

Image Super-Resolution

Tuan Phan, Tung Pham
AI For Everyone

Ngày 20 tháng 3 năm 2022

Outline

1. Bài toán image super-resolution

2. Image Interpolation

3. CNN for Super-Resolution

3.1 SRCNN

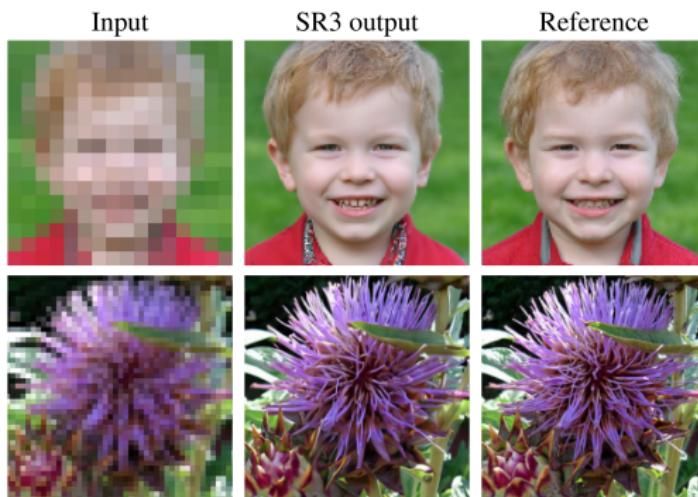
3.2 ResNet Super Resolution

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

Mục tiêu project

Upscale ảnh với độ phân giải gấp đôi và tăng chi tiết của ảnh



Hình 2: Upscale ảnh (169, 255, 3) thành (338, 510, 3)

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

Image Interpolation

Image Interpolation

```
[7] img = np.array([[10., 20.], [30., 40.]])  
     img
```

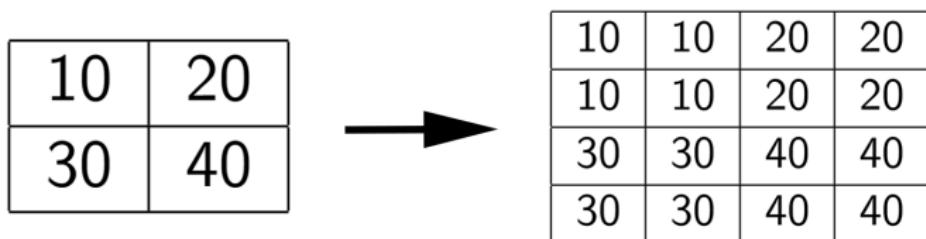
```
array([[10., 20.],  
       [30., 40.]])
```

```
[8] img = cv2.resize(img, (4,4), interpolation=cv2.INTER_LINEAR)  
     img
```

```
array([[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: Interpolation trong OpenCV

Nearest-neighbor interpolation



Hình 4: Nearest-neighbor interpolation

Bilinear Interpolation

10	20		
30	40		

→

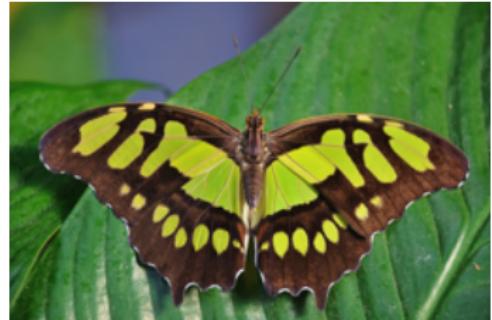
10	12.5	17.5	20
15	17.5	22.5	25
25	27.5	32.5	35
30	32.5	37.5	40

Hình 5: Bilinear Interpolation

Image Interpolation

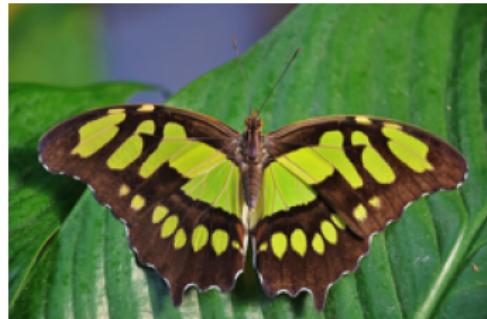
- Lanczos interpolation
- Bicubic interpolation
-

Image Interpolation



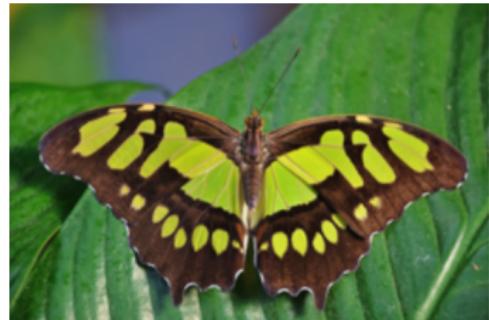
Hình 6: Ảnh high-resolution và low-resolution

Image Interpolation



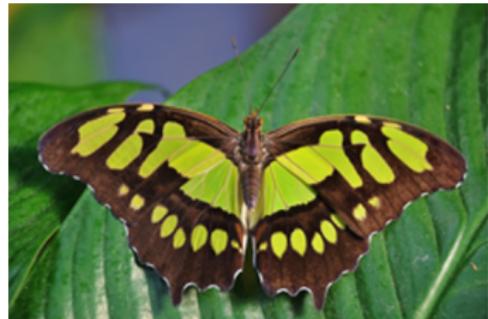
Hình 7: Ảnh high-resolution và Nearest-neighbor interpolation

Image Interpolation



Hình 8: Ảnh high-resolution và Bilinear interpolation

Image Interpolation

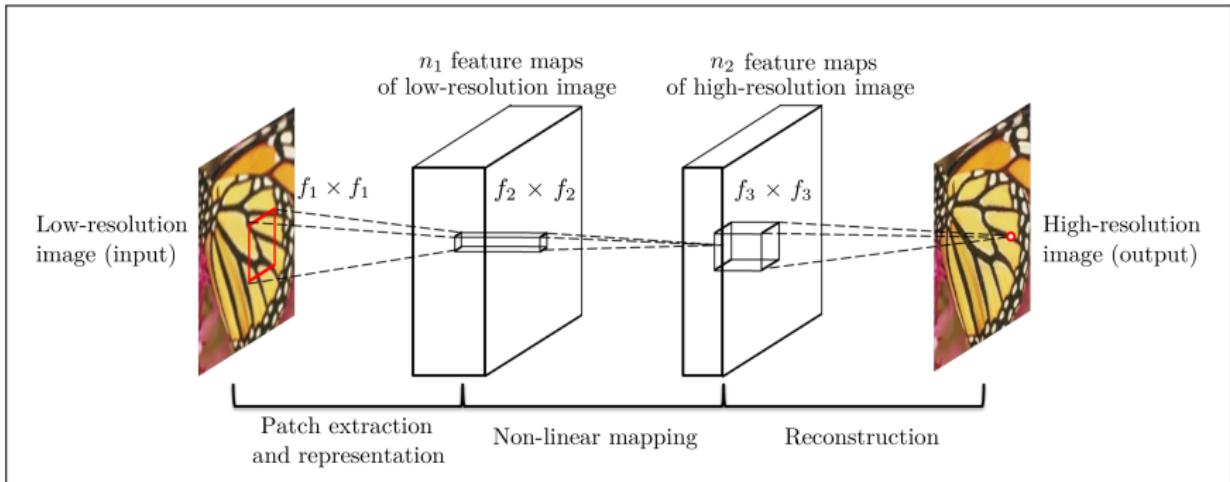


Hình 9: Ảnh high-resolution và Bicubic interpolation

CNN for Super-Resolution

SRCNN

Kiến trúc mạng



Hình 10: 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

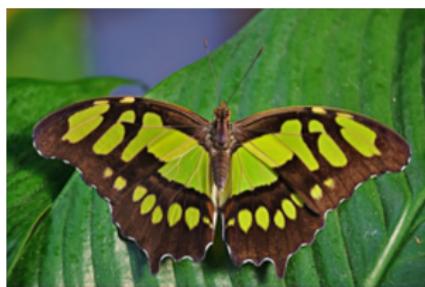
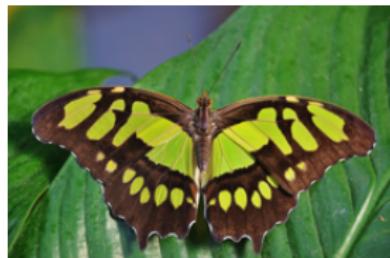
Kiến trúc SRCNN

Block	Đầu vào	Phép toán	Số channel
X_1	(Input)	Interpolation	3
X_2	AxBx3	Conv2D	64
X_3	AxBx64	Batch Normalization	
X_4	AxBx64	Conv2D	32
X_5	AxBx32	Batch Normalization	
X_6	AxBx32	Conv2D	3

Bảng 1: Kiến trúc SRCNN

Tổng tham số: 20,483

Kết quả



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

ResNet Super Resolution

Residual block

Block	Đầu vào	Phép toán	Số channel
X_0	(Input)		
X_1	AxBx64	Conv2D	64
X_2	AxBx64	Batch Normalization	
X_3	AxBx64	Conv2D	64
X_4	AxBx64	Batch Normalization	
X_5	AxBx64	Add[X_0, X_4]	

Bảng 2: Residual block

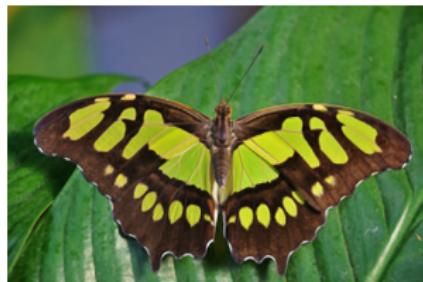
Kiến trúc SRResNet

Block	Đầu vào	Phép toán	Số channel
X_1	(Input)	Interpolation	3
X_2	AxBx3	Conv2D	64
X_3	AxBx64	residual_block	64
X_4	AxBx64	residual_block	64
X_5	AxBx64	residual_block	64
X_6	AxBx64	residual_block	64
X_7	AxBx64	residual_block	64
X_8	AxBx64	residual_block	64
X_9	AxBx64	Add[X_2, X_8]	
X_{10}	AxBx64	Conv2D	3

Bảng 3: Kiến trúc SRResNet

Kết quả

Tổng tham số: 449,731



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

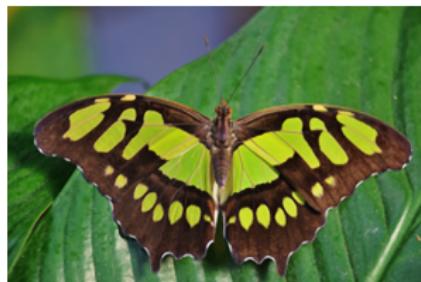
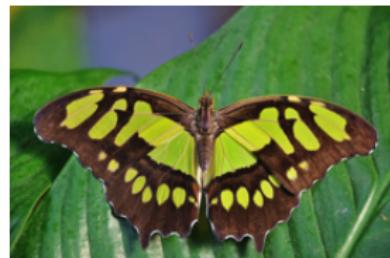
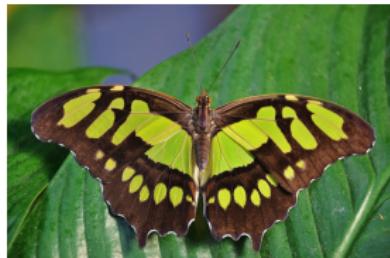
Kiến trúc SRResNet(lite)

Block	Đầu vào	Phép toán	Số channel
X_1	(Input)	Interpolation	3
X_2	AxBx3	Conv2D	32
X_3	AxBx32	Conv2D	32
X_4	AxBx32	Batch Normalization	
X_5	AxBx32	Conv2D	32
X_6	AxBx32	Batch Normalization	
X_7	AxBx32	Add[X_6, X_2]	
X_8	AxBx32	Add[X_7, X_2]	
X_9	AxBx32	Conv2D	64
X_3	AxBx64	Conv2D	3

Bảng 4: Kiến trúc SRResNet(lite)

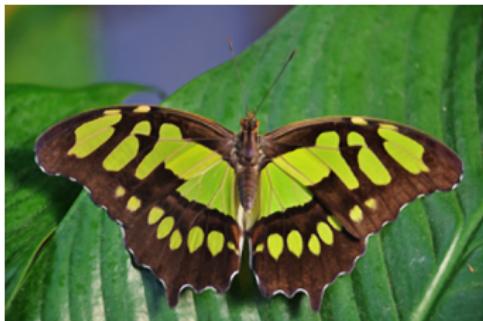
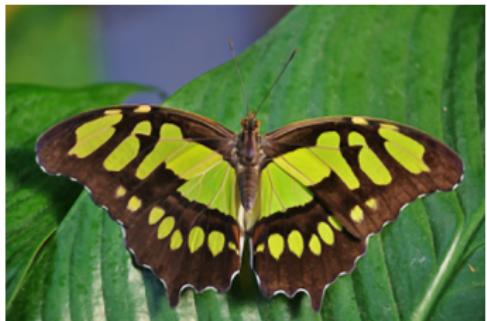
Kết quả

Tổng tham số: 39,875



Hình 13: Ảnh high-resolution, low-resolution và output SRResNet(lite)

SRResNet và SRResNet(lite)



Hình 14: 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	AxBx3	Conv2D	64
X_3	AxBx64	Conv2D	64
X_4	AxBx64	MaxPooling2D	
X_5	$(A/2) \times (B/2) \times 64$	Conv2D	128
X_6	$(A/2) \times (B/2) \times 128$	Conv2D	128
X_7	$(A/2) \times (B/2) \times 128$	MaxPooling2D	
encoder	$(A/4) \times (B/4) \times 128$	Conv2D	256

Bảng 5: Kiến trúc encoder

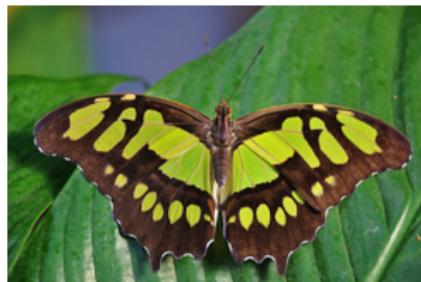
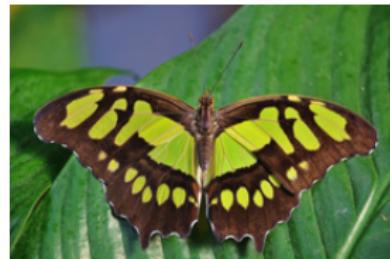
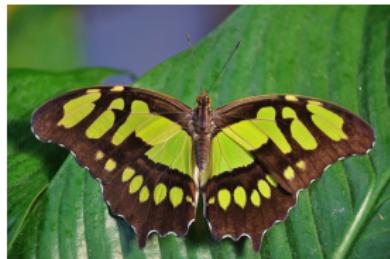
Kiến trúc SRAutoencoder

Block	Đầu vào	Phép toán	Số channel
X_8	$(A/4) \times (B/4) \times 256$	Conv2DTranspose	256
X_9	$(A/2) \times (B/2) \times 256$	Conv2D	128
X_{10}	$(A/2) \times (B/2) \times 128$	Conv2D	128
X_{11}	$(A/2) \times (B/2) \times 128$	Add[X_6, X_{10}]	
X_{12}	$(A/2) \times (B/2) \times 128$	Conv2DTranspose	128
X_{13}	AxBx128	Conv2D	64
X_{14}	AxBx64	Conv2D	64
X_{15}	AxBx64	Add[X_3, X_{14}]	
decoder	AxBx64	Conv2D	3

Bảng 6: Kiến trúc decoder

Kết quả

Tổng tham số: 1,848,067



Hình 15: 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:

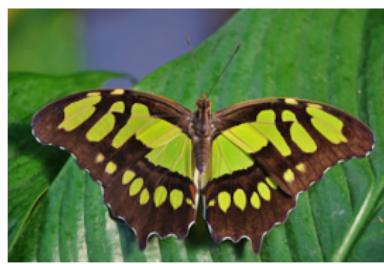
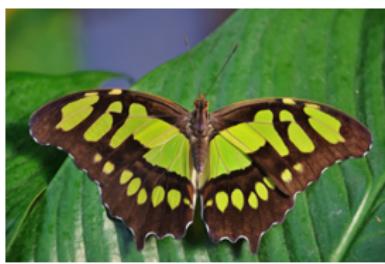
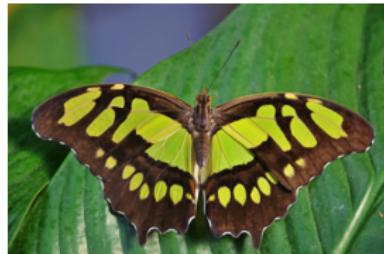
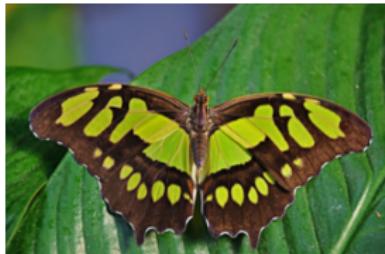
$$PSNR = 10 * \log_{10} \left(\frac{MAX^2}{MSE} \right)$$

Tổng kết

Kết quả đo được từ tập validation

Tên	Parameters	Loss (MSE)	PSNR	FPS
SRCNN	20,483	506.34	21.94	8.81
SRResnet(lite)	39,875	410.28	23.11	7.70
SRResnet	449,731	392.09	23.37	2.58
Autoencoder	1,848,067	401.55	23.18	3.94

Tổng kết



Hình 16: SRCNN, SRResNet, SRResNet(lite) và SRAutoencoder

Một số ảnh khác



Hình 17: Ảnh high-resolution và low-resolution

Một số ảnh khác



Hình 18: SRCNN, SRResNet, SRResNet(lite) và SRAutoencoder

Một số ảnh khác



Hình 19: Ảnh high-resolution và low-resolution

Một số ảnh khác



Hình 20: SRCNN, SRResNet, SRResNet(lite) và Autoencoder

Kết thúc