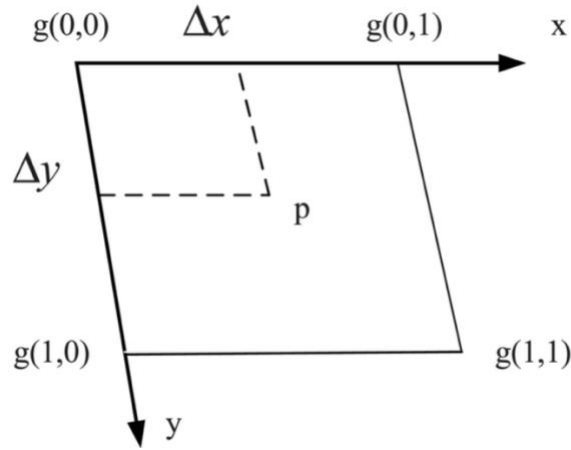
**Hình 2.** Mô tả cách thực hiện của phương pháp nội suy lân cận và nội suy song tuyến

Ví dụ, Theo Hình 3 ta gọi điểm  $A_0$  là điểm cần tìm,  $A_0$  sẽ ánh xạ đến  $A_1$  tuy nhiên nếu điểm  $A_1$  không chứa giá trị điểm ảnh thì điểm  $A_0$  sẽ tiếp tục được ánh xạ đến điểm  $A_2$  khi đó ta có giá trị  $A_0 = A_2$ .



**Hình 3.** Cách thực hiện của phương pháp nội suy lân cận dưới dạng vector

**Phương pháp nội suy song tuyến** là phương pháp nội suy sử dụng giá trị của 4 điểm ảnh liên kề  $g(0,0), g(0,1), g(1,0), g(1,1)$  xung quanh điểm  $p$  để tiến hành tính toán giá trị cho  $p$  theo 2 trục  $x, y$ . Được biểu diễn tại Hình 4.



**Hình 4.** Cách thực hiện của phương pháp nội suy song tuyến dưới dạng vector

Giả sử khoảng cách giữa các điểm ảnh như trong Hình 4. Khi đó ta có công thức tính giá trị màu của điểm p như sau (1):

$$D_p = [\omega(\Delta x)\omega(1 - \Delta x)] \begin{bmatrix} D_{00} & D_{01} \\ D_{10} & D_{11} \end{bmatrix} \begin{bmatrix} \omega(\Delta y) \\ \omega(1 - \Delta y) \end{bmatrix} \quad (1)$$

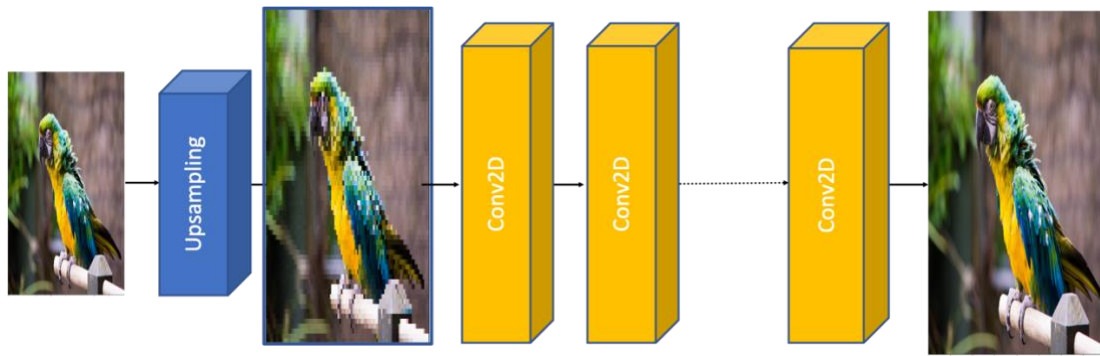
Phương pháp nội suy khối có công thức tính khá tương đồng so với thuật toán nội suy song tuyến tuy nhiên ở nội suy khối chúng ta sử dụng 16 điểm liên kề với điểm p thay vì là 4.

Cả hai thuật toán nội suy song tuyến và nội suy khối đều cho ra kết quả là hình ảnh bằng cách lấy mẫu lại theo yêu cầu tỷ lệ, nhưng thuật toán nội suy khối có độ phức tạp cao hơn thuật toán nội suy song tuyến.

## 2.2. Mô hình học sâu cho bài toán Super-Resolution

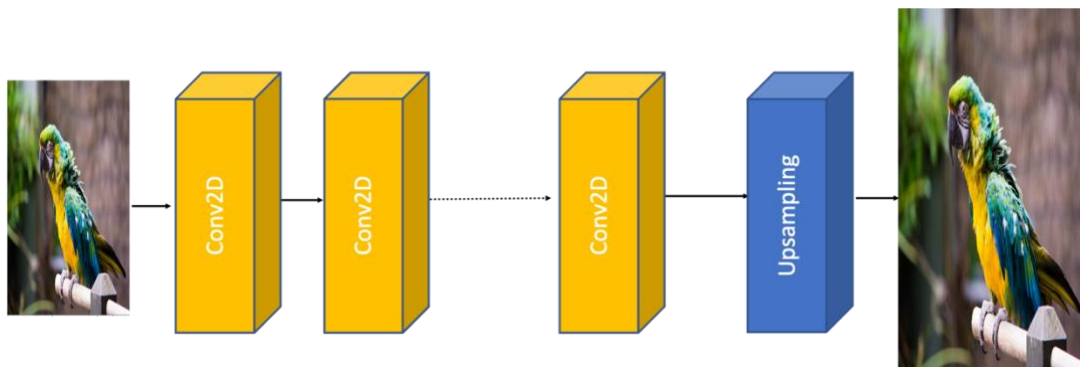
Mạng nơ-ron tích chập (CNN) là một mô hình mạng nơ-ron học sâu được sử dụng trong các bài toán sử dụng đầu vào là hình ảnh. Mạng nơ-ron tích chập thường được sử dụng rộng rãi trong việc nhận dạng (detection), phân vùng (segmentation) và phân loại (classification). Ngoài ra đối CNN đã khá thành công trong việc giải quyết vấn đề của bài toán SISR. Tiếp theo ở dưới đây tôi sẽ nói về những phương pháp xử lý SISR bằng các mô hình học sâu.

**Pre-Usampling Super-Resolution:** Tương tự như những phương pháp nội suy truyền thống, bước đầu tiên phương pháp này cũng Upsampling hình ảnh lên. Sau đó thay vì điền trị vào pixel bằng công thức nội suy thì phương pháp này cho qua một mạng CNN để trích xuất đặc trưng và huấn luyện. Giá trị tại điểm ảnh khi điền vào đã được học qua nhiều hình ảnh khác. Những mô hình sử dụng phương pháp này có thể kể đến như: SRCNN (sẽ được đề cập ở bên dưới), VDSR [8].



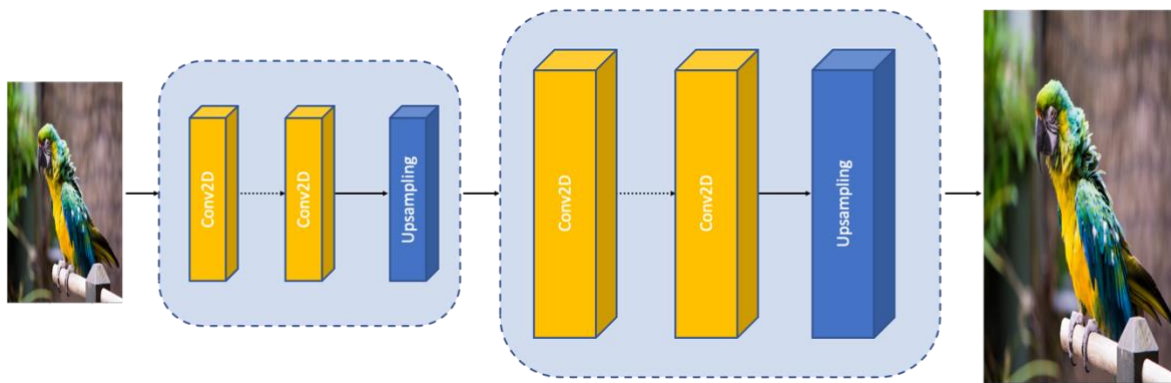
**Hình 5.** Mô hình học sâu sử dụng phương pháp Pre-Upsampling

**Post-Upsampling Super-Resolution:** Ở phương pháp Upsampling tiền xử lý quá trình Upsampling được thực hiện trước quá trình huấn luyện vì thế trong quá trình huấn luyện số lượng tính toán cũng tăng lên rất nhiều. Phương pháp Upsampling hậu xử lý cố gắng giải quyết điều đó bằng việc trích xuất đặc trưng cũng như huấn luyện sau đó mới thực hiện Upsampling qua đó giảm thiểu được đáng kể số lần tính toán. Nhờ đó những mô hình sử dụng phương pháp này đã trở nên rất phổ biến. Ngoài ra trong quá trình Upsampling thay vì sử dụng phương pháp nội suy mỗi lớp Convolution sẽ được thay vào giúp cho mạng này được huấn luyện từ đầu đến cuối. Một vài mô hình sử dụng phương pháp này như: FSRCNN [9], ESPCN [10], EDSR [4] (sẽ được đề cập thêm).



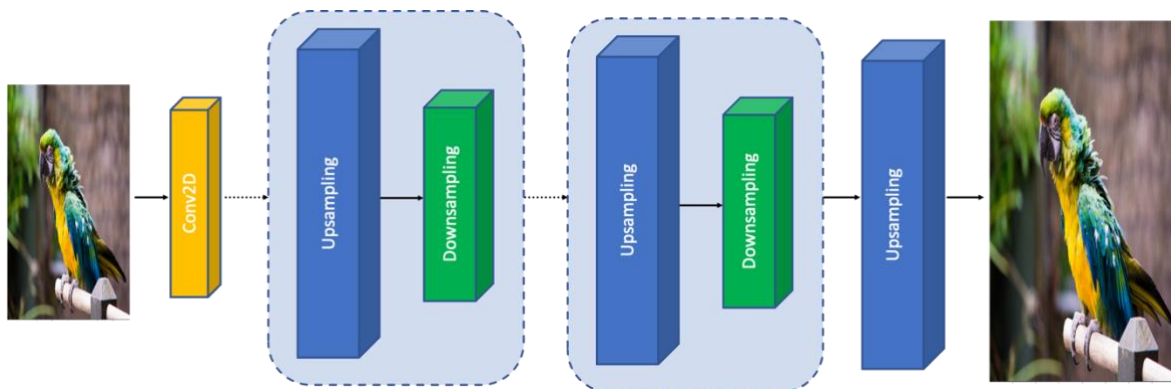
**Hình 6.** Mô hình học sâu sử dụng phương pháp Post-Upsampling

**Progressive Upsampling Super-Resolution:** Trong những tình huống yêu cầu độ lớn của kết quả cao hơn nhiều so với hình ảnh ban đầu x8, x16 ... Ta có thể giải quyết đơn giản bằng cách thêm nhiều lớp Upsampling tại những điểm khác nhau của mô hình thay vì chỉ một lớp Upsampling thật lớn. ví dụ nếu bạn được yêu cầu xử lý hình ảnh x8, thay vì chỉ thêm vào một lớp Upsampling x8 ở đầu hoặc cuối mô hình thì bạn có thể chèn 3 lớp upsampling vào những vị trí trong mô hình. Tuy nhiên, cách tiếp cận này đòi hỏi các chiến lược đào tạo nâng cao và thiết kế mô hình nhiều giai đoạn phức tạp để đảm bảo sự ổn định đào tạo tổng thể.



**Hình 7.** Mô hình học sâu sử dụng phương pháp Progressive Upsampling

**Iterative up-and-down Sampling Super-Resolution:** Đây là một phương pháp được giới thiệu gần đây đúng như cách gọi của nó. Phương pháp này sử dụng các lớp Upsampling và Downsampling (Thuật ngữ chỉ quá trình ngược lại của Upsampling) tại những phần khác nhau của mô hình mạng. Nhờ vậy chúng ta có thể nắm bắt được sự phụ thuộc lẫn nhau giữa LR và HR một cách chính xác hơn và vì thế kết quả thu được sẽ có chất lượng tốt hơn. Những mô hình như DBPN [11], SRFBN [12], ... đã sử dụng phương pháp này.



**Hình 8.** Mô hình học sâu sử dụng phương pháp Iterative up-and-down Sampling

Như Hình 5, chúng ta có thể thấy lớp Upsampling đứng ở đầu của cấu trúc mô hình mạng sử dụng phương pháp Pre-Upsampling. Trái ngược với đó phương pháp Post-Upsampling lại có lớp Up-Sampling đứng cuối mô hình. Ở mô hình sử dụng phương pháp Progressive Upsampling ta có thể thấy có nhiều hơn một lớp Upsampling và ở các vị trí khác nhau trong mô hình. Cuối cùng là phương pháp Iterative up-and-down Sampling ta thấy có thêm những lớp DownSampling sau mỗi lớp Upsampling (Trừ lớp Upsampling cuối cùng) Hình 8.

### 2.3. Những phương pháp Upsampling

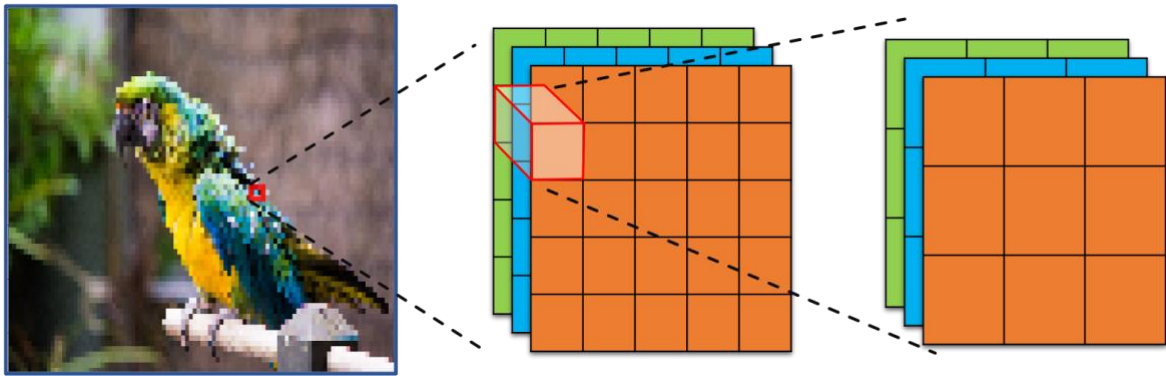
**Upsampling dựa trên nội suy:** Nội suy hình ảnh là những thuật toán ban đầu trong việc xử lý bài toán SISR về bản chất thì những phép toán nội suy cũng sử dụng tăng kích thước hình ảnh trước. Vì thế ta có thể sử dụng phương pháp nội suy như một lớp Upsampling trong mô hình mạng nơ-ron. Tuy nhiên nhược điểm lớn nhất của phương pháp nội suy là chúng chỉ sử dụng giá trị thông tin của chính hình ảnh ban đầu và không có khả năng học tập vì thế khi sử dụng trong thường dễ tạo ra nhiễu cũng như bị mờ. Và vì chúng khá tốn



kém trong việc tính toán nên chúng không thường được sử dụng trong những mô hình gần đây. Thay vào đó người ta sử dụng phương pháp upsampling có thể huấn luyện được.

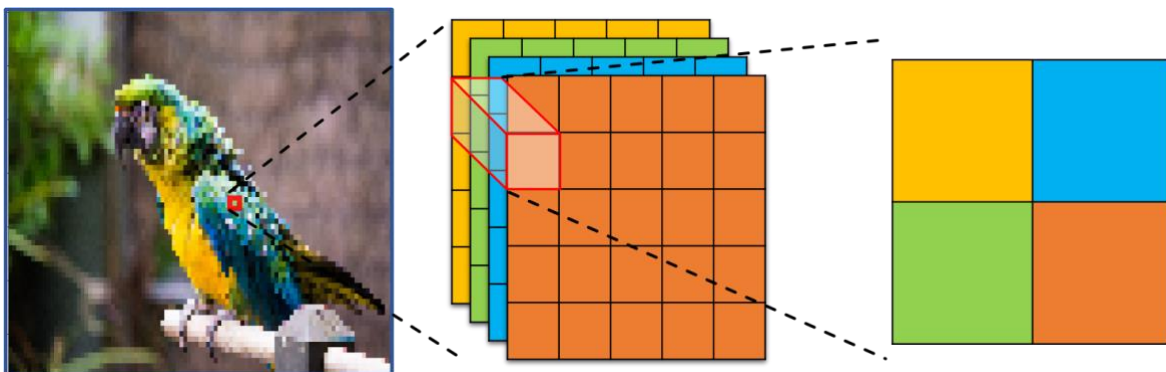
**Upsampling dựa trên huấn luyện:** Người ta sử dụng các lớp Convolution có thể huấn luyện trọng số để làm lớp Upsampling cho mô hình. Có thể kể đến như: Lớp tích chập chuyển vị (Transposed Convolution Layer), lớp điểm ảnh phụ Sub-Pixel Convolution, Meta Upscale Module [13]

**Transposed Convolution Layer:** hay còn được gọi là lớp giải chập. Lớp này thực hiện ngược lại với lớp tích chập nếu đầu vào của lớp tích chập có kích thước là  $w_1, h_1$  và đầu ra là  $w_2, h_2$  thì lớp giải chập sẽ có đầu vào là  $w_2, h_2$  và đầu ra là  $w_1, h_1$ . Lớp giải chập cũng sử dụng một cửa sổ trượt giống lớp tích chập. Bước đầu tiên lớp này sẽ phóng to hình ảnh lên bằng cách thêm các điểm ảnh có giá trị bằng 0 vào, tiếp theo sử dụng cửa sổ trượt qua từng điểm ảnh. Qua đó ta thu được một ma trận có kích thước bằng cửa sổ trượt và có giá trị bằng tích giữa điểm ảnh ban đầu và cửa sổ trượt. Cuối cùng lấy tổng giá các vị trí tương ứng trên các ma trận và thu được hình ảnh đầu ra.



Hình 9. Lớp giải chập

**Sub-pixel Convolution Layer:** Ở lớp này bước đầu (Convolution) người ta tăng số kênh màu của hình ảnh lên  $s^2$  lần. Nếu hình ảnh ban đầu có kích thước  $w, h, c$  thì sau bước đầu kích thước sẽ là  $w, h, s^2c$ . Bước tiếp theo được gọi là bước tái định hình (Reshaping) còn được gọi là shuffle thực hiện, kết quả của bước này là tạo ra hình ảnh lớp kích thước  $sw, sh, c$ . So với lớp tích chập chuyển vị thì lớp này cung cấp nhiều thông tin về ngữ cảnh hơn qua đó giúp quá trình học tập sẽ cho ra kết quả tốt hơn. Tuy nhiên việc dự đoán độc lập từng điểm ảnh trong một vùng sẽ dẫn đến việc cho ra kết quả không mượt mà.



Hình 10. Lớp Sub-pixel tích chập

### 3. PHƯƠNG PHÁP ĐỀ XUẤT:

#### 3.1. Tổng quan phương pháp

Trong bài viết hôm nay tôi sẽ sử dụng hai mô hình học sâu là SRCNN [3] sử dụng phương pháp Pre-Upsampling được đề xuất bởi *Dong et al.* vào năm 2015 và EDSR [4] sử dụng phương pháp Post-Upsampling được đề xuất bởi một nhóm đứng đầu là *Lim et al.* và đã thắng cuộc thi NTIRE2017 [5]. Hai mô hình trên là những mô hình tiêu biểu nhất của bài toán SISR.

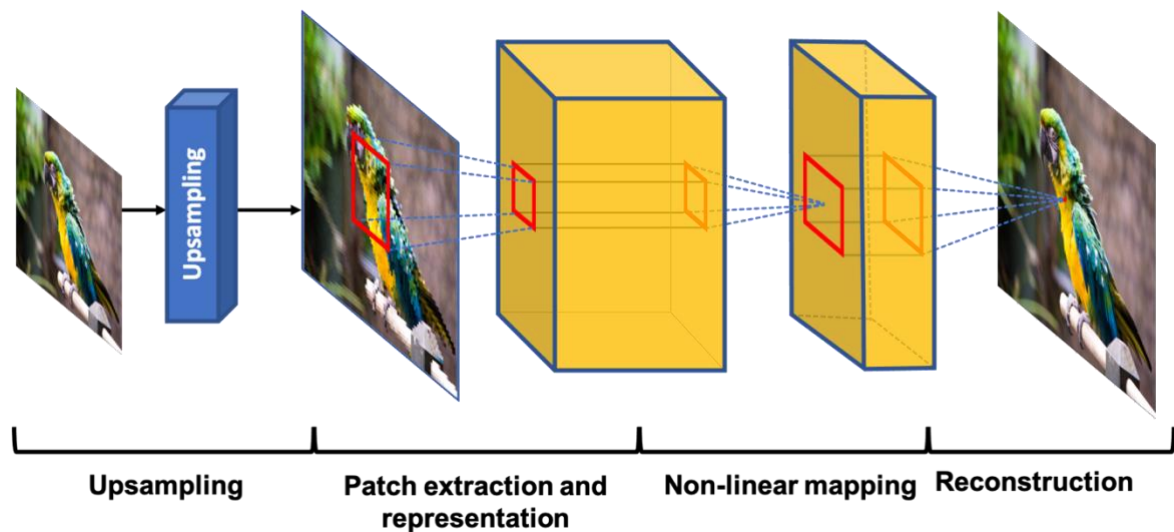
Đầu vào của cả hai mô hình là hình ảnh LR đã được giảm độ phân giải từ hình ảnh HR. Kết quả thu được sau khi xử lý sẽ là hình ảnh SR có kích thước tương ứng với HR và có độ phân giải gần giống HR nhất.

#### 3.2. Cấu trúc mô hình mạng

##### 3.2.1. Convolutional Neural Networks for Super-Resolution (SRCNN)

Mô hình này sử dụng phương pháp Pre-Upsampling vì thế lớp đầu tiên là lớp Upsampling sau đó là những lớp Convolution. Cấu trúc bao gồm 3 bước là: Patch extraction and representation, Non-linear mapping và Recontruction.

Trong đó Patch extraction and representation có nhiệm vụ trích từng khung hình trong hình ảnh đầu vào có hình dáng  $h, w, c$  thành 1 vector  $n_1$  chiều bằng cách cho qua một lớp cửa sổ có kích thước  $f_1 * f_1$ . Kế tiếp dữ liệu sẽ qua các lớp Non-linear mapping có nhiệm vụ ánh xạ từng vector  $n_1$  chiều thành vector  $n_2$  bằng cách lọc qua các bộ lọc.



**Hình 11.** Cấu trúc mô hình mạng SRCNN

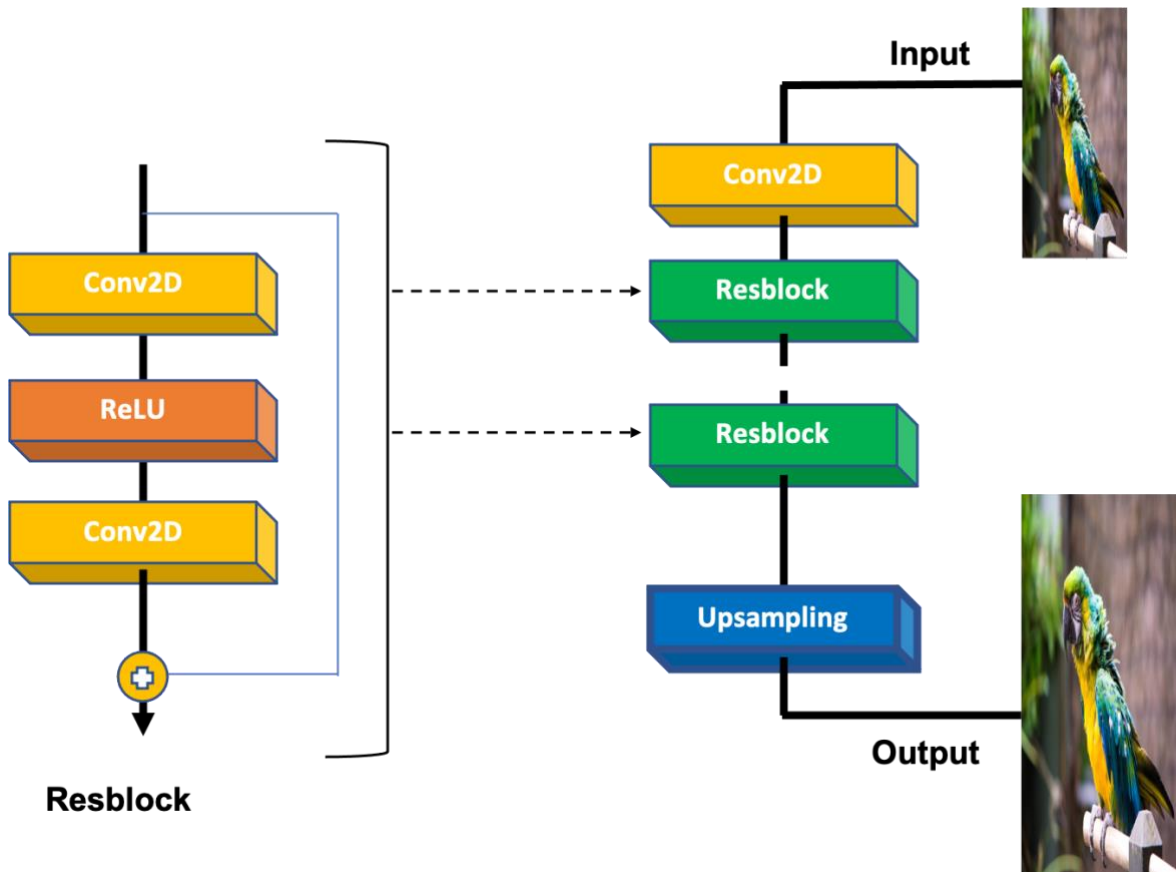
Ở đây tôi tiến hành xây dựng lại cấu trúc của *Dong et al* vì thực nghiệm đã biết rằng nếu tăng thêm số lớp ở bước non-linear mapping sẽ không cho kết quả tốt hơn. Cuối cùng dữ liệu sẽ được truyền qua lớp Recontruction có nhiệm vụ tái tạo lại hình dáng của hình như ban đầu có kích thước  $h, w, c$ .



### 3.2.2. Enhanced Deep Residual Networks (EDSR)

Khác với SRCNN [3] mô hình này sử dụng phương pháp Post-Upsampling nên lớp Upsampling sẽ đứng ở cuối mô hình. EDSR [4] được phát triển dựa trên mạng Residual Network. ResNet là một mạng học sâu sử dụng các kết nối tắt, những kết nối tắt này sẽ bỏ qua một hoặc nhiều lớp trong mạng. Nhờ vậy giúp quá trình hội tụ của mô hình trở nên nhanh chóng hơn. Cách lớp trong mạng bị kết nối tắt bỏ qua sẽ được gọi là Resblock.

Tuy nhiên vì là bài toán Super-Resolution nên trong những Resblock đã được tinh chỉnh so với mô hình gốc. Tại các Resblock các lớp Batch-Norm đã bị loại bỏ vì chúng sẽ khiến hình ảnh mất đi chi tiết điều hoàn toàn trái ngược với chúng ta mong muốn.

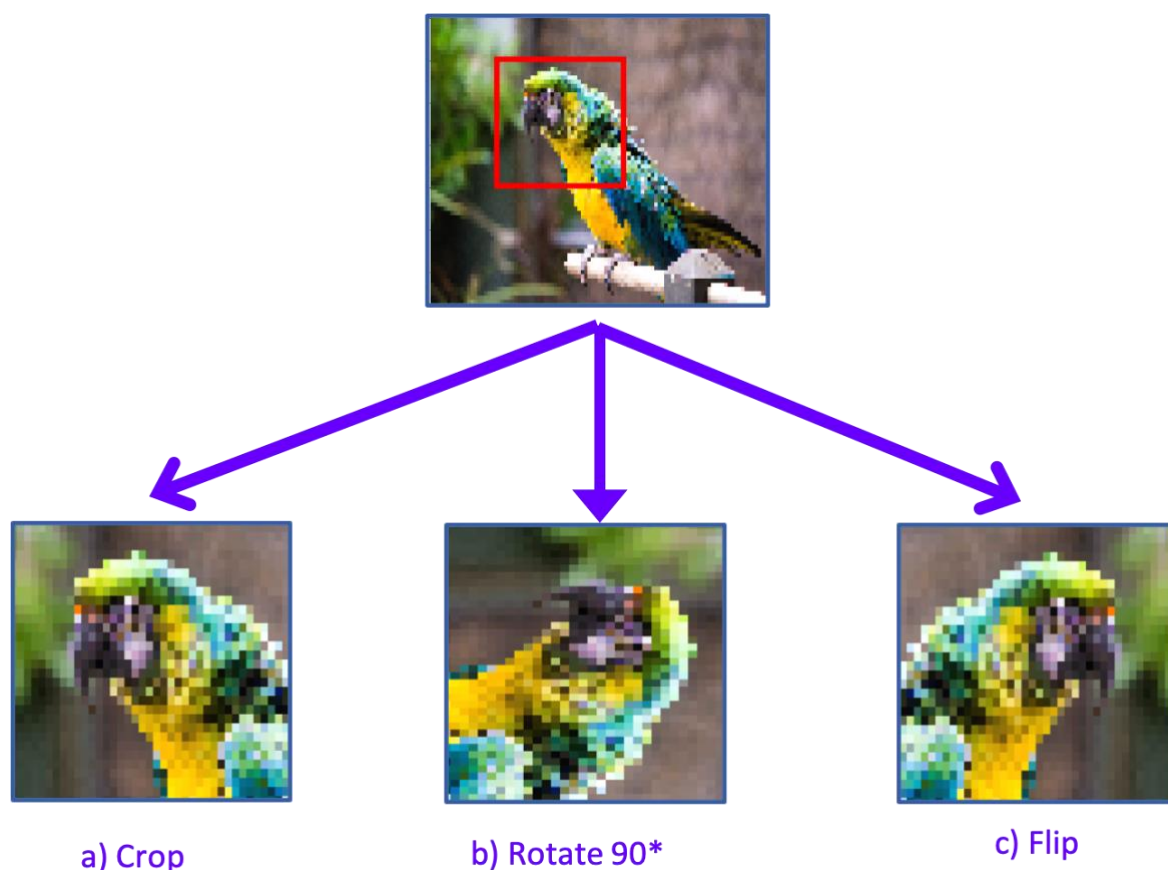


**Hình 12.** Cấu trúc mô hình mạng EDSR

### 3.3. Phương pháp xử lý đầu vào:

Vì tập dữ liệu huấn luyện là những hình ảnh có độ phân giải lên đến 2K, nếu sử dụng toàn bộ hình ảnh trong một lần số lượng thông số sẽ là quá lớn cho quá trình huấn luyện vì thế ta cần xử lý bằng cách cắt nhỏ ngẫu nhiên hình ảnh thành những hình ảnh có kích thước  $w * h$  (px). Tuy nhiên số lượng hình ảnh chỉ có 800 ảnh là một con số khá nhỏ vì thế ta tiến hành tăng dữ liệu bằng các cách xoay hình ảnh hoặc là lật hình ảnh.

Tại Hình 13 ta có thể thấy từ mỗi hình ảnh ta có thể biến đổi theo 3 cách là Cắt ngẫu nhiên (Crop), Xoay hình ảnh (Rotate 90°) và Lật hình ảnh (Flip). Khi phối hợp cả 3 phương pháp trên và áp dụng vào bộ dữ liệu đầu vào ta sẽ thu được một bộ dữ liệu mới dựa trên những hình ảnh ban đầu.



**Hình 13.** Cách phương pháp biến đổi hình ảnh đầu vào

### 3.4. Hàm mất mát (Loss Function)

Hàm mất mát trong học sâu là một hàm tính toán tương quan giữa kết quả đầu ra  $\hat{I}$  so với hình ảnh thật  $I$ . Trong bài toán Super-resolution hàm mất mát được sử dụng để đo lường thất thoát khi tái cấu trúc hình ảnh qua đó hỗ trợ giúp mô hình trở nên tối ưu hơn.

Những hàm mất mát ban đầu được sử dụng đối với bài toán SISR là Mean Square Error (L2), Mean Absolute Error (L1). Đây là hai hàm mất mát cơ bản của những mô hình học sâu. Hai hàm mất mát trên cho ra những kết quả rất khả quan trong một số trường hợp. Tuy nhiên đối với bài toán có kết quả và thực tế (growth truth) đều là hình ảnh L1, L2 trở nên yếu thế vì không thể đo lường chất lượng hình ảnh sau khi tái tạo tại một cách chính xác nhất. Vì thế những hàm mất mát mới được sử dụng như Content Loss[1] [14], Adversarial Loss [15]...

$$L2 = \frac{1}{N} \sum_{i=1}^N (I_i - \hat{I}_i)^2 \quad (2)$$

$$L1 = \frac{1}{N} \sum_{i=1}^N |(I_i - \hat{I}_i)| \quad (3)$$

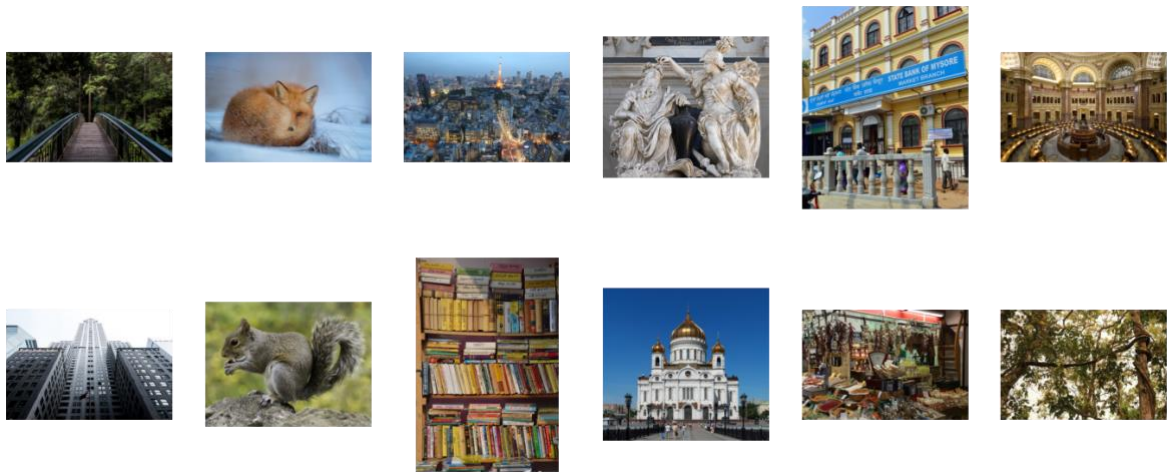
Trong phần thực nghiệm kế tiếp tôi sẽ sử dụng L1, L2 làm hàm mất mát cho bài toán.

## 4. THỰC NGHIỆM:

### 4.1. Tập dữ liệu:

#### 4.1.1. Dữ liệu huấn luyện - DIV2K:

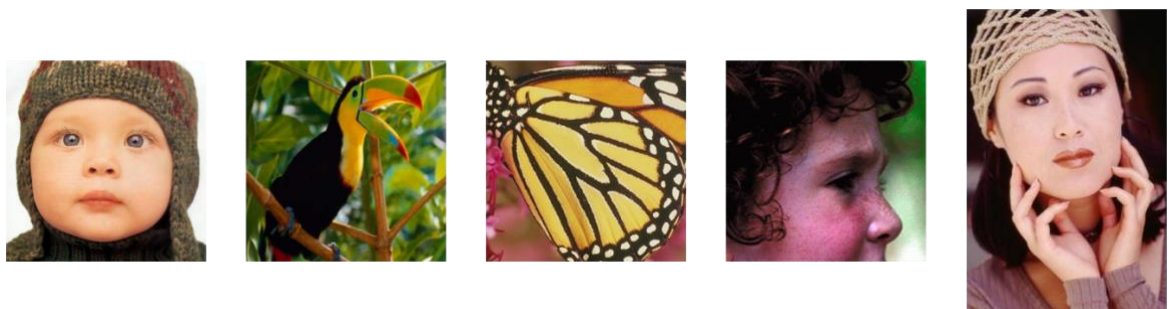
Được giới thiệu lần đầu tại cuộc thi NTIRE 2017 Challenge on Single Image Super-Resolution: Dataset and Study [5] bởi *Eirikur Agustsson et al.* Tập dữ liệu bao gồm 1000 hình ảnh khác nhau có độ phân giải 2K được chia ra thành 800 hình ảnh cho tập huấn luyện, 100 hình ảnh cho tập kiểm thử và 100 hình ảnh cho tập kiểm tra. Tập dữ liệu này chứa hai kiểu giảm độ phân giải khác nhau gồm bicubic, unknown và có nhiều độ phóng to khác nhau từ x2, x3, x4, x8 và được cập nhật qua các cuộc thi NTIRE. Đây là một trong những tập dữ liệu tiêu biểu cho quá trình huấn luyện của bài toán SISR. Trong bài viết này tôi sẽ sử dụng tập huấn luyện của bộ dữ liệu DIV2K làm dữ liệu cho quá học của các mô hình.



**Hình 14.** Tập dữ liệu DIV2K

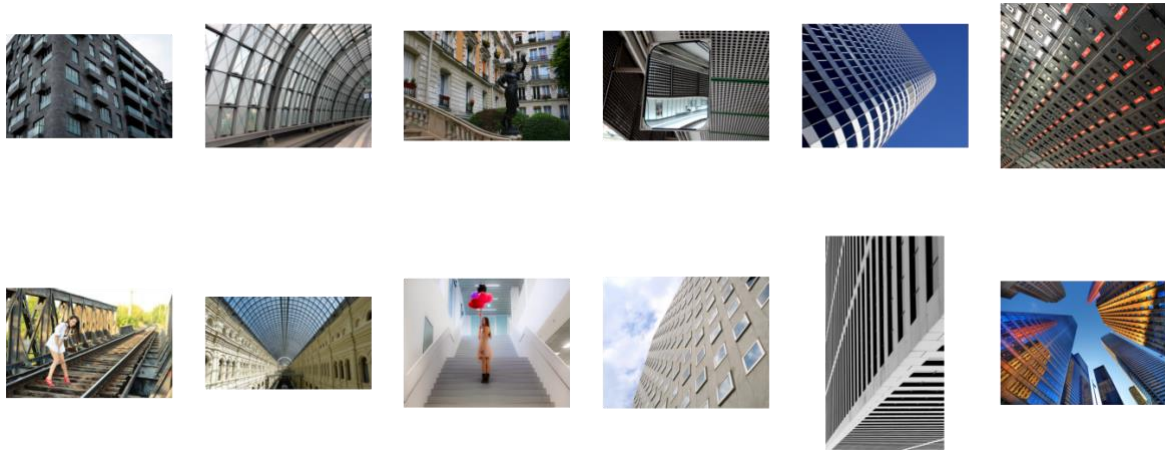
#### 4.1.2. Dữ liệu kiểm tra:

**Set 5** [6]: Được giới thiệu bởi *Marco Bevilacqua et al.* dữ liệu bao gồm 5 hình ảnh (“baby”, “bird”, “butterfly”, “head”, “woman”) thường được sử dụng để kiểm tra hiệu suất của các mô hình SISR



**Hình 15.** Tập dữ liệu Set5

**Urban100** [7]: Của tác giả *Huang et al.* gồm 100 hình ảnh đô thị cũng là một trong những tập dữ liệu được sử dụng rất nhiều cho bài toán SISR.



**Hình 16.** Tập dữ liệu Urban100

#### 4.2. Phương pháp đánh giá

Trong suốt quá trình huấn luyện mô hình tôi sẽ sử dụng Peak Signal-to-Noise Ratio (PSNR) để làm phương pháp đánh giá trong quá trình huấn luyện. PSNR là một hàm dùng để xác định chất lượng của kết quả hình ảnh được tính dựa trên  $L_2$ . PSNR có giá trị cực tiểu bằng 0, giá trị PSNR càng cao thì kết quả càng tốt. Công thức được trình bày như sau:

$$PSNR = 10 \cdot \log_{10} \left( \frac{L^2}{L_2} \right) \quad (4)$$

Sau khi huấn luyện mô hình tôi sẽ sử dụng phương pháp Structural Similarity (SSIM) hay còn được gọi là độ tương quan cấu trúc. Phương pháp này sử dụng đồng thời 3 thông số là độ sáng, độ tương phản và cấu trúc để đánh giá hai hình ảnh có tương đồng hay không. SSIM có giá trị chạy từ 0 đến 1 với giá trị bằng 1 có nghĩa là hai hình ảnh hoàn toàn tương đồng và ngược lại với 0

#### 4.3. Môi trường thực nghiệm

Ở đây tôi sử dụng ngôn ngữ python làm ngôn ngữ để lập trình, dùng các thư viện hỗ trợ như tensorflow, numpy trong việc xây dựng mô hình cũng như xử lý dữ liệu. Dùng thư viện matplotlib và seaborn để hiển thị kết quả cũng như thông số của quá trình huấn luyện.

Về môi trường tôi sử dụng Google Colaboratory gọi tắt là Colab làm môi trường thực nghiệm. Colab là một máy ảo dựa trên hệ điều hành linux được Google phát triển giúp người dùng có thể viết và xử lý code trên nền tảng của Jupyter Notebook. Colab được phát triển cho mục đích nghiên cứu vì thế Google đã hỗ trợ người dùng bằng cách cung cấp GPU và TPU giúp việc tính toán cũng như huấn luyện các mô hình trở nên nhanh chóng hơn.

#### 4.4. Chi tiết thực nghiệm

Mục tiêu của quá trình thực nghiệm là kiểm tra mối quan hệ giữa lớp Upsampling với những mô hình học sâu cho bài toán SISR. Tôi sẽ thí nghiệm và so sánh kết quả giữa những mô hình có lớp Upsampling khác nhau. Từ đó kết luận mối quan hệ của chúng.

**Bảng 1.** Bảng danh sách thí nghiệm

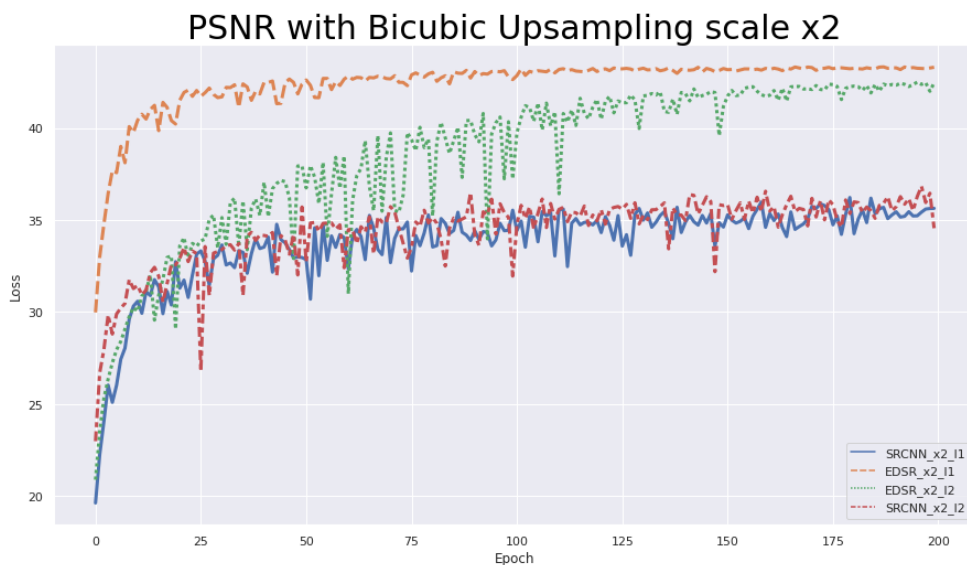
STT	Tên thí nghiệm
1	Thí nghiệm với lớp Upsampling Bicubic
2	Thí nghiệm với lớp Upsampling Pixel-Shuffle

Theo Bảng 1, Tôi sẽ thực nghiệm huấn luyện 2 mô hình với lớp Upsampling Bicubic và lớp Upsampling Pixel-Shuffle. Sử dụng 2 hàm mất mát là L1 và L2 để chọn được phương pháp tối ưu nhất để đánh giá kết quả.

Phương pháp: đầu tiên tôi xử lý hình ảnh và thu được đầu vào là những hình ảnh có kích thước 48x48 và được truyền lần lượt theo bố, mỗi bố gồm 16 hình ảnh. Tôi sử dụng hàm tối ưu là Adam với hệ số học ban đầu (init learning rate) là:  $1e - 3$ . Trong quá trình huấn luyện hệ số học sẽ giảm  $e^{-0.1}$  nếu giá trị hàm mất mát không giảm sau 3 epochs. Các trọng số của quá trình sẽ được lưu trữ lại nếu có cải thiện. Cuối cùng mô hình sau khi huấn luyện và các chỉ số trong suốt quá trình học sẽ được lưu trữ lại. Tôi sẽ truyền trọng số tốt nhất và tiến hành đánh giá trên tập đánh giá.

#### 4.5. Thực nghiệm với Upsampling Bicubic

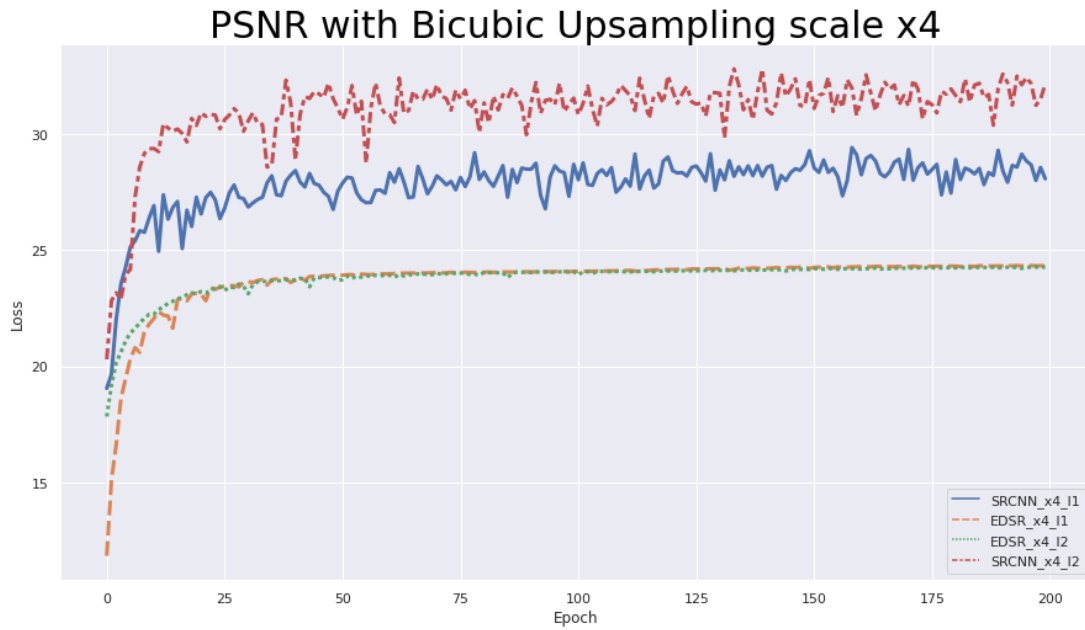
Tôi tiến hành làm thực nghiệm cả hai mô hình với lớp Upsampling là Bicubic. Quá trình huấn luyện là 200 epochs với mỗi hàm loss và mỗi tỉ lệ hình ảnh khác nhau sẽ cho một kết quả.



**Hình 17.** PSNR với Bicubic Upsampling và tỉ lệ x2

Ở tỉ lệ x2 giá trị PSNR của mô hình EDSR với hàm mất mát là L1 rất cao. Độ hội tụ ban đầu rất nhanh khoảng 25 epochs đầu sau đó thấp và ổn định dần, xem tại Hình 17. Ngược lại ở tỉ lệ hình ảnh x4 thì mô hình SRCNN với hàm mất mát là L2 cho ta kết quả trên tập kiểm thử tốt hơn tuy nhiên độ ổn định của mô hình không đảm bảo Hình 18.





**Hình 18.** PSNR với Bicubic Upsampling và tỉ lệ x4

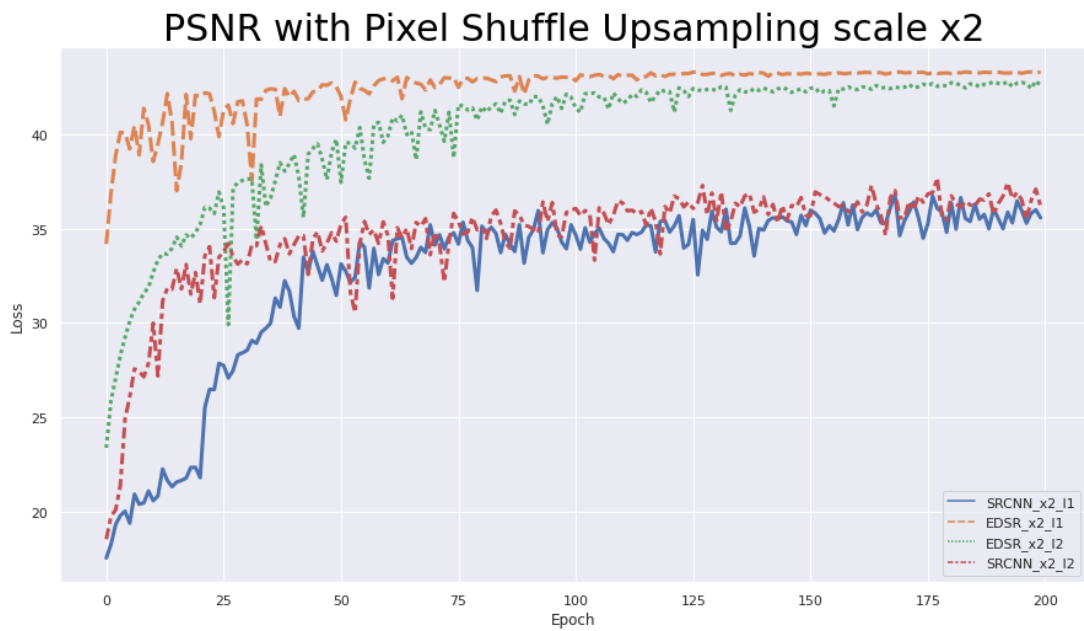
**Kiểm tra:** Khi sử dụng lớp Upsampling là Bicubic mô hình tốt nhất là EDSR với loss L1 khi tỉ lệ hình ảnh đầu vào là x2 PSNR = 33.3492 (Bảng 2). Khi đầu vào có tỉ lệ là x4 thì mô hình EDSR với loss L1 vẫn cho ra kết quả tốt nhất. PSNR = 27.929, tuy nhiên mô hình SRCNN và EDSR với loss L2 cũng cho ra kết quả rất tốt 27.182, 27.449.

**Bảng 2.** Kết quả thực nghiệm với lớp Upsampling Bicubic

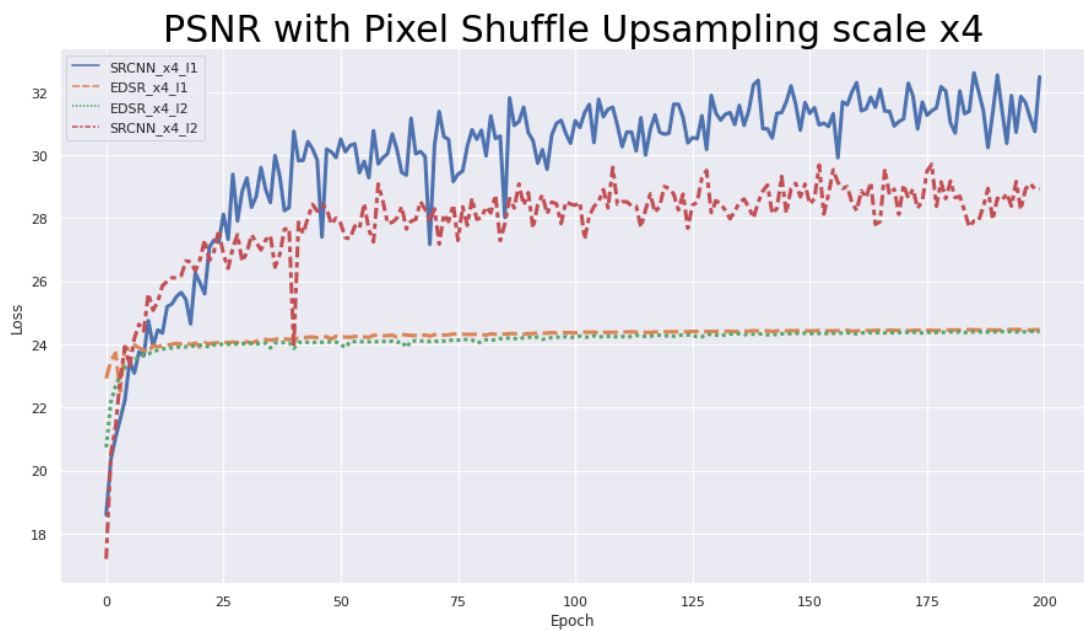
Scale	Model	L2		L1	
		Loss ↓	PSNR ↑	Loss ↓	PSNR ↑
X2	SRCNN	64.049	32.263	4.13	32.227
	EDSR	63.058	32.4142	<b>3.707</b>	<b>33.3492</b>
X4	SRCNN	186.657	27.182	9.67	24.664
	EDSR	179.52	27.449	<b>6.654</b>	<b>27.929</b>

#### 4.6. Thực nghiệm với Upsampling Pixel Shuffle

Tương tự với phương pháp thực nghiệm với lớp Upsampling là Bicubic ở lần thực nghiệm này tôi sử dụng lớp Pixel Shuffle làm lớp Upsampling cho mô hình. Với tập dữ liệu có tỉ lệ x2 EDSR cho thấy kết quả trên tập huấn luyện rất tốt Hình 19. Tuy nhiên khi tiến hành trên tập dữ liệu có tỉ lệ x4 thì kết quả cho ra không cao không thay đổi nhiều trong suốt quá trình huấn luyện Hình 20.



**Hình 19.** PSNR với lớp Pixel Shuffle Upsamling và tỉ lệ x2



**Hình 20.** PSNR với lớp Pixel Shuffle Upsamling và tỉ lệ x4

**Kiểm tra:** Tuy rằng trong quá trình huấn luyện mô hình EDSR với tập dữ liệu x4 không thay đổi nhiều nhưng kết quả trên tập kiểm tra cho ta thấy khi sử dụng lớp Upsamling Pixel Shuffle mô hình có kết quả cao nhất là EDSR với loss L1 28.285 xem tại Bảng 3. Ở tỷ lệ hình ảnh x2 cũng vậy EDSR với loss L1 cho ra kết quả đánh giá cao nhất là 33.5688.

**Bảng 3.** Kết quả thực nghiệm với lớp Upsampling Pixel Shuffle




Scale	Model	L2		L1	
		Loss ↓	PSNR ↑	Loss ↓	PSNR ↑
<b>X2</b>	<b>SRCNN</b>	58.999	32.623	4.089	32.39
	<b>EDSR</b>	55.3173	32.9665	<b>3.6098</b>	<b>33.5688</b>
<b>X4</b>	<b>SRCNN</b>	270.005	25.215	6.879	27.533
	<b>EDSR</b>	158.3279	28.0993	<b>6.374</b>	<b>28.285</b>

#### 4.7. Đánh giá trên các tập kiểm tra.

Sau hai thực nghiệm trên cho ta thấy rằng 2 mô hình sẽ cho ra kết quả tương đối tốt khi sử dụng loss L1. Và từ Bảng 2 và Bảng 3 ta có thấy được rằng khi sử dụng lớp Upsampling Pixel Shuffle kết quả đánh giá cao hơn lớp Upsampling Bicubic 33.5688 với tỉ lệ x2 và 28.285 đối với tỉ lệ x4

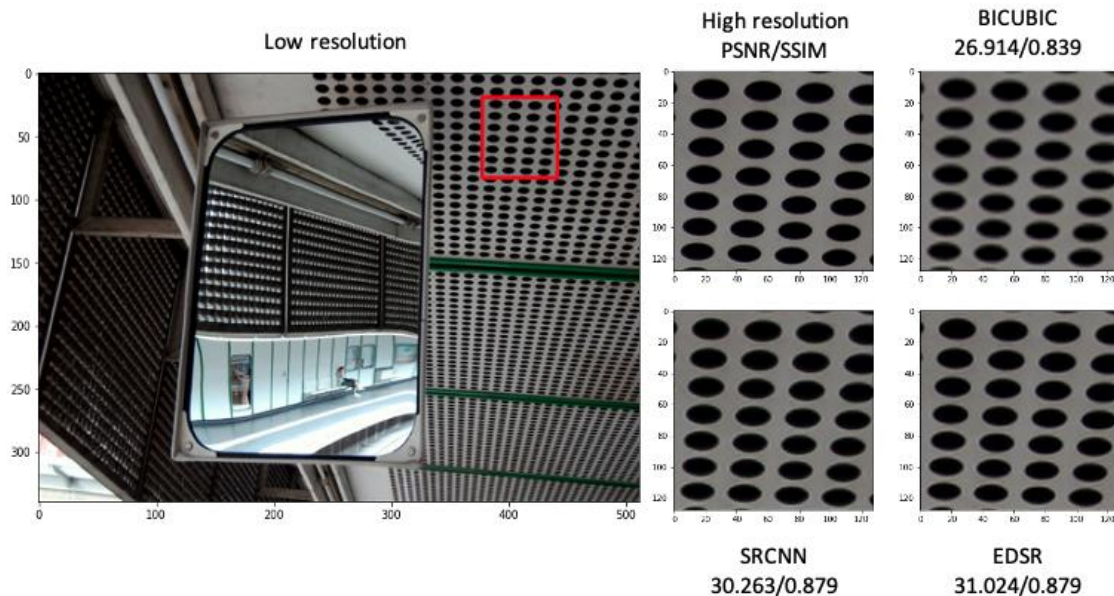
Vì thế ta tiến hành kiểm tra 2 mô hình sử dụng lớp Upsampling Pixel-Shuffle với loss L1 kiểm tra trên 3 tập dữ liệu là Set5, Urban100 và DIV2K (Valid Set)

**Bảng 4.** Bảng đánh giá trên các tập kiểm tra

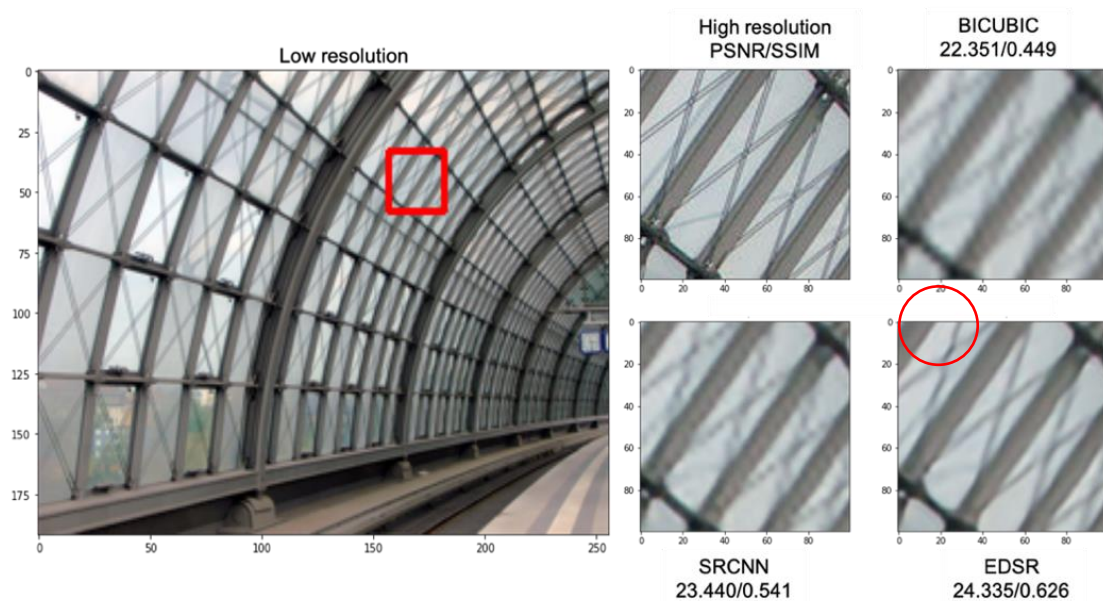
Dataset	Scale	BICUBIC 	SRCNN 	EDSR 
<b>Set5</b>	<b>x2</b>	23.2937	33.5029	<b>34.9619</b>
	<b>x4</b>	28.5764	28.0139	<b>29.1378</b>
<b>Urban100</b>	<b>x2</b>	21.7142	26.8515	<b>28.3631</b>
	<b>x4</b>	22.9085	22.5235	<b>23.4222</b>
<b>DIV2K</b>	<b>x2</b>	24.664	32.3902	<b>33.5613</b>
	<b>x4</b>	27.929	26.8611	<b>28.285</b>

Qua Bảng 4 ta có thể kết luận được rằng khi sử dụng mô hình EDSR với lớp Upsampling là Pixel-Shuffle sẽ cho ta kết quả tốt nhất và khi sử dụng với tập dữ liệu Set5 và DIV2K kết quả đạt được rất cao.

Dưới đây là một vài hình thực nghiệm trên tập dữ liệu Urban100 và DIV2K(Valid Set). Tại Hình 21 so sánh kết quả trên tập dữ liệu Urban ta có thể thấy EDSR, và SRCNN đã có thể tái tạo rất tốt hình ảnh vượt trội hơn so với nội suy Bicubic, những chi tiết hình tròn rõ nét. Hình 22 là kết quả thực nghiệm trên tập dữ liệu Urban100 với tỉ lệ scale là x4 ta có thể nhận thấy rằng EDSR đã làm việc rất hiệu quả đặc biệt đối với vùng cạnh, hình ảnh tái tạo rõ ràng hơn so với 2 phương pháp còn lại hình ảnh tương đối mờ.

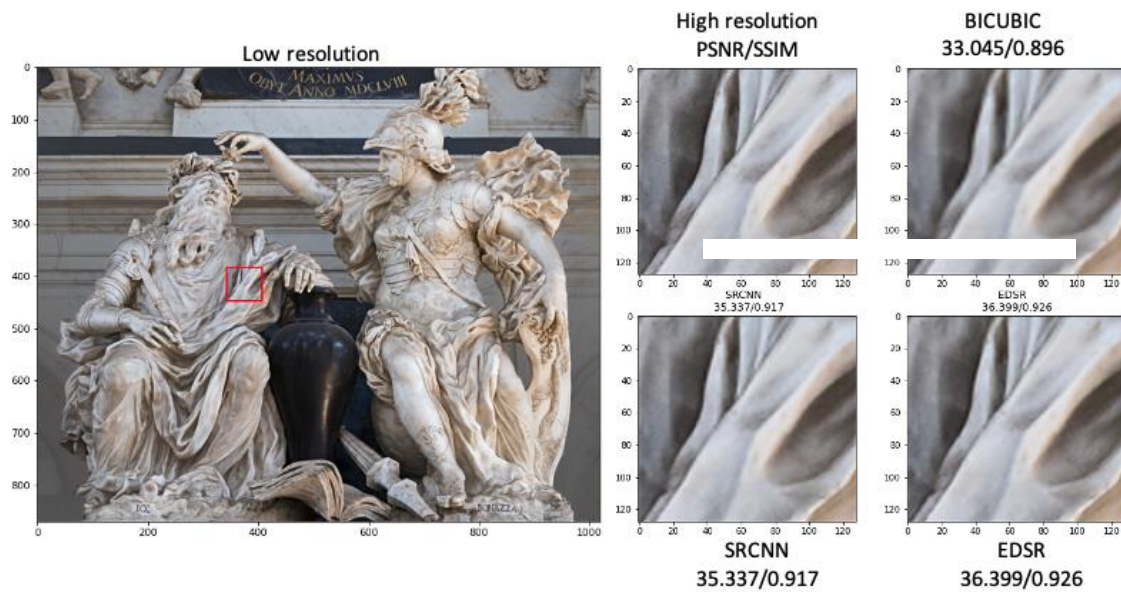


**Hình 21.** Đánh giá trên tập Urban100 - x2 scale

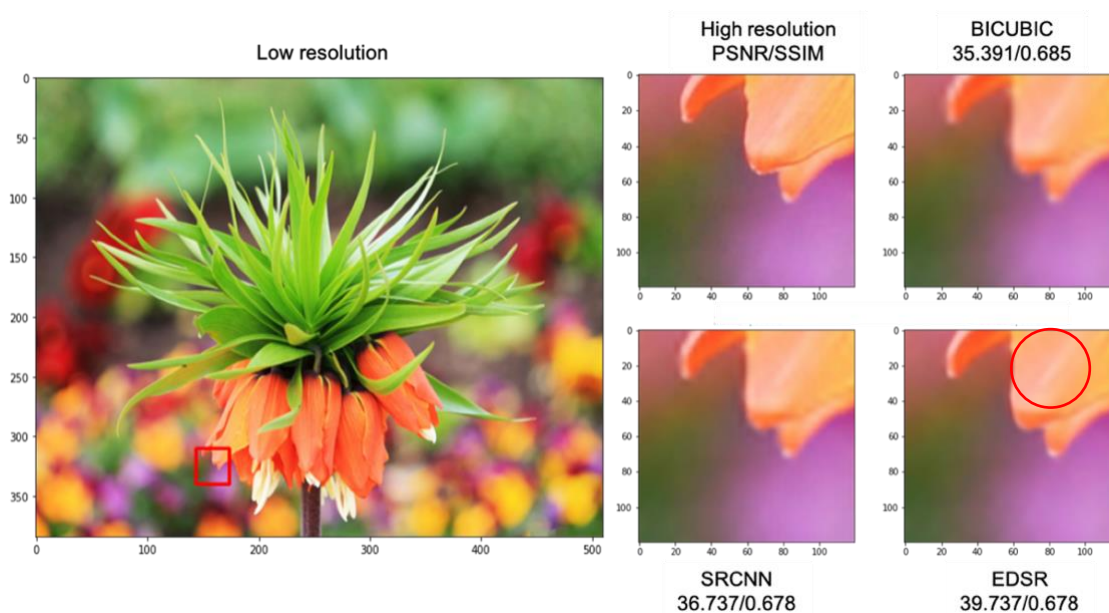


**Hình 22.** Đánh giá trên tập Urban100 - x4 scale





**Hình 23.** Tái tạo hình ảnh từ tập Div2k - scale x2



**Hình 24.** Tái tạo hình ảnh từ tập Div2k - scale x4

#### 4.8. Triển khai trên ứng dụng thử nghiệm

Cuối cùng sau các trình thực nghiệm và đánh giá để chọn ra mô tốt nhất, tôi tiến hành triển khai mô hình lên ứng dụng mạng (Web application). Tôi sử dụng Dash, một thư viện được viết trên framework Flask. Giúp tôi có thể xử lý backend bằng ngôn ngữ Python. Cùng với đó, tôi sử dụng Tailwindcss để phát triển giao diện cho ứng dụng. Khi sử dụng chúng ta chỉ cần tải hình ảnh lên, chọn kích thước tỉ lệ và chọn vùng ảnh cần nâng cao độ phân giải. Ứng dụng sẽ xử lý bằng các mô hình ở phần thực nghiệm và cho ra kết quả. Hình 25 là toàn bộ giao diện của ứng dụng. Tại hộp "Drag and Drop or Select file" là nơi chúng ta tải hình ảnh lên, sau đó chúng ta chọn tỉ lệ hình ảnh tại "Select Scale", tiếp theo ta chỉ cần kéo thả khu vực mong muốn ở hình "Low Resolution". Tất cả kết quả và hình ảnh ban đầu sẽ xuất hiện bên dưới dòng "Predict".





**Hình 25.** Giao diện ứng dụng thử nghiệm

## 5. KẾT LUẬN

Qua các quá trình thực nghiệm ta đã biết được mô hình SRCNN và EDSR khi sử dụng lớp Upsampling Pixel Shuffle đã cho kết quả tốt nhất, đặc biệt là mô hình EDSR. Trong các tập dữ liệu dùng để kiểm tra ta thấy rằng tập dữ liệu Set5 và DIV2K là 2 tập dữ liệu đạt kết quả cao nhất (xem Bảng 4). Trên tập dữ liệu Urban100 tuy điểm đánh giá không cao bằng 2 tập còn lại vì tập dữ liệu quá khác biệt so với tập huấn luyện, chủ yếu có cấu trúc của đô thị cũng như các tòa nhà. Nhưng khi hiển thị kết quả ta thấy rằng hình ảnh đã được tái tạo tương đối tốt qua đó ta thấy được độ hiệu quả của mô hình EDSR là rất tốt.

Trong bài báo này tôi đã triển khai và thực nghiệm 2 mô hình SRCNN, EDSR và đã thực nghiệm trên 2 lớp Upsampling khác nhau. Tôi cũng đã phát triển một ứng dụng mạng

cho bài toán này. Tuy nhiên tôi chỉ dừng lại ở việc lấy dữ liệu đầu vào với một tỉ lệ duy nhất. Trong tương lai tôi có thể phát triển thêm về mặt dữ liệu đầu vào nhiều tỉ lệ hơn (multi-scale input), cũng như phát triển những mô hình cho bài toán riêng biệt như Super-Resolution cho hình ảnh y học hoặc cho hình ảnh thiên văn học...

## TÀI LIỆU THAM KHẢO

- [1] Z. Wang, J. Chen, and S. C. H. H. Hoi, “Deep learning for image super-resolution: A survey,” *IEEE Trans Pattern Anal Mach Intell*, vol. 43, no. 10, pp. 3365–3387, 2020, doi: 10.1109/TPAMI.2020.2982166.
- [2] I. J. Goodfellow *et al.*, “Generative Adversarial Nets.” [Online]. Available: <http://www.github.com/goodfeli/adversarial>
- [3] C. Dong, C. C. Loy, K. He, and X. Tang, “Image super-resolution using deep convolutional networks,” *IEEE Trans Pattern Anal Mach Intell*, vol. 38, no. 2, pp. 295–307, 2015, doi: 10.1109/TPAMI.2015.2439281.
- [4] B. Lim, S. Son, H. Kim, S. Nah, K. M. Lee, and K. Mu Lee, “Enhanced deep residual networks for single image super-resolution,” in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2017, vol. 2017-July, pp. 136–144. doi: 10.1109/CVPRW.2017.151.
- [5] E. Agustsson and R. Timofte, “Ntire 2017 challenge on single image super-resolution: Dataset and study,” in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2017, pp. 126–135.
- [6] M. Bevilacqua, A. Roumy, C. Guillemot, and M. A. Morel, “Low-Complexity Single-Image Super-Resolution based on Nonnegative Neighbor Embedding,” in *Proceedings of the British Machine Vision Conference*, 2012, pp. 135.1--135.10. doi: <http://dx.doi.org/10.5244/C.26.135>.
- [7] J.-B. Huang *et al.*, “Single image super-resolution from transformed self-exemplars,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, vol. 52, no. 10, pp. 5197–5206.
- [8] S. Hitawala, Y. Li, X. Wang, and D. Yang, “Image super-resolution using VDSR-ResNeXt and SRCGAN,” *arXiv preprint arXiv:1810.05731*, 2018.
- [9] L. S. Passarella, S. Mahajan, A. Pal, and M. R. Norman, “Reconstructing High Resolution ESM Data Through a Novel Fast Super Resolution Convolutional Neural Network (FSRCNN),” *Geophysical Research Letters*, vol. 49, no. 4, p. e2021GL097571, 2022.
- [10] W. Shi *et al.*, “Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network,” in *Proceedings of the IEEE*

*conference on computer vision and pattern recognition*, 2016, vol. 2016-Decem, pp. 1874–1883. doi: 10.1109/CVPR.2016.207.

[11] M. Haris, G. Shakhnarovich, and N. Ukita, “Deep back-projection networks for super-resolution,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1664–1673.

[12] Z. Li, J. Yang, Z. Liu, X. Yang, G. Jeon, and W. Wu, “Feedback network for image super-resolution,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, vol. 2019-June, pp. 3867–3876. doi: 10.1109/CVPR.2019.00399.

[13] X. Hu, H. Mu, X. Zhang, Z. Wang, T. Tan, and J. Sun, “Meta-SR: A magnification-arbitrary network for super-resolution,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, vol. 2019-June, pp. 1575–1584. doi: 10.1109/CVPR.2019.00167.

[14] J. Johnson, A. Alahi, and L. Fei-Fei, “Perceptual losses for real-time style transfer and super-resolution,” in *European conference on computer vision*, 2016, vol. 9906 LNCS, pp. 694–711. doi: 10.1007/978-3-319-46475-6\_43.

[15] C. Ledig *et al.*, “Photo-realistic single image super-resolution using a generative adversarial network,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4681–4690.