

# Super-Resolución de Imágenes: Redes Convolucionales Residuales con Pérdida de Similitud Estructural

Marcos Piotto<sup>1</sup> y Segundo Santos Torrado<sup>1</sup>

**Resumen**—La super-resolución de imágenes individuales (SR) se trata de un problema clásico en visión por computadora por el que se busca recuperar una imagen en alta resolución a partir de una imagen individual en baja resolución. En este trabajo, se analiza la arquitectura de red neuronal convolucional FSRCNN y se propone la adición de conexiones residuales, el reemplazo de la capa de mapeo no lineal por bloques residuales y la utilización de funciones de pérdida más propias para similaridad de imágenes que el *Mean Squared Error* (MSE), esto es, basadas en las métricas *Peak Signal-to-Noise Ratio* (PSNR) y *Structural Similarity Index Measure* (SSIM). Se obtuvo que la arquitectura de red propuesta RSRCNN (Residual Super-Resolution Convolutional Neural Network), así como las funciones de pérdida introducidas NPSNR y CSSIM, logran mejorar la *performance* en términos de PSNR y SSIM respecto al modelo original FSRCNN optimizado con costo MSE.

**Index Terms**—Super-Resolución de Imágenes, Redes Neuronales Convolucionales Residuales, Índice de Similitud Estructural

## 1. INTRODUCCIÓN

La super-resolución de imágenes individuales (SR), cuyo objetivo es recuperar una imagen en alta resolución a partir de una imagen individual en baja resolución, es un problema clásico en visión por computadora. La SR es un problema *ill-posed*, esto es, un problema inverso indeterminado que carece de solución única.

Mientras que técnicas tradicionales de SR como la interpolación bicúbica producen imágenes borrosas y artefactos de alta frecuencia, los métodos basados en aprendizaje profundo han demostrado resultados superiores. En particular, las redes neuronales convolucionales (CNN) se convirtieron en el enfoque dominante gracias a su capacidad para aprender representaciones jerárquicas de características visuales.

La primera arquitectura de red neuronal convolucional para SR fue SRCNN [1], que demostró ser efectiva pero computacionalmente costosa. Posteriormente, FSRCNN [2] propuso una arquitectura más eficiente y rápida, que consiste en una red de dos etapas: una etapa de extracción de características y una etapa de mapeo de píxeles. FSRCNN logra resultados comparables a SRCNN con una fracción del costo computacional.

En este trabajo, se propone una mejora a la arquitectura FSRCNN mediante la adición de técnicas modernas de aprendizaje profundo. En particular, se propone la adición de conexiones residuales [3], el reemplazo de la capa de mapeo no lineal por bloques residuales [3] y la optimización del modelo con funciones de pérdida más adecuadas para similaridad de imágenes como el *Peak Signal-to-Noise Ratio* (PSNR) y el *Structural Similarity Index Measure* (SSIM) [4], en lugar del *Mean Squared Error* (MSE) utilizado en FSRCNN.

El modelo propuesto, denominado RSRCNN (Residual Super-Resolution Convolutional Neural Network), fue entrenado y evaluado en varios conjuntos de datos estándar de SR. Para el entrenamiento, se utilizaron los conjuntos de datos General100 [2] y T91 [5]; para validación, se empleó BSD100 [6]; y para la evaluación final, se utilizaron Set5 y Set14 [7].

Los resultados experimentales muestran que RSRCNN, junto con las funciones de pérdida NPSNR (Negative PSNR) 3.2.2 y CSSIM (Complement SSIM) 3.2.3, logra mejorar la *performance* en términos de PSNR 3.1.2 y SSIM 3.1.3 respecto al modelo original FSRCNN optimizado con costo MSE 3.1.1.

En las subsiguientes secciones se describe la arquitectura de red RSRCNN propuesta y sus diferencias con FSRCNN, las métricas de evaluación de similaridad de imágenes utilizadas, así como las funciones de pérdida confeccionadas para la optimización de los modelos. Se detallan los conjuntos de datos utilizados y el preprocesamiento realizado sobre las muestras de entrenamiento, además de los hiperparámetros y la metodología de entrenamiento empleada en cuanto a la evaluación de los modelos, su implementación y su optimización en *hardware*. Por último, se presentan los resultados experimentales obtenidos y se discuten las conclusiones y los posibles trabajos futuros.

## 2. CNNS PARA SUPER-RESOLUCIÓN

### 2.1. FSRCNN (Fast Super-Resolution CNN)

Tal como se observa en la figura 1, FSRCNN [2] puede ser descompuesta en cinco etapas:

1. **Feature Extraction:** Consiste en una capa convolucional de  $5 \times 5$  con  $d$  filtros y un *stride* de 1 que aprende características de la imagen de baja resolución,

• <sup>1</sup> Departamento de Ingeniería, Universidad de San Andrés, Buenos Aires, Argentina.  
E-mail: {mpiotto, ssantostorrado}@udesa.edu.ar

2. **Shrinking:** Es una capa convolucional de  $s$  filtros de  $1 \times 1$  que reduce la cantidad de canales de la imagen,
3. **Non-Linear Mapping:** Son  $m$  capas convolucionales de  $s$  filtros de  $3 \times 3$  que aprenden una función no lineal de mapeo de píxeles,
4. **Expanding:** Es una capa convolucional de  $d$  filtros de  $1 \times 1$  que expande la cantidad de canales de la imagen,
5. **Deconvolution:** Es una capa de convolución transpuesta que aumenta la resolución de la imagen a la escala deseada. Luego, la cantidad de parámetros de modelos con distintas escalas difiere únicamente por esta capa,

donde todas las capas convolucionales son seguidas por una función de activación PReLU [8] de fórmula:

$$\text{PReLU}(x) = \max(0, x) + \alpha \min(0, x), \quad (1)$$

donde  $\alpha$  es un parámetro aprendible que permite la propagación de gradientes negativos, a diferencia de ReLU que los anula.

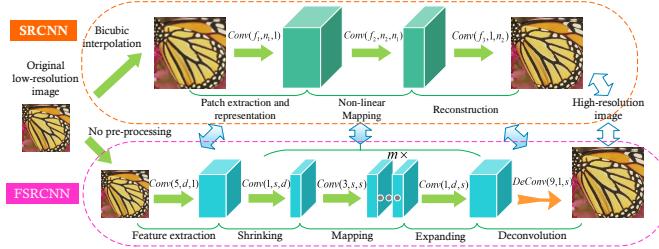


Figura 1. Arquitectura de FSRCNN (Fast Super-Resolution Convolutional Neural Network) [2] en relación a SRCNN (Super-Resolution Convolutional Neural Network) [1].

## 2.2. RSRCNN (Residual Super-Resolution CNN)

Para mejorar la arquitectura de FSRCNN, se propone el reemplazo de la capa de mapeo no lineal por  $m$  bloques residuales [3], como se observa en la figura 2. Cada bloque residual se compone de dos capas convolucionales de  $3 \times 3$  con activación PReLU y Batch Normalization [9]. Adicionalmente, se agregan conexiones residuales entre los *outputs* de las capas *Feature Extraction* y *Expanding*, así como entre las salidas de *Shrinking* y el *Residual Block*, tal como se muestra en la figura 3.

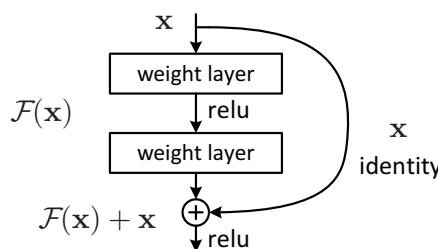


Figura 2. Bloque residual [3].

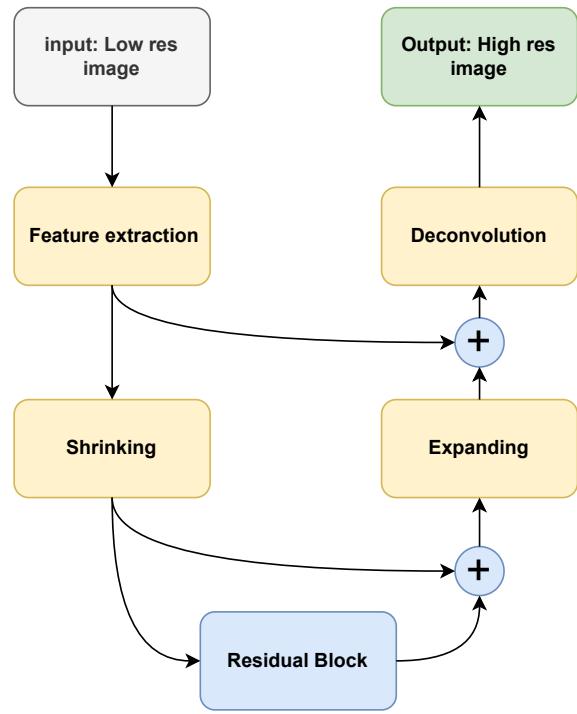


Figura 3. Arquitectura de RSRCNN (Residual Super-Resolution Convolutional Neural Network).

## 3. MÉTRICAS DE SIMILARIDAD DE IMÁGENES

En esta sección, se describen las métricas de evaluación de similaridad de imágenes utilizadas para comparar la calidad de las imágenes originales y reconstruidas. Adicionalmente, se presentan las funciones de pérdida diseñadas en función de estas métricas.

### 3.1. Métricas de Evaluación

En primer lugar, se describen las posibles métricas de evaluación de similaridad de imágenes empleadas para comparar la *performance* de los modelos de SR.

#### 3.1.1. Mean Squared Error (MSE)

Calcula el promedio del error cuadrático entre los valores de los píxeles de la imagen original y los de la imagen reconstruida. Un valor más bajo de MSE indica una mayor similitud entre ambas imágenes.

$$\text{MSE}(x, \hat{x}) = \frac{1}{N} \sum_{i=1}^N (x_i - \hat{x}_i)^2 \quad (2)$$

- $x_i$ : Valor del píxel en la imagen original.
- $\hat{x}_i$ : Valor del píxel en la imagen reconstruida.
- $N$ : Número total de píxeles en la imagen.

### 3.1.2. Peak Signal-to-Noise Ratio (PSNR)

Es una métrica basada en la MSE que mide la calidad de la imagen reconstruida en términos de la relación entre la potencia máxima de la señal (imagen original) y el ruido (diferencias con la reconstrucción). Se expresa en decibelios (dB), y un valor más alto indica mejor calidad.

$$\text{PSNR}(x, \hat{x}) = 10 \log_{10} \left( \frac{\text{MAX}_I^2}{\text{MSE}(x, \hat{x})} \right) \quad (3)$$

- $\text{MAX}_I$ : Valor máximo posible para un píxel en la imagen (por ejemplo, 255 para imágenes de 8 bits).

### 3.1.3. Structural Similarity Index Measure (SSIM)

Evaluá la similitud entre la imagen original y la reconstruida en términos de luminancia, contraste y estructura. Es más perceptualmente relevante que MSE o PSNR, ya que se centra en cómo los humanos perciben las diferencias visuales. Varía entre 0 y 1, siendo 1 la máxima similitud.

$$\text{SSIM}(x, \hat{x}) = \frac{(2\mu_x\mu_{\hat{x}} + C_1)(2\sigma_{x\hat{x}} + C_2)}{(\mu_x^2 + \mu_{\hat{x}}^2 + C_1)(\sigma_x^2 + \sigma_{\hat{x}}^2 + C_2)} \quad (4)$$

- $\mu_x, \mu_{\hat{x}}$ : Promedio de los valores de píxeles de las imágenes  $x$  y  $\hat{x}$ , respectivamente.
- $\sigma_x^2, \sigma_{\hat{x}}^2$ : Varianzas de las imágenes  $x$  y  $\hat{x}$ , respectivamente.
- $\sigma_{x\hat{x}}$ : Covarianza entre las imágenes  $x$  y  $\hat{x}$ .
- $C_1, C_2$ : Constantes pequeñas para evitar divisiones por cero. Se calculan como  $C_1 = (K_1 L)^2$  y  $C_2 = (K_2 L)^2$ , donde  $L$  es el rango dinámico de la imagen y  $K_1, K_2$  son constantes empíricas.

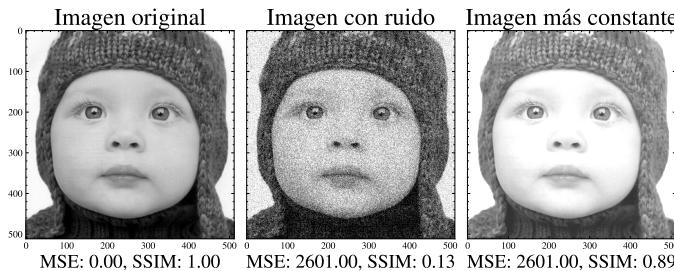


Figura 4. Se analiza que a MSE constante la percepción de calidad visual varía ampliamente.

## 3.2. Funciones de Pérdida

En segundo lugar, se presentan las funciones de costo que son utilizadas en la optimización de los modelos de SR desarrollados.

### 3.2.1. MSE

La función de pérdida *Mean Squared Error* (MSE) es la más comúnmente utilizada en SR y se define al igual que la métrica 3.1.1.

### 3.2.2. NPSNR

Dado que a mayor similaridad entre las imágenes, mayor es el valor de PSNR 3.1.2, se propone la función de pérdida *Negative Peak Signal-to-Noise Ratio* (NPSNR) como el negativo del PSNR:

$$\text{NPSNR}(x, \hat{x}) = -\text{PSNR}(x, \hat{x}). \quad (5)$$

### 3.2.3. CSSIM

De forma análoga, se propone la función de pérdida *Complement Structural Similarity Index Measure* (CSSIM) como el complemento a uno del SSIM 3.1.3:

$$\text{CSSIM}(x, \hat{x}) = 1 - \text{SSIM}(x, \hat{x}). \quad (6)$$

## 4. DESARROLLO EXPERIMENTAL

En esta sección, se detallan los conjuntos de datos utilizados para cada etapa del desarrollo de los modelos de SR, así como el preprocesamiento realizado sobre las muestras de entrenamiento. Se describen los hiperparámetros y la metodología de entrenamiento empleada, incluyendo la evaluación de los modelos, su implementación y su optimización en *hardware*.

### 4.1. Conjuntos de Datos

En este estudio se utilizaron diferentes conjuntos de datos tanto para el entrenamiento, validación y evaluación final del modelo. Los conjuntos de datos utilizados son los siguientes:

- **General100** y **T91** [5]: Empleados para el entrenamiento del modelo. Estos conjuntos de datos contienen imágenes de alta calidad, variadas en términos de textura, forma y contenido, ideales para aprender características representativas del problema de super-resolución.
- **BSD100** [6]: Utilizado para validación. Este conjunto proviene del *Berkeley Segmentation Dataset* y se compone de imágenes con estructuras y texturas complejas, lo que permite evaluar la capacidad del modelo para generalizar.
- **Set5** y **Set14** [7]: Usados para calcular las métricas finales de rendimiento, como el PSNR (*Peak Signal-to-Noise Ratio*) y SSIM (*Structural Similarity Index Measure*). Ambos son conjuntos estándar en tareas de super-resolución, consistiendo en imágenes clásicas ampliamente utilizadas para comparar modelos en términos de calidad visual.

### 4.2. Preprocesamiento de Datos

Para mejorar la robustez del modelo y maximizar la cantidad de datos disponibles para el entrenamiento, se realizó un proceso de preprocesamiento y aumento de datos (*data augmentation*), descrito a continuación.

#### Conversión de RGB a YCbCr

Las imágenes en color están representadas comúnmente en el espacio de color RGB (Rojo, Verde, Azul). Sin embargo, en tareas de super-resolución, es habitual transformar las imágenes al espacio de color YCbCr, ya que este separa la información de luminancia ( $Y$ , que representa el brillo de la imagen) de las componentes de crominancia ( $Cb$  y  $Cr$ , que representan el color).

La mayoría de los modelos de super-resolución se entran únicamente en el canal  $Y$  (luminancia), ya que este contiene la mayor parte de la información estructural de la imagen. Posteriormente, las componentes  $Cb$  y  $Cr$  se

interpolan bicúbicamente y se combinan con el canal Y generado por el modelo para reconstruir la imagen en color. Este enfoque simplifica el problema y, al mismo tiempo, mantiene la calidad visual.

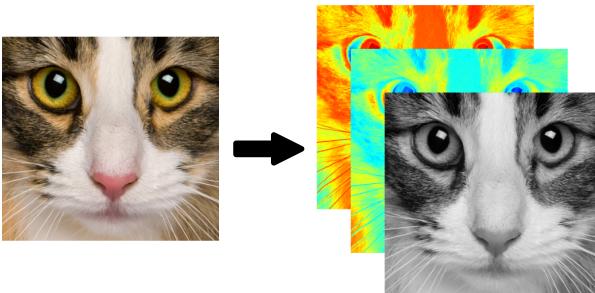


Figura 5. Transformación de espacio de color RGB a YCbCr (luma y croma). Se predice sobre el canal Y.

### Rotaciones

Cada imagen fue rotada por ángulos de  $90^\circ$ ,  $180^\circ$  y  $270^\circ$  para aumentar la diversidad de patrones geométricos disponibles para el modelo. Las rotaciones aseguran que el modelo sea capaz de aprender características invariantes a la orientación de los objetos.

### Escalado

Para simular diferentes niveles de resolución, las imágenes fueron escaladas por factores de **0.9, 0.8, 0.7 y 0.6**. Este procedimiento aumenta la robustez del modelo frente a imágenes de diferentes tamaños y resoluciones originales.

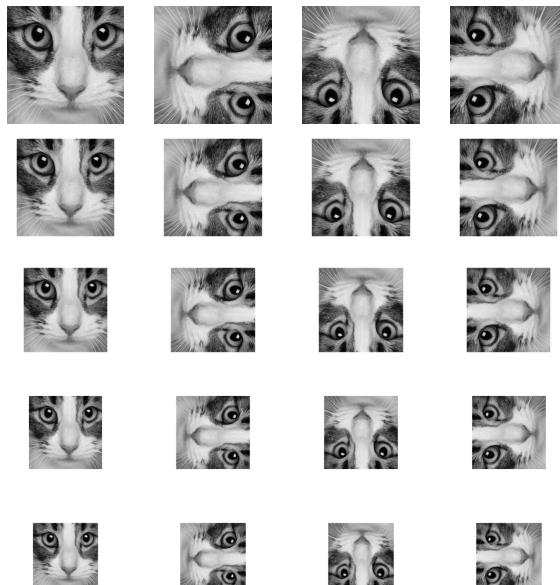


Figura 6. Rotación de  $90^\circ$ ,  $180^\circ$  y  $270^\circ$ , y reescalado según factores de 0,9, 0,8, 0,7 y 0,6.

### División en Parches

Finalmente, cada imagen fue dividida en parches de **64x64 píxeles**. Este paso es esencial en problemas de superresolución por varias razones:

- Dividir las imágenes en parches más pequeños permite un entrenamiento más eficiente y asegura que el modelo se enfoque en aprender características locales.
- El tamaño de 64x64 es un estándar que equilibra la capacidad de aprendizaje del modelo y la eficiencia computacional.
- Entrenar sobre imágenes completas o parches más grandes es computacionalmente costoso.

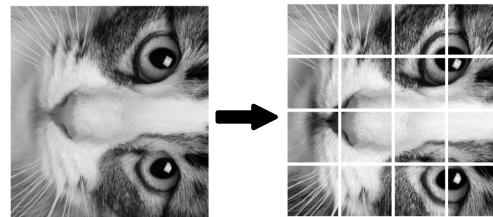


Figura 7. Fragmentación en parches uniformes para *batches*.

Este proceso de preprocesamiento y aumento de datos no solo incrementa la cantidad de datos disponibles, sino que también mejora la capacidad del modelo para generalizar, haciéndolo más robusto frente a variaciones en orientación, tamaño y resolución de las imágenes.

### 4.3. Métodología de Entrenamiento

En esta sección, se detalla el procedimiento experimental llevado a cabo en el proyecto, así como las decisiones de implementación tomadas para el desarrollo de los modelos evaluados en este informe.

Dadas las dos arquitecturas FSRCNN 2.1 y RSRCNN 2.2, se entrenaron modelos de dos tamaños distintos:

	Cantidad de Parámetros	
Factor de Escalado	$2\times$	$4\times$
FSRCNN (32, 5, 1)	2006	3542
RSRCNN (32, 5, 1)	2256	3792
FSRCNN (56, 12, 4)	9169	11857
RSRCNN (56, 12, 4)	14593	17281

Cuadro 1

Comparación de modelos implementados según su cantidad de parámetros entrenables.

De este modo, se entrenaron 2 arquitecturas (FSRCNN 2.1 y RSRCNN 2.2) para 2 tamaños de modelos ( $d = 56, s = 12, m = 4$  y  $d = 32, s = 5, m = 1$ ), 2 factores de escalado ( $2\times$  y  $4\times$ ) y con 3 funciones de pérdida distintas (MSE 3.2.1, NPSNR 3.2.2 y CSSIM 3.2.3). Se utilizó un *learning rate* de  $1 \times 10^{-3}$ , un tamaño de *batch* de 64 y se entrenó durante 1000 épocas a cada modelo. En cuanto al número de parches utilizados para el entrenamiento, se emplearon  $n = 1000$  para el factor de escalado  $2\times$  y  $n = 10000$  para el factor de escalado  $4\times$  para reducir el *overfitting* ocasionado por la mayor cantidad de parámetros en la capa deconvolucional del modelo con mayor *upsampling factor*. En particular, para el proceso de optimización se utilizó una GPU NVIDIA GeForce GTX 1650 Mobile con 4 GB de VRAM y requirió no más de 30 minutos de entrenamiento por modelo.

## 5. RESULTADOS Y ANÁLISIS

En esta sección se presentan los resultados obtenidos y se discuten en relación con los objetivos planteados, destacando hallazgos relevantes.

En primer lugar se estudiaron las curvas de validación tanto para la métrica PSNR como SSIM, optimizados con la función de perdida que maximiza a cada uno respectivamente, esto es, NPSNR y CSSIM.

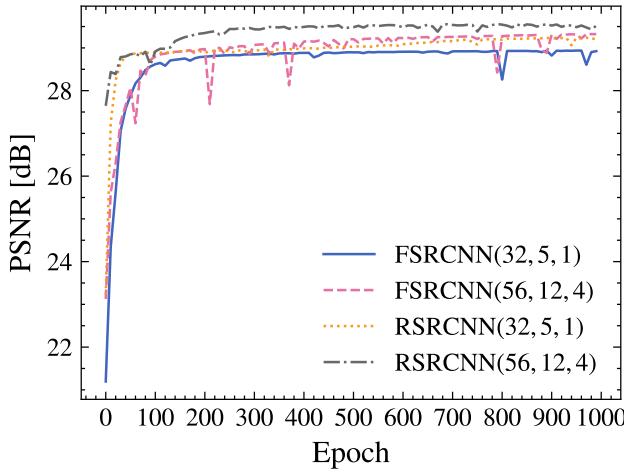


Figura 8. Curvas de PSNR de validación para factor de escalado  $2\times$  durante 1000 epochs empleando NPSNR como función de pérdida. Se utilizó  $1 \times 10^{-3}$  de learning rate y batch size de 64.

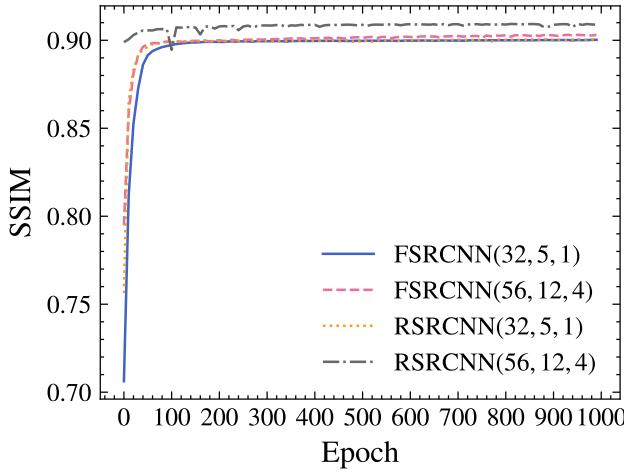


Figura 9. Curvas de SSIM de validación para factor de escalado  $2\times$  durante 1000 epochs empleando CSSIM como función de pérdida. Se utilizó  $1 \times 10^{-3}$  de learning rate y batch size de 64.

Tanto en la figura 8 como en 9 se observa que los modelos residuales obtienen mejores resultados que FSRCNN en sus versiones de más parámetros, así como para los modelos más pequeños. Por su parte, el modelo de RSRCNN de menor escala se aproxima al rendimiento del FSRCNN más grande.

Por otro lado, se presentan los resultados cuantitativos en dos conjuntos de evaluación: Set5 y Set14. En particular, en las tablas 2 y 3 se exponen los resultados de PSNR y SSIM para interpolación bicúbica y para los modelos FSRCNN y RSRCNN con diferentes configuraciones de función de

pérdida y factor de escala. Se analiza que en casi todos los casos la utilización de la función de costo MSE no es la mejor opción, tanto si se busca maximizar PSNR (es máximo con NPSNR) como si la métrica de mayor interés es SSIM (es óptimo con CSSIM).

De este modo, se obtuvo que el mejor modelo desarrollado tanto para las escalas  $2\times$  y  $4\times$  es *RSRCNN(56, 12, 4)* optimizado con NPSNR si se busca maximizar PSNR (puntúa 35.06 dB. en Set5  $2\times$  y 30.67 dB. en Set14  $2\times$ ) u optimizado con CSSIM si se busca maximizar SSIM (puntúa 0.9529 en Set5  $2\times$  y 0.9057 en Set14  $2\times$ ).

Por ultimo se presentan resultados visuales para una imagen del set de test, con factor de escalado  $4\times$ . Donde el canal Y (luma) es obtenido por el modelo RSRCNN, mientras que los canales Cb y Cr (croma) son ampliados empleando interpolación bicúbica. Se observa, que el modelo predictivo tiene una mejora significativa frente a la interpolación bicúbica.

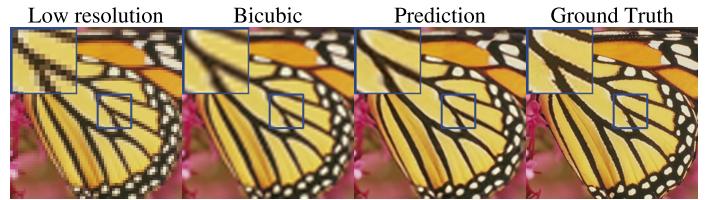


Figura 10. Resultados visuales de super-resolución, utilizando el modelo RSRCNN(56,12,4) para escala  $4\times$ .

## 6. CONCLUSIÓN

A modo de síntesis, el objetivo principal de este proyecto fue mejorar la resolución de imágenes de baja calidad a través de la implementación y entrenamiento de redes neuronales convolucionales (CNNs). Para ello, se realizó un proceso de aumento de datos, que consistió en aplicar transformaciones como rotaciones, escalado y partición de las imágenes en pequeños parches de  $64 \times 64$  píxeles, lo cual permitió generar una mayor variedad de datos de entrenamiento.

Para abordar el problema de la super-resolución, se desarrollaron dos arquitecturas de redes neuronales convolucionales: FSRCNN y RSRCNN. RSRCNN introduce un bloque residual que mejora la capacidad de aprendizaje del modelo. Estas redes fueron entrenadas utilizando tres funciones de pérdida diferentes: MSE 3.2.1, NPSNR 3.2.2 y CSSIM 3.2.3, lo que permitió explorar las ventajas y desventajas de cada una para la mejora de las distintas métricas de similaridad de imágenes (PSNR 3.1.2 y SSIM 3.1.3).

Además, se realizaron experimentos con factores de escala de  $2\times$  y  $4\times$ , para analizar cómo el modelo se comporta frente a diferentes niveles de incremento en la resolución de la imagen. Los modelos fueron evaluados utilizando dos conjuntos de datos de imágenes estándar en la literatura, Set5 y Set14 [7].

Los resultados obtenidos en los experimentos indican que la introducción del bloque residual y de las conexiones residuales en la arquitectura RSRCNN mejora los resultados en comparación con el modelo FSRCNN para todos los casos evaluados y en ambos tamaños de modelo analizados.

Modelo	Función de Perdida	2x PSNR	2x SSIM	4x PSNR	4x SSIM
Bicúbico	-	32.62 (db)	0.9322	28.99 (db)	0.8757
FSRCNN (32, 5, 1)	MSE	33.60 (db)	0.9419	28.39 (db)	0.8365
	NPSNR	33.58 (db)	0.9424	26.66 (db)	0.7801
	CSSIM	33.37 (db)	0.9431	27.12 (db)	0.8185
RSRCNN (32, 5, 1)	MSE	33.95 (db)	0.9446	26.95 (db)	0.7947
	NPSNR	34.08 (db)	0.9456	28.21 (db)	0.8314
	CSSIM	32.74 (db)	0.9432	27.22 (db)	0.8306
FSRCNN (56, 12, 4)	MSE	34.26 (db)	0.9474	26.85 (db)	0.7786
	NPSNR	34.35 (db)	0.9476	26.73 (db)	0.7748
	CSSIM	33.90 (db)	0.9476	27.91 (db)	0.8441
RSRCNN (56, 12, 4)	MSE	34.64 (db)	0.9488	28.44 (db)	0.8399
	NPSNR	35.06 (db)	0.9519	28.32 (db)	0.8375
	CSSIM	34.78 (db)	0.9529	28.14 (db)	0.8494

Cuadro 2

Métricas sobre el *dataset* de evaluación Set 5. Evaluación de los modelos Bicúbico, FSRCNN (en dos tamaños) y RSRCNN con diferentes configuraciones. Las métricas PSNR y SSIM están evaluadas para factores de escalado de  $2\times$  y  $4\times$ .

Modelo	Función de Perdida	2x PSNR	2x SSIM	4x PSNR	4x SSIM
Bicúbico	-	27.30 (db)	0.8096	24.67 (db)	0.7048
FSRCNN (32, 5, 1)	MSE	29.75 (db)	0.8922	25.44 (db)	0.7337
	NPSNR	29.74 (db)	0.8924	24.29 (db)	0.6873
	CSSIM	29.66 (db)	0.8956	24.62 (db)	0.7255
RSRCNN (32, 5, 1)	MSE	30.06 (db)	0.8955	24.55 (db)	0.7021
	NPSNR	30.15 (db)	0.8962	25.33 (db)	0.7309
	CSSIM	29.48 (db)	0.8967	24.80 (db)	0.7358
FSRCNN (56, 12, 4)	MSE	30.22 (db)	0.8959	24.38 (db)	0.6838
	NPSNR	30.27 (db)	0.8961	24.30 (db)	0.6818
	CSSIM	30.08 (db)	0.8997	25.24 (db)	0.7437
RSRCNN (56, 12, 4)	MSE	30.40 (db)	0.8985	25.59 (db)	0.7365
	NPSNR	30.67 (db)	0.9018	25.45 (db)	0.7351
	CSSIM	30.61 (db)	0.9057	25.43 (db)	0.7483

Cuadro 3

Métricas sobre el *dataset* de evaluación Set 14. Evaluación de los modelos Bicúbico, FSRCNN (en dos tamaños) y RSRCNN con diferentes configuraciones. Las métricas PSNR y SSIM están evaluadas para factores de escalado de  $2\times$  y  $4\times$ .

Esto sugiere que los bloques residuales ayudan a preservar características importantes de las imágenes y facilitan el aprendizaje de detalles más finos. Por otro lado, los experimentos con diferentes funciones de pérdida mostraron que, tal como se muestra en 2 y 3, para optimizar la métrica PSNR, la mejor opción es utilizar la función de pérdida NPSNR, mientras que si el objetivo es maximizar SSIM, se debe emplear CSSIM.

## 7. TRABAJO A FUTURO

En el trabajo futuro, se propone explorar soluciones del *state-of-the-art* en el campo de la super-resolución de imágenes (ISR), como las redes transformadoras (ViT) o las redes generativas adversariales (GANs), que han demostrado ser prometedoras en diversas aplicaciones de visión por computadora. Estas arquitecturas podrían ofrecer nuevas formas de abordar el problema de la super-resolución y mejorar los resultados obtenidos en este proyecto.

Además, si bien se obtuvieron resultados satisfactorios con la metodología experimental utilizada, se podría realizar una búsqueda más exhaustiva de los hiperparámetros como las dimensiones de las redes, el *learning rate* o el tamaño de los parches, entre otros. Lo que podría contribuir a un mejor rendimiento de los modelos. Este trabajo de optimización permitiría evaluar el impacto de diferentes configuraciones de parámetros, proporcionando una comprensión más profunda de cómo cada uno influye en la calidad final de la super-resolución.

## REFERENCIAS

- [1] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 2, pp. 295–307, 2016.
- [2] C. Dong, C. C. Loy, and X. Tang, "Accelerating the super-resolution convolutional neural network," 2016. [Online]. Available: <https://arxiv.org/abs/1608.00367>
- [3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.
- [4] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, 09 2004.
- [5] J. Yang, J. Wright, T. S. Huang, and Y. Ma, "Image super-resolution via sparse representation," *IEEE Transactions on Image Processing*, vol. 19, no. 11, pp. 2861–2873, 2010.
- [6] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, vol. 2, 2001, pp. 416–423 vol.2.
- [7] M. Bevilacqua, A. Roumy, C. Guillemot, and M.-L. Alberi-Morel, "Low-complexity single image super-resolution based on nonnegative neighbor embedding," 09 2012.
- [8] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," *IEEE International Conference on Computer Vision (ICCV 2015)*, vol. 1502, 02 2015.
- [9] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 02 2015.