

# Image Super-Resolution

Tuan Phan, Tung Pham  
AI For Everyone

Ngày 8 tháng 3 năm 2022

# Outline

---

## 1. Bài toán image super-resolution

## 2. Image Interpolation

- 2.1 Nearest-neighbor interpolation
- 2.2 Bilinear interpolation
- 2.3 Bicubic interpolation

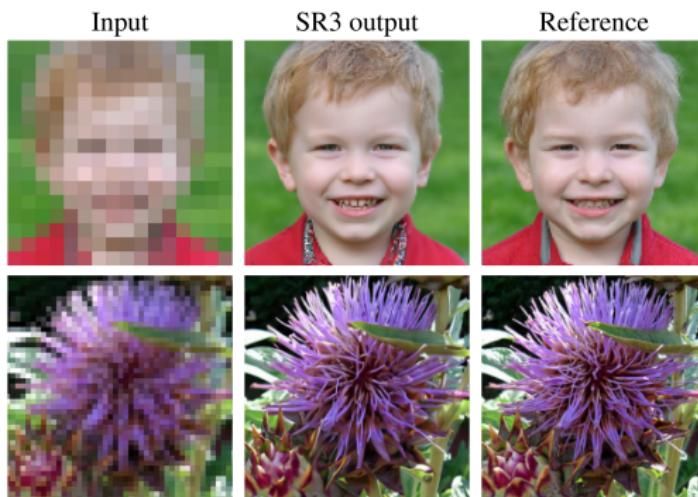
## 3. CNN for Super-Resolution

- 3.1 SRCNN
- 3.2 ResNet Super Resolution
- 3.3 Autoencoder Super Resolution

## 4. Tổng kết

# Bài toán image super-resolution

Cho ảnh  $X$  là một ảnh high-resolution (HR),  $Y$  là ảnh low-resolution (LR) tương ứng của  $X$ . Gọi hàm dùng để upscale ảnh LR thành HR là  $F$ . Bài toán image super-resolution hướng đến việc tìm  $F$  sao cho  $F(Y) \approx X$ . Hay ảnh upscale từ  $Y$  giống  $X$  nhất có thể



Hình 1: Ví dụ về bài toán image super-resolution

# Dataset

---

Bộ dữ liệu sử dụng: DIV2K

DIV2K bao gồm 1000 ảnh 2K trong đó:

- 800 ảnh cho training
- 100 ảnh cho validation
- 100 ảnh cho testing

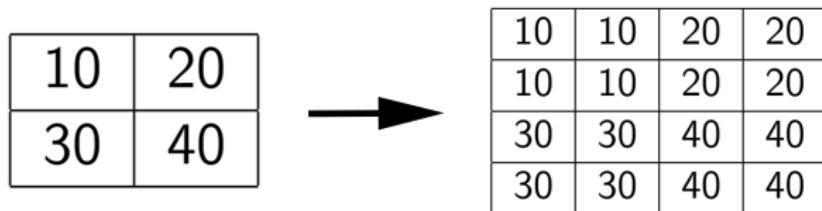
**Nguồn:** <https://data.vision.ee.ethz.ch/cvl/DIV2K/>

# Image Interpolation

# Image Interpolation

---

- Nearest-neighbor interpolation: là phương pháp đơn giản nhất. Trong phương pháp này, các pixel trong ảnh  $F(Y)$  sẽ dùng giá trị của pixel trong ảnh  $Y$  gần nó nhất.



Hình 2: Nearest-neighbor interpolation

# Image Interpolation

---

- Bilinear interpolation: phương pháp này sẽ nội suy giá trị của một pixel bằng cách tính trung bình có trọng số 4 (2x2) pixel lân cận.

10	20		
30	40		

→

10	12.5	17.5	20
15	17.5	22.5	25
25	27.5	32.5	35
30	32.5	37.5	40

Hình 3: Bilinear interpolation

# Image Interpolation

---

- Bicubic interpolation: phương pháp này, các pixel trong ảnh  $F(Y)$  sẽ dùng giá trị của pixel trong ảnh  $Y$  gần nó nhất.

10	20		
30	40		

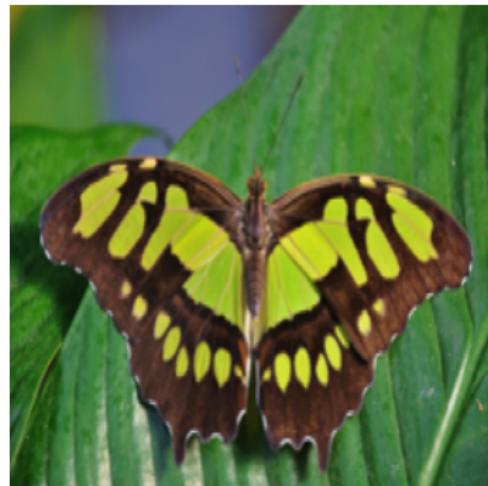
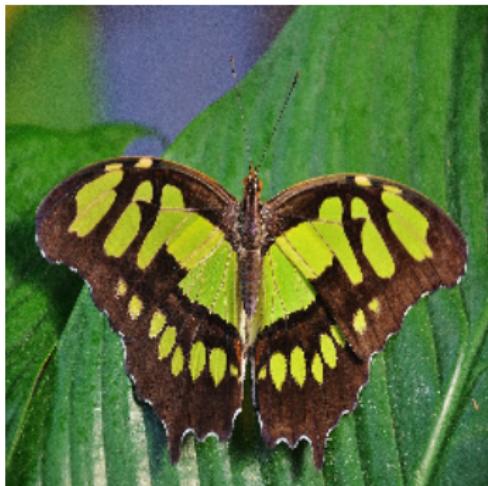
→

6.84	10.16	15.63	18.95
13.48	16.80	22.27	25.59
24.41	27.73	33.20	36.52
31.05	34.38	39.84	43.16

Hình 4: Bicubic interpolation

# Image Interpolation

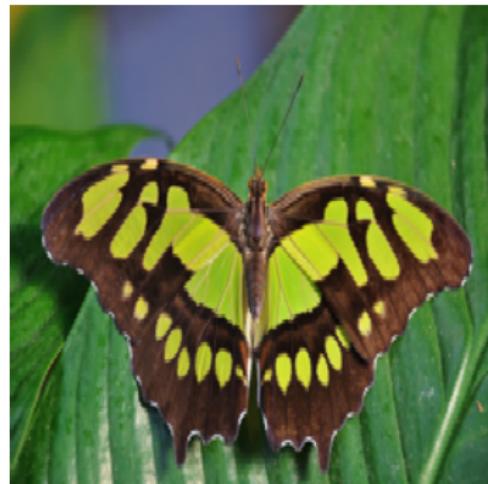
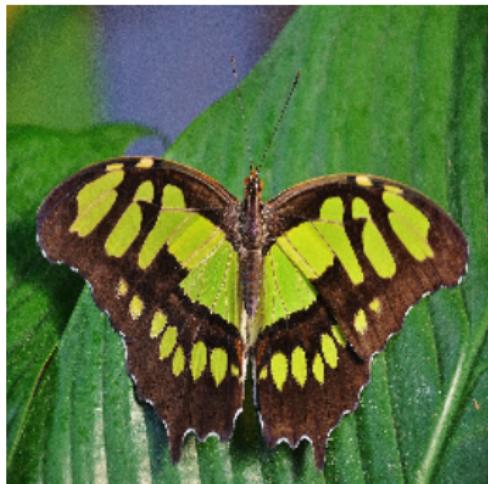
---



Hình 5: Ảnh high-resolution và low-resolution

# Image Interpolation

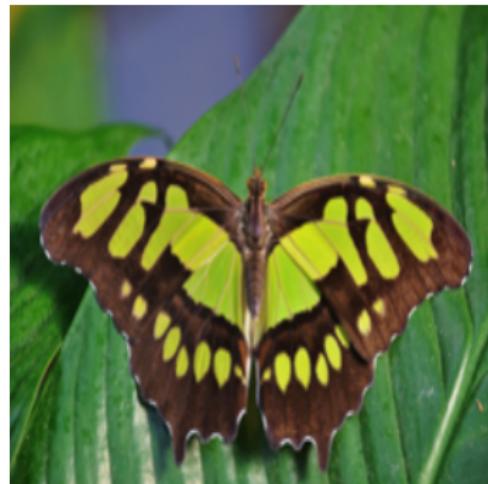
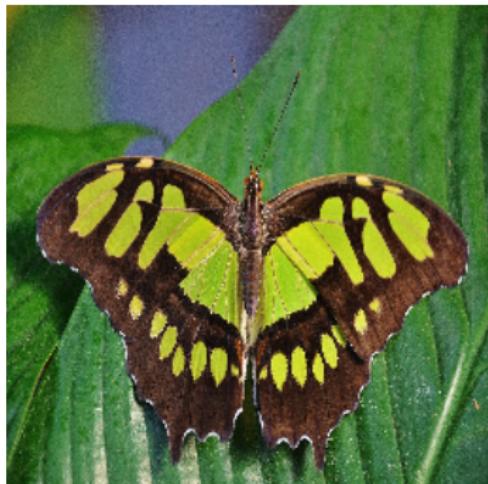
---



Hình 6: Ảnh high-resolution và Nearest-neighbor interpolation

# Image Interpolation

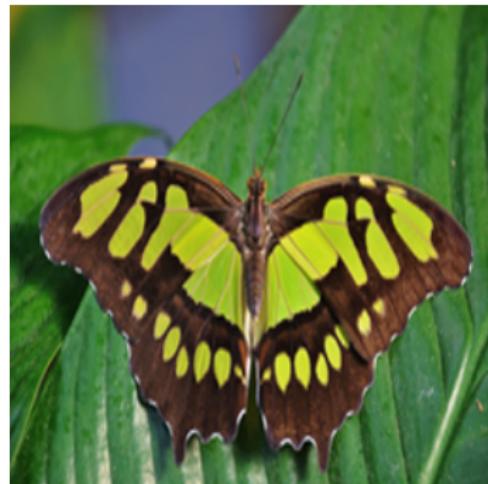
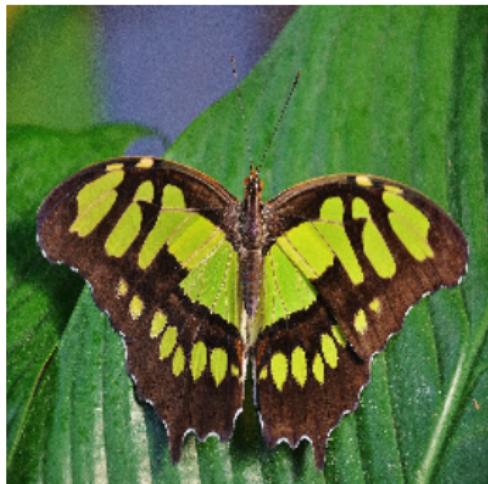
---



Hình 7: Ảnh high-resolution và Bilinear interpolation

# Image Interpolation

---

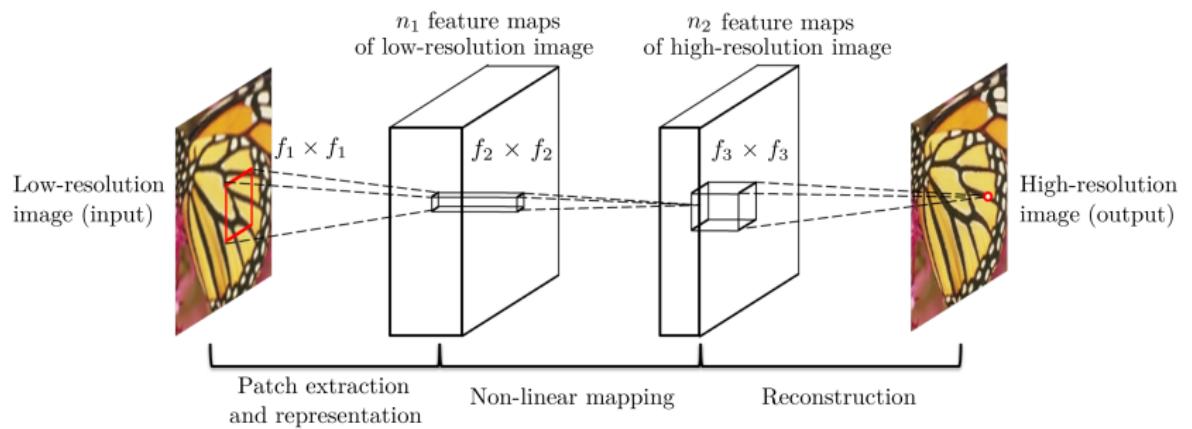


Hình 8: Ảnh high-resolution và Bicubic interpolation

# CNN for Super-Resolution

# **SRCNN**

# Kiến trúc mạng



Hình 9: Kiến trúc mạng SRCNN

# Patch extraction and representation

---

Lấy các patch overlap nhau ở input bằng cách cho một kernel trượt trên ảnh đầu vào. Số lượng feature map sẽ tương ứng với số chiều của vector. Phép toán được thực hiện ở layer đầu tiên là:

$$F_1(\mathbf{Y}) = \max(0, W_1 \circledast \mathbf{Y} + B_1)$$

Trong đó:

- $W_1$  là filter, kích thước  $f_1 \times f_1 \times c$ , có  $n_1$  filter như này.
- $B_1$  là bias
- $\circledast$  là phép conv

Output của layer này là một vector  $n_1$  chiều tương ứng với  $n_1$  feature map.

# Non-linear mapping

---

Ánh xạ vector  $n_1$  chiều sang vector  $n_2$  chiều

$$F_2(\mathbf{Y}) = \max(0, W_2 \circledast F_1(\mathbf{Y}) + B_2)$$

Trong đó:

- $W_2$  là filter, kích thước  $f_2 \times f_2 \times n_1$ , có  $n_2$  filter như này.
- $B_2$  là bias

**Câu hỏi đặt ra:** Có nên thêm nhiều layer này để tăng tính non-linear?

# Reconstruction

---

Phục hồi ảnh từ vector  $n_2$  chiều

$$F_3(\mathbf{Y}) = W_3 \circledast F_2(\mathbf{Y}) + B_3$$

Trong đó:

- $W_3$  là filter, kích thước  $n_2 \times f_3 \times n_3$ , có  $c$  filter như này.
- $B_3$  là bias

# Training

---

Loss function:

$$L(\Theta) = \frac{1}{n} \sum_{i=1}^n \|F(\mathbf{Y}_i; \Theta) - \mathbf{x}_i\|^2$$

Với n là số sample được dùng để train

# Kiến trúc SRCNN

---

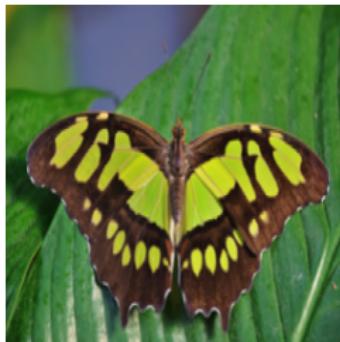
Block	Đầu vào	Phép toán	Số channel
$X_1$	(Input)	Interpolation	3
$X_2$	256x256x3	Conv2D	64
$X_3$	256x256x64	Batch Normalization	
$X_4$	256x256x64	Conv2D	32
$X_5$	256x256x32	Batch Normalization	
$X_6$	256x256x32	Conv2D	3

Bảng 1: Kiến trúc SRCNN

**Tổng tham số:** 20,483

# Kết quả

---



Hình 10: Ảnh high-resolution, low-resolution và output SRCNN

# ResNet Super Resolution

# Kiến trúc SRResNet

---

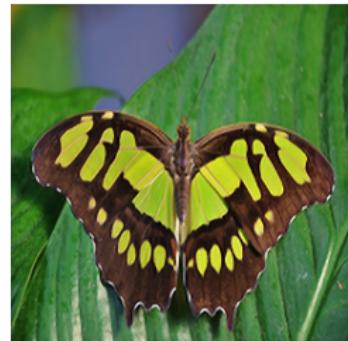
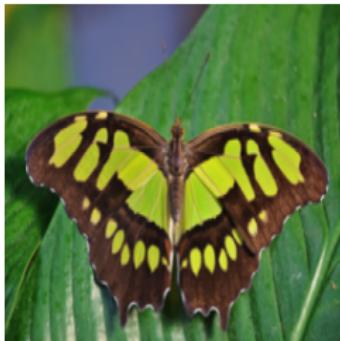
Block	Đầu vào	Phép toán	Số channel
$X_1$	(Input)	Interpolation	3
$X_2$	256x256x3	Conv2D	64
$X_3$	256x256x64	residual_block	64
$X_4$	256x256x64	residual_block	64
$X_5$	256x256x64	residual_block	64
$X_6$	256x256x64	residual_block	64
$X_7$	256x256x64	residual_block	64
$X_8$	256x256x64	residual_block	64
$X_9$	256x256x64	Add[ $X_2, X_8$ ]	64
$X_{10}$	256x256x64	Conv2D	3

Bảng 2: Kiến trúc SRResNet

# Kết quả

---

Tổng tham số: 449,731



Hình 11: Ảnh high-resolution, low-resolution và output SRResNet

# Kiến trúc SRResNet(lite)

---

Block	Đầu vào	Phép toán	Số channel
$X_1$	(Input)	Interpolation	3
$X_2$	256x256x3	Conv2D	32
$X_3$	256x256x32	residual_block	32
$X_4$	256x256x32	residual_block	32
$X_5$	256x256x32	Conv2D	64
$X_6$	256x256x64	Conv2D	3

Bảng 3: Kiến trúc SRResNet(lite)

# Kết quả

---

Tổng tham số: 39,875



Hình 12: Ảnh high-resolution, low-resolution và output SRResNet(lite)

# SRResNet và SRResNet(lite)

---



Hình 13: Ảnh SRResnet và SRResNet(lite)

# Autoencoder Super Resolution

# Kiến trúc SRAutoencoder

---

Block	Đầu vào	Phép toán	Số channel
$X_1$	(Input)	Interpolation	3
$X_2$	256x256x3	Conv2D	64
$X_3$	256x256x64	Conv2D	64
$X_4$	256x256x64	MaxPooling2D	
$X_5$	128x128x64	Conv2D	128
$X_6$	128x128x128	Conv2D	128
$X_7$	128x128x128	MaxPooling2D	
<i>encoder</i>	64x64x128	Conv2D	256

Bảng 4: Kiến trúc encoder

# Kiến trúc SRAutoencoder

---

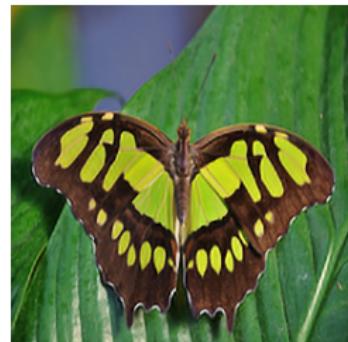
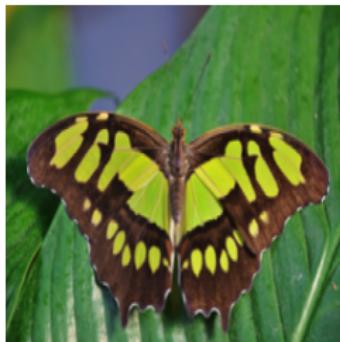
Block	Đầu vào	Phép toán	Số channel
$X_8$	64x64x256	Conv2DTranspose	256
$X_9$	128x128x256	Conv2D	128
$X_{10}$	128x128x128	Conv2D	128
$X_{11}$	128x128x128	Add[ $X_6, X_{10}$ ]	
$X_{12}$	128x128x128	Conv2DTranspose	128
$X_{13}$	256x256x128	Conv2D	64
$X_{14}$	256x256x64	Conv2D	64
$X_{15}$	256x256x64	Add[ $X_3, X_{14}$ ]	
<i>decoder</i>	256x256x64	Conv2D	3

Bảng 5: Kiến trúc decoder

# Kết quả

---

Tổng tham số: 1,848,067



Hình 14: Ảnh high-resolution, low-resolution và output Autoencoder

# Tổng kết

# Tổng kết

---

Peak signal-to-noise ratio (PSNR): dùng để tính tỉ lệ giữa giá trị năng lượng tối đa của một tín hiệu và năng lượng nhiễu ảnh hướng đến độ chính xác của thông tin

PSNR được sử dụng để đo chất lượng tín hiệu khôi phục của các thuật toán nén có mất mát dữ liệu

Công thức:

$$PSNR = 10 * \log_{10} \left( \frac{MAX^2}{MSE} \right)$$

# Tổng kết

---

Kết quả đo được từ tập validation

Tên	Parameters	Loss (MSE)	PSNR	Thời gian
SRCNN	20,483	579.3018	21.23924	0.252615s
SRResnet(lite)	39,875	411.07260	23.1243	0.266597s
SRResnet	449,731	383.9288	23.3590	0.432384s
Autoencoder	1,848,067	395.28299	23.2699	0.883944s

# Tổng kết

---



Hình 15: SRCNN, SRResNet, SRResNet(lite) và Autoencoder

# Một số ảnh khác

---



Hình 16: Ảnh high-resolution và low-resolution

# Một số ảnh khác

---



Hình 17: SRCNN, SRResNet, SRResNet(lite) và Autoencoder

# Một số ảnh khác

---



Hình 18: Ảnh high-resolution và low-resolution

# Một số ảnh khác

---



Hình 19: SRCNN, SRResNet, SRResNet(lite) và Autoencoder

# Kết thúc