



ugr | Universidad
de Granada

TRABAJO FIN DE GRADO
INGENIERÍA EN INGENIERÍA INFORMÁTICA

Restauración de imágenes usando aprendizaje profundo

Autor
David Gil Bautista

Directores
Rafael Molina Soriano (director)
Santiago López Tapia (tutor)



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS INFORMÁTICA Y DE
TELECOMUNICACIÓN

—
Granada, 8 de septiembre de 2020

Restauración de imágenes usando aprendizaje profundo

David Gil Bautista

Palabras clave: restauración, deconvolución ciega, aprendizaje profundo, redes convolucionales

Resumen

La deconvolución ciega de imágenes se ha abordado a lo largo de los años utilizando métodos analíticos que, utilizando información previa sobre la imagen, han conseguido ser la mejor propuesta a todo tipo de imágenes por su fiabilidad y adaptabilidad a distintos desenfoques. En este trabajo combinaremos las técnicas analíticas y su adaptabilidad junto a los nuevos métodos de aprendizaje profundo que ofrecen un buen resultado con un coste computacional mucho menor que las técnicas analíticas.

Blind image deblurring using deep learning

David Gil Bautista

Keywords:restoration, blind image deconvolution, deblurring, deep learning, convolutional neural networks

Abstract

Blind image deblurring has been addressed over the years using analytical methods that, using previous information on the image, have been proved to be the best proposal for all types of images due to their reliability and adaptability to different blurs. In this work we combine analytical techniques and their adaptability together with new deep learning methods that provide a good result with a much lower computational cost than analytical ones.

Yo, **David Gil Bautista**, alumno de la titulación INGENIERÍA INFORMÁTICA de la Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada, con DNI 45925324M, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: David Gil Bautista

Granada a 8 de mes septiembre de 2020 .

D. **Rafael Molina Soriano (director)**, Profesor del Área de Visual Information Processing del Departamento DECSAI de la Universidad de Granada.

D. **Santiago López Tapia (tutor)**, Profesor del Área de Visual Information Processing del Departamento DECSAI de la Universidad de Granada.

Informan:

Que el presente trabajo, titulado ***Restauración de imágenes usando aprendizaje profundo***, ha sido realizado bajo su supervisión por **David Gil Bautista**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a 8 de mes septiembre de 2020 .

Los directores:

Rafael Molina Soriano (director)

Santiago López Tapia (tutor)

Agradecimientos

A mi familia por el constante apoyo incondicional, a todos los amigos que han compartido su tiempo conmigo durante este indeleble trabajo y a Santiago, que, a parte de la ayuda, ha hecho que se esfume un arrepentimiento que me ha perseguido estos últimos años.

Índice general

1. Introducción al problema de la deconvolución ciega de imágenes	1
1.1. Definición del problema	1
1.2. Estado del arte	4
1.3. Objetivos	7
1.4. Metodología	7
2. Métodos analíticos y de aprendizaje profundo para la deconvolución ciega de imágenes	9
2.1. Blind Image Deblurring via Deep Discriminative Priors	9
2.1.1. Clasificador binario	10
2.1.2. Función de pérdida	10
2.1.3. Modelo de deconvolución ciega	12
2.2. Fast Bayesian blind deconvolution with Huber Super Gaussian priors	13
2.2.1. Estimación del emborronamiento	13
2.2.2. Estimación de la imagen y de los parámetros	14
2.2.3. Estimación del emborronamiento	15
2.2.4. Estimación multiescala del emborronamiento	17
2.2.5. Reconstrucción de la imagen final	17
2.3. Fast High-Quality non-Blind Deconvolution Using Sparse Adaptive Priors	18
2.4. Non-uniform Deblurring for Shaken Images	23
2.4.1. Modelo de restauración	23
2.4.2. Estimación del emborronamiento	24
2.4.3. Reconstrucción de la imagen	25
2.5. Scale-recurrent Network for Deep Image Deblurring	26
2.5.1. Red recurrente	27
2.5.2. Codificador/decodificador con ResBlocks	27
2.6. Deep Stacked Hierarchical Multi-Patch Network for Image Deblurring	30
2.6.1. Arquitectura codificadora/decodificadora	30
2.6.2. Arquitectura de la red	31

2.6.3. Red apilada de parches múltiples	32
2.7. Efficient And Interpretable Deep Blind Image Deblurring Via Algorithm Unrolling	34
2.7.1. Estimación del emborronamiento en el dominio de los filtros	34
2.7.2. Minimización mediante HQS (<i>Half-quadratic Splitting</i>)	35
2.7.3. Construcción de la red	37
3. Estudio comparativo de métodos analíticos y de aprendizaje profundo para la deconvolución ciega de imágenes	41
3.1. Conjuntos de datos usados para el estudio comparativo	41
3.2. Representación de los métodos usados en la comparación	42
3.3. Resultados para el dataset de Levin et al.	43
3.4. Resultados para el dataset de Köhler et al.	46
3.5. Resultados para el dataset de Sun et al.	49
3.6. Resultados para el dataset de Nah et al.	52
4. Propuesta de combinación de métodos analíticos y de apren- dizaje profundo para la deconvolución ciega de imágenes	55
4.1. Estructura de la propuesta	56
4.2. Entrenamiento de la red	59
5. Resultados experimentales de la propuesta de combinación de métodos analíticos y de aprendizaje profundo	63
6. Conclusiones del trabajo realizado	69

Índice de figuras

1.1.	Convolución usando un filtro de desenfoque uniforme	2
1.2.	Convolución usando un filtro de subexposición	3
1.3.	Convolución usando un filtro de sobreexposición	3
1.4.	Convolución usando un filtro de desenfoque de movimiento .	4
2.1.	Estructura de la red que genera el clasificador binario de Li [1]	11
2.2.	Algoritmo de deconvolución no ciega de Fortunato [2]	21
2.3.	Estructura de la red propuesta por Tao et al. [3].	26
2.4.	Estructura en detalle de la red propuesta por Tao et al. [3] .	28
2.5.	Estructura de la red codificadora-decodificadora propuesta por [4]	30
2.6.	Bloque de diagramas que representan la forma funcional de resolver algunos pasos del algoritmo 4	38
2.7.	DUBLID usando en cascada filtros de tamaño 3×3 en lugar de filtros fijos de mayor tamaño	39
3.1.	Resultados de la deconvolución del dataset de Levin et al. . .	45
3.2.	Resultados de la deconvolución del dataset de Kohler et al. .	48
3.3.	Resultados de la deconvolución del dataset de Sun et al. . . .	51
3.4.	Resultados de la deconvolución del dataset de Nah et al. . . .	54
4.1.	Imágenes de entrada de RDBNet	56
4.2.	Comparación entre imagen borrosa, original y resultado de RDBNet	57
4.3.	Estructura de RDBNet	60
5.1.	Comparativa de los métodos analíticos y de la propuesta de mejora para el dataset de Sun et al.	66
5.2.	Comparativa de los métodos analíticos y de la propuesta de mejora para el dataset de Köhler et al.	67

Índice de cuadros

2.1.	Composición interna de la red clasificadora de Li [1]	11
3.1.	Información sobre del tipo de imágenes del dataset y la métrica que usaremos para compararlos	42
3.2.	Número de imágenes y núcleos de emborronamiento del dataset	42
3.3.	Tiempos usando el dataset de Levin et al.	44
3.4.	PSNR y SSIM del dataset de Levin et al.	44
3.5.	Tiempos usando el dataset de Köhler et al.	46
3.6.	PSNR y SSIM del dataset de Köhler et al.	47
3.7.	Tiempos usando el dataset de Sun et al.	49
3.8.	PSNR y SSIM del dataset de Sun et al.	49
3.9.	Tiempos usando el dataset de Nah et al.	52
3.10.	PSNR y SSIM del dataset de de Nah et al.	53
5.1.	PSNR, SSIM y tiempo medio total de distintos métodos y RDBNet	64

Capítulo 1

Introducción al problema de la deconvolución ciega de imágenes

1.1. Definición del problema

El objetivo de la deconvolución ciega de imagen, es recuperar una imagen \mathbf{x} a partir de una versión observada \mathbf{y} , una imagen borrosa y con ruido, cuando no conocemos el emborronamiento \mathbf{H} ni el ruido [5]. Generalmente, podemos representar la imagen \mathbf{y} como

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{n} \quad (1.1)$$

donde \mathbf{n} es el ruido. Tanto \mathbf{x} e \mathbf{y} , son vectores de tamaño $N \times 1$ que representan las imágenes correspondientes ordenadas lexicográficamente. \mathbf{H} es una matriz de tamaño $N \times N$, correspondiendo a un operador de convolución de una función de dispersión de punto (*Point Spread Function*, PSF) \mathbf{h} de tamaño $M \times 1$. En este trabajo, suponemos que \mathbf{H} es espacialmente invariante.

Antes de estudiar cómo podemos resolver este problema veremos un ejemplo del operador de convolución, y cómo influye su funcionamiento en el resultado final. Una convolución puede verse como una operación entre matrices cuya finalidad es extraer características. Para ello, representaremos una imagen como una matriz numérica con valores comprendidos en un rango limitado que determinará su profundidad de color. Generalmente, para una imagen en escala de grises, contamos con una matriz de tamaño $W \times H$, donde W y H se corresponden con el número de columnas y de

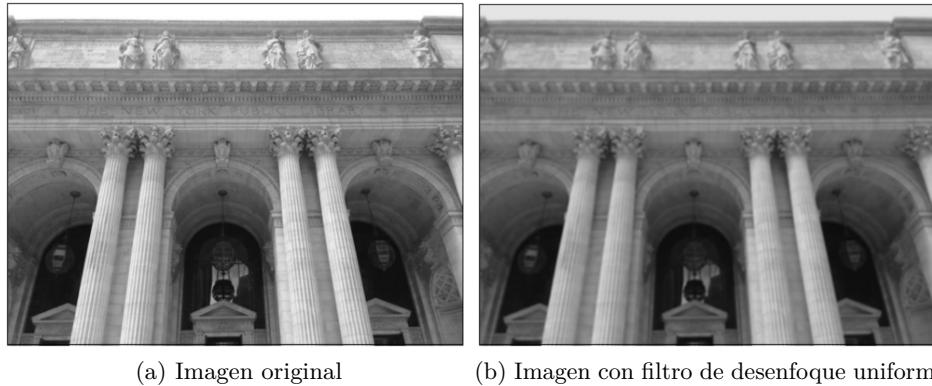


Figura 1.1: En (a) tenemos la imagen original y en (b) el resultado de aplicar el filtro de desenfoque uniforme 1.2 a la imagen original

filas de la imagen respectivamente. Los elementos de dicha matriz toman valores comprendidos entre $[0, 255]$. Para representar el filtro que aplicaremos usando la convolución usamos una matriz numérica de tamaño $N \times N$ con $N = W * H$ pero, en este caso, los valores que tenga determinará cómo se distorsionará la información de la imagen. A la matriz que contiene el filtro la denominaremos núcleo. Para evitar modificar los niveles de la imagen y que mantenga la misma media, los valores del filtro que aplicaremos deben sumar 1. También supondremos que son no negativos para que correspondan a una función de emborronamiento.

En la siguiente ecuación podemos ver un núcleo que corresponde al filtro que aplicaremos a una imagen. La fracción que aparece multiplicando al núcleo que contiene el filtro la usamos para normalizarlo y que su suma sea 1.

$$\frac{1}{9} * \begin{Bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{Bmatrix} \quad (1.2)$$

Usando la herramienta online [6], se puede ver en 1.1, cómo, tras hacer la convolución, obtenemos una imagen similar a la original, pero desenfocada. Esto se debe a que ahora, cada píxel comparte la información de sus 8 vecinos. Es decir, cada píxel contiene ahora la suma del valor del píxel central y de sus 8 vecinos, y todo ello dividido entre 9.

Cuando los valores del núcleo no suman 1, al hacer la convolución estamos modificando la media de la imagen original, lo que resulta en una imagen que puede ser más clara o más oscura. Lo veremos con un ejemplo.

Introducción al problema de la deconvolución ciega de imágenes3



(a) Imagen original



(b) Imagen con filtro de subexposición

Figura 1.2: En (a) tenemos la imagen original y en (b) el resultado de aplicar el filtro de subexposición que proporciona el núcleo anterior



(a) Imagen original



(b) Imagen con filtro de sobreexposición

Figura 1.3: En (a) tenemos la imagen original y en (b) el resultado de aplicar el filtro de sobreexposición que proporciona el núcleo anterior

En caso de que el núcleo sea $\begin{Bmatrix} 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 \end{Bmatrix}$, al realizar la convolución, la

imagen resultado tendrá en cada píxel su valor multiplicado por $\frac{1}{2}$, lo que hará obtener una imagen más oscura. Podemos ver el resultado en la figura 1.2.

Al contrario, si el núcleo fuera $\begin{Bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{Bmatrix}$, la imagen resultante de la convolución sería más clara. Podemos ver el resultado en la figura 1.3.

Teniendo en cuenta el funcionamiento del operador de convolución vamos a ver más específicamente cómo afecta en este problema y cómo podemos

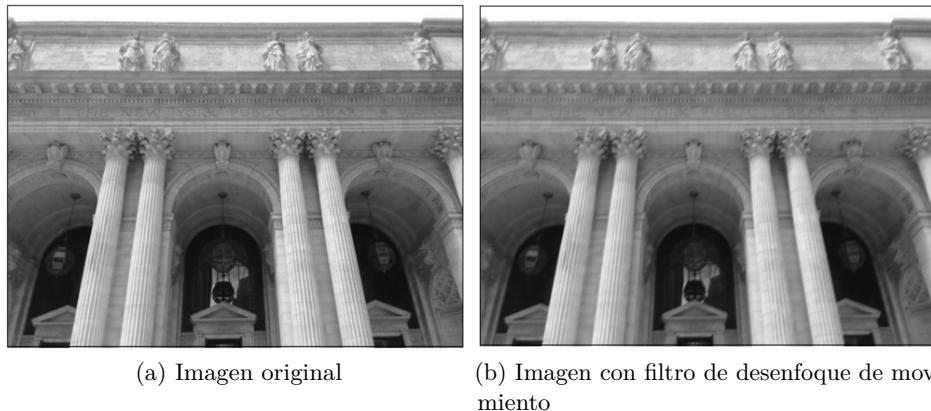


Figura 1.4: En (a) tenemos la imagen original y en (b) el resultado de aplicar el filtro de movimiento que proporciona el núcleo de la ecuación 1.3

invertirlo para solucionar el problema del emborronamiento. Para verlo de una forma más sencilla usaremos la misma imagen que hemos usado en los ejemplos anteriores y un filtro de desenfoque de movimiento 1.3 parecido al que podemos encontrarnos en los problemas de deconvolución ciega actuales. Este tipo de filtros simulan un movimiento durante la obturación de la imagen y es el principal tipo de desenfoque causante del emborronamiento que se trata en los problemas de deconvolución ciega de imágenes.

$$\frac{1}{8} * \begin{Bmatrix} 0 & 1 & 1 \\ 0 & 2 & 0 \\ 4 & 0 & 0 \end{Bmatrix} \quad (1.3)$$

Sabiendo cómo funciona la convolución, buscamos ahora solucionar su inversa, la deconvolución, que se podría definir vulgarmente como la división del filtro de desenfoque entre la imagen borrosa. Para ello, estudiaremos las aproximaciones que han sido estudiadas para resolver problemas de deconvolución ciega de imágenes.

1.2. Estado del arte

En esta sección, realizaremos un estudio del estado del arte en deconvolución ciega de imágenes. En este sentido, las técnicas analíticas han sido estudiadas durante mucho tiempo. Estos métodos describen el modelo de formación de la imagen borrosa explícitamente y lo invierten fijando el criterio para obtener la solución y el procedimiento de optimización. Esto resulta en un mayor control, aunque no ofrecen una buena solución para cualquier

Introducción al problema de la deconvolución ciega de imágenes⁵

problema. Podemos agrupar las técnicas analíticas en deterministas y estocásticas. Para el caso de las técnicas deterministas se produce una sola salida, fijando previamente el criterio de optimización. El conocimiento previo se añade con la regularización. Por otro lado, para las técnicas estocásticas, se tratan las variables $(\mathbf{x}, \mathbf{y}, \mathbf{H})$ como variables aleatorias con una distribución conjunta. En estos casos puede seguirse un enfoque completamente Bayesiano [5] para recuperar la imagen nítida. Podemos decir entonces que las técnicas analíticas hacen uso de ese conocimiento previo y lo integran regularizando mediante el uso de distribuciones a priori, mientras que para el caso de las técnicas estocásticas, partimos de un modelo de observación cuyo objetivo es hallar una imagen \mathbf{x} que maximice $p(\mathbf{x}|\mathbf{y})$ dada una imagen observada \mathbf{y} que presenta emborronamiento. A lo largo de los años, se ha usado conocimiento previo (de ahora en adelante a priori o distribuciones a priori) de la imagen diseñado para que delimite las soluciones del problema y consigamos un resultado más parecido al de la imagen original. Dentro de estas distribuciones a priori podemos resaltar las restricciones en los gradientes de la imagen. Estas son las principales encargadas de recuperar los detalles de la imagen. El resto, que suelen ser más restrictivos, se han usado para la estimación del emborronamiento, pues se necesita una solución robusta, consultar [7]. Algunos ejemplos de distribuciones a priori basados en restricciones en los gradientes de la imagen son las log-TV [8], mezcla de Gaussianos (*Mixture of Gaussians*, MoG) [9], Super Gaussianos (*Super Gaussian*, SG) [10], y los modelos basados en la norma ℓ_p/ℓ_q [11]. Al tratar con métodos analíticos, la estimación del emborronamiento y de la imagen latente final puede resolverse secuencial o alternativamente. Para el caso secuencial, primero se estima el emborronamiento que presenta la imagen usando una versión filtrada de la misma, una vez se ha estimado el emborronamiento se usa un algoritmo de deconvolución no ciega para recuperar la imagen final. Algunos ejemplos que optan por una solución secuencial los podemos ver en [9, 12, 7, 13, 10, 14, 11]. Por otro lado, para el caso de una solución alterna, se alterna entre la estimación del emborronamiento y la estimación de la imagen latente en un proceso iterativo. Algunos trabajos que buscan resolver el problema mediante este enfoque son [5, 8, 15].

Usando las técnicas analíticas tenemos que optimizar una función de energía para cada nueva imagen y emborronamiento, lo que les permite adaptarse a una amplia variedad de emborronamientos pero con un gran coste computacional. La desventaja de estas técnicas no es sólo el coste computacional. La principal ventaja que otorgan las distribuciones a priori puede convertirse en una desventaja en el caso de querer usar la misma para solucionar todos los problemas. Estas distribuciones a priori nos permiten trasladar un conocimiento previo al método con el fin de delimitar el espacio de soluciones y que la escogida sea la óptima para el problema de entrada. Esto hace que sean métodos perfectos para trabajar sobre un dominio específico.

Recientemente, gracias al campo del aprendizaje automático y el *deep learning* (DL en adelante), han surgido métodos que consiguen adaptarse a un dominio del problema más extenso proporcionando un resultado similar o incluso mejor que el de las técnicas analíticas siendo significativamente más rápidos. A diferencia de los métodos analíticos, donde el problema se define explícitamente y contamos con un conocimiento previo del dominio, en los métodos basados en DL, las redes neuronales profundas (*Deep Neural Networks*, DNN) no se benefician de ese conocimiento previo, en su lugar, usan un gran conjunto de datos para aprender de cada imagen \mathbf{y} su núcleo \mathbf{h} para llegar a estimar la imagen original \mathbf{x} . Trabajos en la eliminación del ruido de las imágenes [16], reconstrucción de imágenes antiguas (*inpainting*) [17] y super-resolución [18], han mostrado que el uso de estos métodos puede superar a los analíticos. Usando estas técnicas podemos dividir el problema de recuperación en dos subproblemas [19]: uno regularizado que usa como penalización la distancia Euclídea al cuadrado hasta la imagen (*subproblema A*). Dicha imagen se estima usando una técnica de eliminación de ruido (*subproblema B*). Dividir el problema principal en dos subproblemas nos permite generar problemas que son más sencillos de resolver individualmente. Para el subproblema A, la estimación del movimiento se puede resolver fácilmente utilizando la transformada discreta de Fourier (*Discrete Fourier Transform*, DFT), mientras que para el subproblema B podemos seleccionar cualquier algoritmo de eliminación de ruido. Para el caso del subproblema B, en lugar de establecer manualmente el término para la distribución a priori, se han propuesto redes neuronales profundas para resolver el problema de eliminación de ruido de forma automática. Por ejemplo, en [20] y [21] el operador proximal [22] asociado al subproblema B se reemplaza por una red neuronal para eliminar el ruido. De manera similar, [23] usa una red residual para estimar el ruido en el subproblema B, que luego se resta de la imagen estimada para obtener la restaurada. Para la deconvolución de imagen no ciega, RGDN [24] integra ambos subproblemas en una red convolucional recurrente, donde el gradiente de la distribución a priori de imagen se reemplaza por un bloque CNN común. Los métodos presentados usan una aproximación alternativa en la que usan una red para el cálculo de \mathbf{x} , \mathbf{H} o ambos, lo que resulta en un alto coste computacional.

A pesar de que los enfoques actuales hagan uso de una combinación de las técnicas analíticas junto a las de DL también podemos encontrar modelos que sólo hacen uso del DL para resolver ambos subproblemas. Aquellos modelos que sólo usan técnicas de DL para resolver el problema de la deconvolución ciega restringen el dominio del problema a tipos de emborronamiento específicos, como desenfoque de movimiento, en los que no estiman explícitamente el emborronamiento. En [25] se utiliza una red neuronal convolucional (en adelante CNN, *convolutional neural network*) residual multi-escala para eliminar el desenfoque de movimiento no uniforme

Introducción al problema de la deconvolución ciega de imágenes7

de las imágenes. En [26] se propone el uso de una red generativa adversaria (*Generative adversarial network*, GAN) y se mejora en [27]. Aunque estos modelos obtienen buenos resultados en un tiempo mucho menor que los modelos analíticos y combinados, carecen de flexibilidad, lo que limita su aplicabilidad solo a casos concretos de emborronamiento.

1.3. Objetivos

En este trabajo proponemos un método que combina la flexibilidad de las técnicas analíticas junto a la velocidad de las técnicas basadas en DL para resolver el problema de la deconvolución ciega de forma eficaz y eficiente. Para ello buscaremos cumplir los siguientes objetivos:

- Estudio y comparación de algunos métodos del estado del arte
- Selección del mejor método
- Propuesta de mejora que refina el resultado del método escogido y genera una mejor salida
- Validación de la mejora propuesta respecto de los métodos estudiados

1.4. Metodología

Para cumplir el primer objetivo hemos seleccionado una gran variedad de métodos que tratan el problema de la deconvolución ciega haciendo uso de distintas técnicas. Dentro de este estudio podremos encontrar tanto técnicas analíticas como técnicas basadas en DL.

Una vez hemos estudiado los métodos, probaremos cada uno de ellos en cuatro conjuntos de datos que representan distintos posibles escenarios reales. Para medir los resultados usaremos el PSNR (*Peak Signal-to-Noise Ratio*), mediante el cual mediremos la calidad de reconstrucción de la imagen estimada, y el SSIM (*Structural Similarity Index Measure*), que nos proporcionará la similitud que presentan la imagen estimada y la original. Para que la comparativa sea más justa también tendremos en cuenta el tiempo que tarda cada método en ofrecer una salida. De esta forma, no sólo favorecemos a aquél método que ofrezca una buena estimación, si no al que proporciona un buen resultado en un tiempo razonable. De esta comparativa escogeremos un método y propondremos un algoritmo que mejora su salida.

Una vez hemos seleccionado el mejor método nos basamos en las técnicas de DL para presentar una red neuronal convolucional que mejora las imágenes estimadas de dicho método eliminando los artificios causados durante la

deconvolución. Para entrenar la red hemos usado un conjunto de más de 300.000 imágenes.

Para cumplir el último objetivo realizaremos una nueva comparativa entre la combinación del método escogido junto al algoritmo respecto del conjunto de métodos estudiados previamente siguiendo los mismos criterios que hemos usado para la comparativa del estado del arte.

Capítulo 2

Métodos analíticos y de aprendizaje profundo para la deconvolución ciega de imágenes

En este capítulo realizamos un estudio exhaustivo del funcionamiento de métodos analíticos y de aprendizaje profundo. Veremos en detalle las distintas formas de afrontar el problema de la deconvolución y cómo, dependiendo del dominio del problema, ciertas aproximaciones son más indicadas que otras para solucionarlo. Para ello, la información que muestro en las distintas secciones la extraigo de los trabajos publicados de los autores.

2.1. Blind Image Deblurring via Deep Discriminative Priors

Para la solución de este problema, los autores de [1] se basan en una versión de 1.1, donde los problemas basados en optimización buscan resolver el problema

$$\min_{x,h} \|x \otimes h - y\|_2^2 + \gamma \|h\|_2^2 + \lambda p(x) \quad (2.1)$$

donde x es la imagen nítida, h el núcleo, y la imagen que presenta embotamiento, $p(x)$ el término de regularización (distribución a priori) y $\|\cdot\|_2^2$ denota la norma L_2 . Los parámetros λ y γ son pesos de regularización. El modelo favorece aquellas imágenes que presentan un buen resultado inicial. Para ello cuentan con un clasificador binario basado en redes neuronales profundas

10 2.1. Blind Image Deblurring via Deep Discriminative Priors

que determina si la entrada es una imagen nítida o no, y desarrolla una aproximación numérica basada en el método de división media cuadrática (*half-quadratic splitting*, HQS) y el algoritmo de gradiente descendente.

Para entrenar la red y manejar distintos tamaños de imagen se usa una capa de agrupamiento medio global y se usa una estrategia de entrenamiento multiescala para asegurar que el clasificador da un buen resultado con imágenes de cualquier tamaño. Se usa el clasificador aprendido como un término de regularización de la imagen latente (nítida) y se desarrolla un algoritmo de optimización eficiente para resolver la estimación del núcleo que presenta un filtro de desenfoque de movimiento.

El método propuesto por los autores, permite, además de resolver el problema de la deconvolución ciega, extender el dominio del problema y conseguir eliminar la posible neblina de la imagen (*dehazing*).

2.1.1. Clasificador binario

Se entrena una red neuronal profunda que determina la probabilidad de que un entrada sea una imagen borrosa. Para ello, en lugar de usar capas totalmente conectadas se usa una capa de agrupamiento global, que convierte los mapas de características de distintos tamaños en un sólo escalar antes de la capa sigmoide. En la figura 2.1 se aprecia la arquitectura y los parámetros del clasificador. En la tabla 2.1 podemos ver las distintas capas de la red. “CR” denota una capa convolucional seguida de una función no lineal ReLU, “M” denota una capa de agrupación máxima, “C” denota una capa deconvolucional , “G” indica una capa de agrupamiento promedio global y “S” indica la función sigmoidea no lineal. La columna *stride* hace referencia al número de píxeles que se usamos para desplazar el filtro sobre la matriz a la que aplicamos la convolución. La columna *padding* tiene el valor de 1 si en esa capa se rellena el filtro para que tenga el mismo tamaño que la matriz o 0 para el caso contrario.

2.1.2. Función de pérdida

Siendo x la imagen de entrada y θ los parámetros de la red a ser optimizados. La red aprende la función $f(x; \theta) = P(x \in \text{Borrosa} | x)$ que predice la probabilidad de que la imagen de entrada sea borrosa. Para optimizar la red

¹Imagen obtenida del artículo [1] en la que los autores presentan la arquitectura de la red clasificadora.

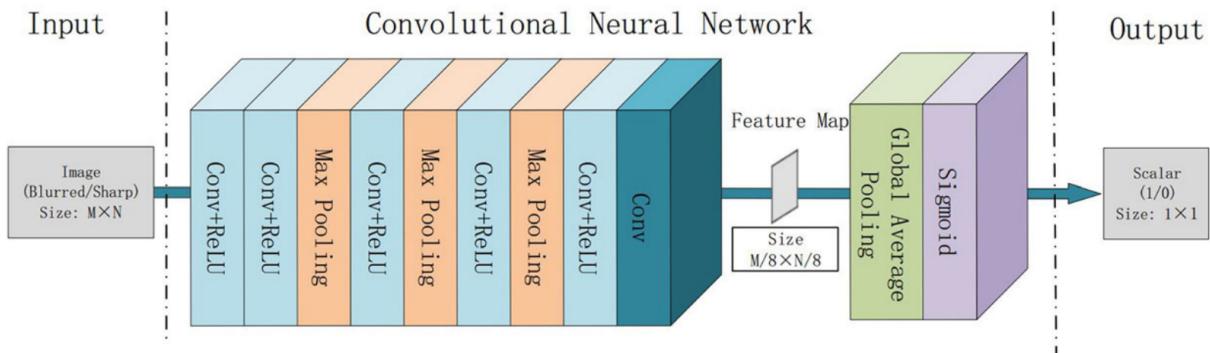


Figura 2.1: Estructura de la red usada para generar el clasificador binario.¹

Capas	Tamaño del filtro	<i>Stride</i>	<i>Padding</i>
CR1	$3 \times 3 \times 1 \times 64$	1	1
CR2	$3 \times 3 \times 64 \times 64$	1	1
M3	2×2	2	0
CR4	$3 \times 3 \times 64 \times 64$	1	1
M5	2×2	2	0
CR6	$3 \times 3 \times 64 \times 64$	1	1
M7	2×2	2	0
CR8	$3 \times 3 \times 64 \times 64$	1	1
C9	$3 \times 3 \times 64 \times 1$	1	1
G10	$(M/8) \times (N/8)$	1	0
S11	-	-	-

Cuadro 2.1: Composición interna de la red neuronal usada por los autores para generar el clasificador binario.

12 2.1. Blind Image Deblurring via Deep Discriminative Priors

usan la siguiente función de pérdida:

$$L(\theta) = -\frac{1}{N} \sum_{i=1}^N \hat{y}_i \log(y_i) + (1 - \hat{y}_i) \log(1 - y_i) \quad (2.2)$$

donde N es el número de muestras de entrenamiento en un batch, $y_i = f(x_i)$ es la salida del clasificador y \hat{y}_i es la etiqueta de la imagen de entrada, que será igual a 1 para el caso de una imagen borrosa y 0 para una nítida.

2.1.3. Modelo de deconvolución ciega

El modelo propuesto para la solución a la estimación del emborronamiento uniforme es el siguiente:

$$\min_{x,h} \|x \otimes h - y\|_2^2 + \gamma \|h\|_2^2 + \mu \|\nabla x\|_0 + \lambda f(x) \quad (2.3)$$

donde γ , μ y λ son pesos. Optimizan 2.3 resolviendo x y h alternativamente. Dividen entonces el problema en el subproblema x:

$$\min_x \|x \otimes h - y\|_2^2 + \mu \|\nabla x\|_0 + \lambda f(x) \quad (2.4)$$

y el subproblema h:

$$\min_h \|x \otimes h - y\|_2^2 + \gamma \|h\|_2^2 \quad (2.5)$$

Para resolver el subproblema x, los autores adoptan el método de división media cuadrática de Xu. [28] para introducir las variables auxiliares u y $g = (gh, gv)$ con respecto a la imagen y sus gradientes en direcciones horizontales y verticales, respectivamente. Minimizando x , g y u alternativamente resuelven el subproblema evitando así también minimizar las funciones no convexas. Para resolver el problema de estimar la imagen llevan el problema al dominio de Fourier donde, una vez obtenida una solución para x se pueden resolver g y u .

En el caso del subproblema h, los autores se basan en [29, 30, 31] para estimar el núcleo usando los gradientes de la imagen. Para ello también usan la transformada de Fourier (*Fast Fourier Transform*, FFT). Después fijan a 0 todos los elementos negativos y normalizan el núcleo para que en conjunto sume 1. Tras esto, para la deconvolución no ciega, usan el algoritmo propuesto por Pan et al. en [30].

2.2. Fast Bayesian blind deconvolution with Huber Super Gaussian priors

Los autores de [32] tratan principalmente el uso de distribuciones a priori de imagen llamados Huber Super Gaussian, los cuales mejoran la estabilidad numérica y la velocidad de convergencia. También implementan un método rápido y fiable que evita converger en puntos estacionarios.

2.2.1. Estimación del emborronamiento

En esta aproximación, los autores optan por resolver primero el problema del emborronamiento y para ello, usando el enfoque de expectativas de maximización (*expectation–maximization*, EM) resuelven

$$\begin{aligned} \hat{h} &= \arg \max_h \int p(x, h, y) dx \\ \text{s.t. } h(i) &\geq 0, \sum_i h(i) = 1 \end{aligned} \tag{2.6}$$

siendo h el núcleo, x la imagen nítida e y la imagen borrosa. Teniendo \hat{h} pueden usar una distribución condicionada $p(x|\hat{h}, y)$ para obtener la imagen original, pero debido a las distribuciones que usan, una buena estimación del emborronamiento resulta en una peor estimación de la imagen latente debido a la eliminación de algunos detalles de la misma durante la regularización.

Desarrollando 2.6 después de haber expresado la ecuación como una función de energía asociada a una distribución gaussiana se obtiene finalmente

$$\hat{h} = \arg \max_h E[\log(Q(y, x, h, \hat{\varepsilon}))]_{\hat{q}(x)} \tag{2.7}$$

$$\hat{q}(x) \propto Q(y, x, \hat{h}, \hat{\varepsilon}) \tag{2.8}$$

$$\hat{\varepsilon} = \arg \max_{\varepsilon \geq 0} E[\log(Q(y, x, \hat{h}, \varepsilon))]_{\hat{q}(x)} \tag{2.9}$$

Siendo ahora $\hat{q}(x)$ 2.8 una aproximación de la distribución $p(x, |\hat{h}, y)$ y $\hat{\varepsilon}$ un hiperparámetro que se actualizará en cada iteración y formará parte de la estimación de la imagen.

2.2.2. Estimación de la imagen y de los parámetros

Para la estimación de la imagen final, los autores se basan en el algoritmo 1, donde en sus primeros pasos fijan las distribuciones a priori (explicadas en la sección 4 de [32]) y los parámetros iniciales para después resolver alternativamente la estimación de la imagen usando un método de optimización basado en la dirección alterna de multiplicadores (*Alternating Direction Method of Multipliers*, ADMM).

Algorithm 1 Algoritmo de estimación de imagen Xu et al. [32]

Require: y, h, x inicial, $\rho_\epsilon(s), \beta_\nu, \sigma$

- 1: Precomputar $\mathcal{H}, \bar{\mathcal{H}}, \bar{\mathcal{H}}\mathcal{F}y, \Lambda_h, \Lambda_\gamma$
 - 2: $d\nu_\gamma = 0, C_x = 0$
 - 3: Inicializar ε_γ con una constante grande o 2.13
 - 4: **while** la variación relativa de x sea grande **do**
 - 5: Usar ADMM para actualizar la solución x para 2.12
 - 6: Actualizar ε_γ usando 2.13
 - 7: Aproximar $C_x(i, i)$ con $1/C_x^{-1}(i, i)$
 - 8: **end while**
 - 9: Devolver x
-

Para la imagen final, se obtiene de la ecuación 2.8,

$$\log \hat{q}(x) = -\frac{1}{2\sigma^2} \|\hat{H}x - y\|_2^2 - \frac{1}{2} \sum_{\gamma=1}^L x_\gamma^T \text{diag}(\hat{\varepsilon}_\gamma) x_\gamma \quad (2.10)$$

que es una distribución gaussiana multivariable con una matriz de precisión

$$C_x^{-1} = \frac{1}{\sigma^2} \hat{H}^T \hat{H} + \sum_{\gamma=1}^L F_\gamma^T \text{diag}(\hat{\varepsilon}_\gamma) F_\gamma \quad (2.11)$$

donde F_γ es una matriz circulante de convolución de tamaño $N \times N$ formada por el núcleo f_γ . La media \hat{x} se usa para estimar x , que se obtiene resolviendo el siguiente sistema lineal de ecuaciones

$$C_x^{-1} \hat{x} = \frac{1}{\sigma^2} \hat{H}^T y \quad (2.12)$$

Para el parámetro $\hat{\varepsilon}$, desarrollando 2.9 obtienen

$$\hat{\varepsilon}_\gamma(i) = \frac{\rho'_\epsilon(\nu_\gamma(i))}{\nu_\gamma(i)} = \min(\rho'(\nu_\gamma(i))), \rho'(\epsilon)/\epsilon) \quad (2.13)$$

donde $\nu_\gamma(i) = \sqrt{E[x_\gamma^2(i)]}$, $1 \leq i \leq N$, y el valor esperado se calcula usando la distribución $\hat{q}(x)$ y la aproximación $C_x \approx (\text{diag}(C_x^{-1}))^{-1}$

Dada una estimación del núcleo, se necesita resolver el sistema lineal de 2.12 y actualizar los pesos de $\hat{\epsilon}$ 2.13. Para resolver 2.12, los autores tratan la solución del sistema lineal como el minimizador de una función cuadrática, transformando así el problema en uno con restricciones introduciendo las variables $\nu_\gamma = f_\gamma \otimes x$. Consecuentemente, para un dado emborronamiento H , \hat{x} se puede reescribir como el minimizador de

$$f(x, \nu_\gamma) = \frac{1}{2} \|Hx - y\|_2^2 + \sigma^2 \sum_{\gamma=1}^L \nu_\gamma^T \text{diag}(\hat{\epsilon}_\gamma) \nu_\gamma, \text{s.t. } \nu_\gamma = F_\gamma x \quad (2.14)$$

que resuelven de forma eficiente usando ADMM. En particular, el método de Lagrange

$$\mathcal{L}_{\beta_\nu}(x, \nu_\gamma, d\nu_\gamma) = \frac{1}{2} \|Hx - y\|_2^2 + \sigma^2 \sum_{\gamma=1}^L \nu_\gamma^T (\hat{\epsilon}_\gamma) \nu_\gamma + \frac{\beta_\nu}{2} \sum_{\gamma=1}^L \|\nu_\gamma + F_\gamma x + d\nu_\gamma\|_2^2 \quad (2.15)$$

donde β_γ es un parámetro de penalización que hace cumplir las restricciones en 2.14 y controla el ratio de convergencia, $d\nu_\gamma$ son los multiplicadores Lagrangianos. ADMM es un método iterativo en el que $\bar{\mathcal{H}}$ es la conjugada compleja de \mathcal{H} y Λ_h y Λ_γ son los valores propios de $H^T H$ y $F_\gamma^T F_\gamma$

2.2.3. Estimación del emborronamiento

Para resolver el problema del emborronamiento, los autores siguen el algoritmo 2, que es un algoritmo de optimización iterativo basado en ADMM que les permite obtener una estimación del núcleo.

Algorithm 2 Algoritmo de estimación del núcleo Xu et al. [32]

Require: y, x, h inicial, β_h

- 1: Precomputar D_x usando 2.18
 - 2: $d\nu_\gamma = 0$, $C_x = 0$
 - 3: **while** no converge **do**
 - 4: Actualizar \mathcal{H} usando 2.20
 - 5: Actualizar h hallando el punto KKT de 2.21
 - 6: $dH = dH + \mathcal{H} - \mathcal{F}Ph$
 - 7: **end while**
 - 8: Devolver x
-

Tras inicializar los parámetros estiman el emborronamiento resolviendo

$$\hat{h} = \arg \min_h \|H\hat{x} - y\|_2^2 + h^T D_x h \quad (2.16)$$

$$\text{s.t. } h(i) \geq 0, \sum_i h(i) = 1 \quad (2.17)$$

donde D_x es la matriz de tamaño $K \times K$ dada por

$$D_x(m, n) = \sum_{j=1}^N C_x(m+j, n+j) \quad (2.18)$$

Teniendo ya una estimación previa de la imagen, se usa ADMM para resolver el problema de optimización de 2.16 con no negatividad y restricciones de normalización 2.17. Introduciendo la restricción $\mathcal{H} = \mathcal{F}Ph$, se obtiene \hat{h} en 2.16 como el minimizador de

$$\begin{aligned} g(h, \mathcal{H}) &= \|\mathcal{H} \circ \mathcal{X} - \mathcal{F}y\|_2^2 + h^T D_x h \\ \text{s.t. } h(i) &\geq 0, \sum_i h(i) = 1, \mathcal{H} = \mathcal{F}Ph \end{aligned} \quad (2.19)$$

donde $\mathcal{X} = \mathcal{F}Ph$ denota la transformada Fourier de x y \circ la multiplicación elemento a elemento. Para este algoritmo las actualizaciones para \mathcal{H} y h son dadas por

$$\mathcal{H} = \frac{\overline{\mathcal{H}} \circ (\mathcal{F}y) + \beta_H(\mathcal{F}Ph - dH)}{\overline{\mathcal{X}} \circ \mathcal{X} + \beta_H} \quad (2.20)$$

$$\begin{aligned} h &= \arg \min_h \|h - P^T \mathcal{F}^{-1}(\mathcal{H} + dH)\|_2^2 + h^T D_x h \\ \text{s.t. } h(i) &\geq 0, \\ &\sum_i h(i) = 1 \end{aligned} \quad (2.21)$$

El método propuesto divide el problema en dos subproblemas más sencillos, al igual que vimos en la sección anterior con la aproximación de Li [1]. El primer subproblema se puede resolver directamente en el dominio de Fourier mientras que el otro, al ser de menor complejidad se puede resolver iterativamente usando ADMM. Los autores aseguran que con tan sólo 5 iteraciones, para la mayoría de los casos, ADMM converge y ofrece la solución óptima.

2.2.4. Estimación multiescala del emborronamiento

Los autores introducen una aproximación multiescala para el subproblema de la estimación del núcleo de emborronamiento. La estimación de este núcleo en una sola escala puede sufrir de mínimos locales, sobretodo si el núcleo es muy grande, lo que lleva a empeorar el resultado a la hora de estimar la imagen final. Esta aproximación también facilita el trabajo a la hora de estimar el núcleo en los niveles más bajos donde la imagen es más pequeña. Usar una aproximación multiescala mejora tanto en costo de cómputo, pues al tratar con una imagen más pequeña el método converge antes ofreciendo una buena estimación, como en calidad, ya que al tratar con los detalles más relevantes de la imagen en los niveles inferiores se consigue que la estimación del núcleo tenga una buena base y, una vez ya contamos con esa base, es más fácil recuperar los detalles en los niveles superiores.

2.2.5. Reconstrucción de la imagen final

La imagen \hat{x} estimada de la multiescala es apropiada para estimar el emborronamiento, pero está muy suavizada y se pierden las altas frecuencias de la imagen (los bordes y los detalles). Esto se debe a que el peso $\sigma^2 \varepsilon_\gamma$ es demasiado grande como para recuperar los detalles. Para solucionar este problema se reconstruye la imagen x^* resolviendo la siguiente ecuación

$$x^* = \arg \min_x \frac{1}{2} \|Hx - y\|_2^2 + \frac{\lambda_x}{p} \sum_\gamma \|f_\gamma \otimes x\|_p \quad (2.22)$$

donde f_γ es un conjunto de filtros derivativos, p es un parámetro que se encuentra normalmente en el rango $[0.6, 0.8]$ y el regularizador ℓ_p se encuentra en la forma HSG (*Huber Super Gaussian*).

2.3. Fast High-Quality non-Blind Deconvolution Using Sparse Adaptive Priors

Fortunato [2] presenta con este método una propuesta eficiente para la deconvolución no ciega, es decir, aquella en la que ya conocemos el núcleo de emborronamiento, que es más rápida que los métodos actuales y que proporciona un alto PSNR (*Peak signal-to-noise ratio*).

Para ello, se basan en una técnica de regularización que penaliza los valores pequeños de la derivada, que tienden a asociarse con el ruido, y preserva los valores grandes, los cuales suelen asociarse a los bordes de la imagen. A la hora de realizar la deconvolución expresan el problema como un sistema lineal que puede resolverse eficientemente.

Como hemos visto previamente en el capítulo introductorio, para resolver el problema de la deconvolución debemos hallar un núcleo que nos proporcione una buena imagen latente a la hora de deconvolucionar la imagen borrosa 1.1. Con este método usaremos la estimación resultante de ejecutar los métodos de Li [1] y Zhou [32] sobre una imagen borrosa para volver a deconvolucionar esa misma imagen. Para ello, los autores buscan resolver el siguiente problema

$$\hat{x} = \arg \min_x \delta(x) \quad (2.23)$$

$$\delta(x) = \|Hx - y\|_2^2 + \sum_{s=1}^5 \lambda_s \|d_s x - w_s\|_2^2 \quad (2.24)$$

donde \mathbf{x} , \mathbf{H} e \mathbf{y} son la imagen nítida, una matriz correspondiente al operador de convolución de una PSF de emborronamiento h y la imagen borrosa, respectivamente. Las matrices d_s , con $s \in \{1, \dots, 5\}$, representan la primera y segunda derivada horizontal, vertical y diagonal de los operadores del filtro: d_h , d_v , d_{hh} , d_{vv} y d_{hv} , respectivamente. λ_s son pesos positivos, y w_s son las respuestas ideales o esperadas de esos filtros para la imagen final (p. ej., $w_s = d_s f$). El uso de w_s en 2.24 permite especificar un conjunto de distribuciones a priori sobre las derivadas de la imagen x . Por tanto, τ es un umbral que representa algún nivel de ruido, y $dx_{i,j} = (d_s X)_{i,j}$ son las derivadas de la imagen x en las coordenadas (i, j) . Para cada píxel, $w_s = 0$ si $|dx_{i,j}| < \tau$; en otro caso, $w_s = dx_{i,j}$. Por tanto, en 2.24 se penaliza selectivamente la aparición de píxeles con pequeñas derivadas (que suelen representarse como ruido), y preserva los bordes fuertes, representados por valores altos en las derivadas de la imagen.

Se resuelve 2.23 haciendo la diferencia de 2.24 con cada píxel esperado o ideal x_m , siempre y cuando las expresiones resultantes sean 0. Esto produce

el siguiente sistema lineal, donde \square^T indica la matriz traspuesta:

$$(H^T H + \sum_{s=1}^5 \lambda_s d_s^T d_s) \hat{x} = H^T y + \sum_{s=1}^5 \lambda_s d_s^T w_s \quad (2.25)$$

La ecuación 2.25 se puede reescribir como:

$$a \hat{f} = b \quad (2.26)$$

donde

$$\begin{aligned} a &= H^T H + \sum_{s=1}^5 \lambda_s d_s^T d_s, \\ b &= H^T y + \sum_{s=1}^5 \lambda_s d_s^T w_s \end{aligned}$$

La matriz cuadrada a es un operador de convolución, ya que es la suma de productos de operadores de convolución.

La ecuación 2.26 se puede expresar en el dominio de las frecuencias como:

$$A \circ \hat{F} = B \quad (2.27)$$

donde

$$\mathbf{A} = \mathbf{H}^* \circ \mathbf{H} + \sum_{s=1}^5 \lambda_s D_s^* \circ D_s \quad (2.28)$$

$$\mathbf{B} = \mathbf{H}^* \circ \mathbf{Y} + \sum_{s=1}^5 \lambda_s D_s^* \circ W_s \quad (2.29)$$

Aquí, \square^* representa un conjugado complejo, y \circ representa el producto elemento a elemento matricial. $\mathbf{B} = \mathcal{F}(b)$, $\mathbf{H} = \mathcal{F}(H)$, $\mathbf{Y} = \mathcal{F}(y)$, $D_s = \mathcal{F}(d_s)$ y $W_s = \mathcal{F}(w_s)$ son las variables equivalentes en el dominio de las frecuencias de b , h , y , d_s y w_s , respectivamente. $\mathcal{F}(\cdot)$ es el operador de la transformada de Fourier. \hat{x} puede entonces obtenerse como

$$\hat{x} = \mathcal{F}^{-1}(\mathbf{B} ./ \mathbf{A}) \quad (2.30)$$

donde $./$ representa la división matricial elemento a elemento.

Algunas de las propiedades de la ecuación 2.24 son:

- Si todos los coeficientes w_s se establecen en cero, se reduce a una regularización estándar de Tikhonov, que introduce artificios cerca de los bordes.
- En el caso contrario, si uno pudiera conocer de antemano las derivadas de x y usarlas como w_s , la ecuación 2.30 daría una muy buena aproximación a x . La calidad de tal aproximación está limitada por la desviación estándar del ruido agregado a y .

Para comprender cómo funciona el método veremos y analizaremos los pasos del algoritmo de deconvolución 3 que proponen los autores. En la figura 2.2 podemos ver una representación visual del algoritmo que sigue dicho método.

Algorithm 3 Algoritmo de deconvolución no ciega de Fortunato et al. [2]

Require: \mathbf{y} (imagen borrosa) , \mathbf{h} (núcleo de emborronamiento), λ_s (pesos de regularización)

- 1: Evaluar $\hat{x}^{(0)}$ con $w_s = 0$
 - 2: $\hat{x}^{(1)} \leftarrow \mathcal{EPS}(\hat{x}^{(0)})$
 - 3: Calcular w_s usando $\hat{x}^{(1)}$
 - 4: Evaluar \hat{x} con $w_s = \mathcal{F}(w_s)$
-

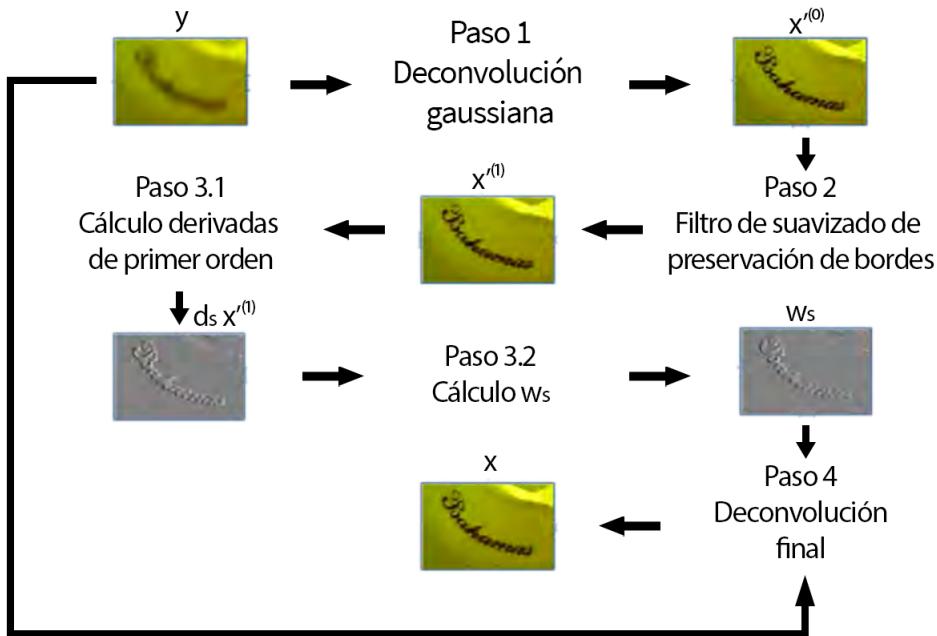


Figura 2.2: Representación gráfica de los pasos del algoritmo que sigue el método de Fortunato [2].²

Paso 1 - Deconvolución gaussiana: obtenemos una aproximación inicial $\hat{x}^{(0)}$ a x deconvolucionando la imagen borrosa y usando la regularización estándar de Tikhonov (es decir, $w_s = 0$). Llamamos a este paso deconvolución gaussiana en el algoritmo 3. Aunque $\hat{x}^{(0)}$ sufre de artificios cerca de los bordes y ruido, proporciona una buena estimación de los bordes de x ;

Paso 2 - Filtro de suavizado de preservación de bordes: una nueva estimación $\hat{x}^{(1)}$ se obtiene aplicando un filtro de suavizado que preserva los bordes para $\hat{x}^{(0)}$ para reducir ruido mientras se conservan bordes importantes.

Paso 3 - Evaluar los parámetros de regularización: Calculamos los parámetros de regularización reales w_s como un conjunto de escaso primer y segundo orden derivados de $\hat{x}^{(1)}$ usando la ecuación 2.31.

Paso 4 - Deconvolución final: la imagen final \hat{x} finalmente se obtiene de y y w_s usando las ecuaciones 2.27, 2.28 y 2.29.

²Imagen obtenida del artículo de Fortunato [2] y traducida al castellano.

Los parámetros de regularización que usan para estimar la imagen final (paso 4 de 3) se obtienen resolviendo la ecuación 2.31.

$$w_s = \phi(d_s \hat{x}^{(1)}) = \frac{d_s \hat{x}^{(1)}}{\left(\frac{\tau}{d_s \hat{x}^{(1)}}\right)^4 + 1} \quad (2.31)$$

donde $d_s = \{d_h, d_v, d_{hh}, d_{vv}, d_{hv}\}$, y la exponencial $\left(\frac{\tau}{d_s \hat{x}^{(1)}}\right)^4$ es elemento a elemento. La ecuación 2.31 penaliza las distribuciones w_s poniendo a cero sus valores. De esta forma se consigue una transición más gradual, lo que resulta en una mejora de PSNR del paso 2 al 4 del algoritmo propuesto por los autores.

En el artículo escrito por los autores introducen también un problema causado por la deconvolución y que añade artificios en los bordes de la imagen (*ringing artifacts*). Para solucionarlo optan por añadir un marco a la imagen antes de realizar la deconvolución y después recortan los píxeles sobrantes. En el artículo se explica más detenidamente los valores que tienen en cuenta para solucionar este problema y generar una imagen final más fiel a la original.

2.4. Non-uniform Deblurring for Shaken Images

El enfoque de los autores de este artículo se centra en un modelo geométrico en el que asumen que, en la mayoría de los casos, el emborronamiento se debe a un movimiento de la cámara, la rotación de la cámara durante la exposición tiene un mayor efecto que la traslación. Asumen que el único movimiento significante de la cámara es la rotación sobre su centro óptico, y que la escena fotografiada es estática.

2.4.1. Modelo de restauración

Las cámaras cuentan con un conjunto discreto de píxeles, cuya salida es una imagen $y \in \mathbb{R}^N$, con $N = H \times W$ píxeles para una imagen con H filas y W columnas. Consideran una imagen borrosa y generada por una imagen nítida $x \in \mathbb{R}^N$ y un conjunto de pesos $\mathbf{w} \in \mathbb{R}^N$, cuyo tamaño $K = N_X \times N_Y \times N_Z$ es controlado por el número de pasos de rotación sobre cada eje. Análogamente, con una convolución en la que se aplica un filtro que causa emborronamiento, consideran w como el núcleo que produce la imagen borrosa. Cada elemento ω_k corresponde a la orientación de la cámara θ_k , y consecuentemente a una homografía \mathbf{H}_k , y en general w será muy disperso, ya que la cámara habrá sufrido alguna de estas orientaciones durante la exposición.

En un caso ideal, cada punto del sensor ve un punto de la escena durante la exposición, dando como resultado una imagen nítida. En caso de que durante la exposición se sufra algún movimiento, parametrizado por $\theta(t)$, cada rayo de la escena estática traza una secuencia de puntos en la imagen. Para cada punto i observado en la imagen borrosa se puede trazar el conjunto de rayos $i'(t)$ de modo que:

$$i'(t) \sim \mathbf{H}_t i \quad (2.32)$$

donde \mathbf{H}_t denota la homografía inducida por la rotación $\theta(t)$, \sim denota igualdad a escala. La imagen observada y es la integral sobre el tiempo de exposición T de todas las versiones transformadas proyectivamente de x , más algo de ruido ϵ de observación:

$$y(i) = \int_0^T x(\mathbf{H}_t i) dt + \epsilon \quad (2.33)$$

donde, con un ligero abuso de la notación, $\mathbf{H}_t i$ denota coordenadas no homogéneas de un punto en x .

En general, una sola imagen borrosa no tiene información temporal asociada, por lo que es conveniente reemplazar la integral temporal de 2.33

por una integral con pesos sobre el conjunto de todas las posibles rotaciones \mathcal{R} :

$$y(i) = \int_{\mathcal{R}} x(\mathbf{H}_\theta i) \omega(\theta) d\theta + \epsilon \quad (2.34)$$

donde la función con pesos $\omega(\theta)$ corresponde con el tiempo que la cámara pasa en la orientación θ mientras el disparador está abierto y el sensor está captando información.

Discretizando la ecuación 2.34, cada píxel y_i observado, se puede modelar como:

$$y_i = \sum_k \left(\sum_j C_{ijk} x_j \right) \omega_k + \epsilon \quad (2.35)$$

donde i , j y k son índices de la imagen observada, la imagen nítida y el núcleo de emborronamiento, respectivamente. Para un píxel g_i observado, con un vector de coordenadas v_i , la sumatoria $\sum_k C_{ijk} x_j$ interpola con el punto $f(\mathbf{H}_k v_i)$ en la imagen nítida, con C_{ijk} siendo coeficientes de una interpolación bilineal, por ejemplo. Gracias a la forma bilineal de 2.35, se ha de tener en cuenta que cuando el núcleo de emborronamiento o la imagen nítida se conocen, la imagen borrosa es lineal en las siguientes incógnitas.

$$\begin{aligned} y &= Ax + \epsilon \\ y &= Bw + \epsilon \end{aligned} \quad (2.36)$$

donde $A_{ij} = \sum_k C_{ijk} \omega_k$ y $B_{ik} = \sum_j C_{ijk} x_j$. En la primera forma, $A \in \mathbb{R}^{N \times N}$ es una gran matriz dispersa, donde cada una de las filas contiene un filtro de desenfoque local que actúa sobre x para generar un píxel borroso. En la segunda forma, cuando se conoce la imagen nítida, cada columna de $B \in \mathbb{R}^{N \times K}$ contiene una proyectiva transformada de la imagen nítida.

2.4.2. Estimación del emborronamiento

Para tratar el problema de la deconvolución ciega, lo primero que realizan es estimar el núcleo que ocasiona el emborronamiento de la imagen. Para ello, aplican las distribuciones a priori de Fergus et al. [33]. Asumen que el ruido observado ϵ es gaussiano, y para que el usuario no tenga que introducir manualmente la varianza del ruido σ^2 , la varianza inversa $\beta_\sigma = \sigma^{-2}$ se considera como una variable latente. Siguen el enfoque de MacKay [34] y recogen las variables x , w y β_σ en un conjunto Θ . El objetivo es encontrar la mejor distribución factorizada $q(\Theta)$ que después se ajuste a

$p(\Theta|y)$ minimizando la función de coste ([34], ecuación 10) sobre la forma y los parámetros de $q(\Theta)$:

$$\mathcal{C}_{KL} = \int q(\Theta) \left[\ln \frac{q(\Theta)}{p(\Theta)} - \ln p(g|\Theta) \right] d\Theta \quad (2.37)$$

Para el modelo propuesto por los autores, el valor óptimo $q(\Theta)$ tiene la misma forma que en [34], aunque las ecuaciones de actualización cambian significativamente. Habiendo encontrado la distribución óptima $q(\Theta)$ se considera que $q(w)$ es el núcleo de emborronamiento óptimo.

2.4.3. Reconstrucción de la imagen

De mismo modo que otros autores, en este artículo buscan resolver el problema de la estimación de la imagen final invirtiendo 1.1. Este método hace uso del algoritmo de deconvolución Richardson-Lucy que puede ser aplicado a sistemas lineales generales y a emborronamientos convolucionales, usando la notación de 2.36. Este algoritmo iterativo mejora la estimación de la imagen final usando la siguiente ecuación de actualización:

$$\hat{x} \leftarrow \hat{x} \odot \left(A^T (y \oslash A\hat{x}) \right) \quad (2.38)$$

donde y es la imagen borrosa observada, A una matriz que depende del desenfoque no uniforme, \odot representa el producto elemento a elemento y \oslash la división elemento a elemento.

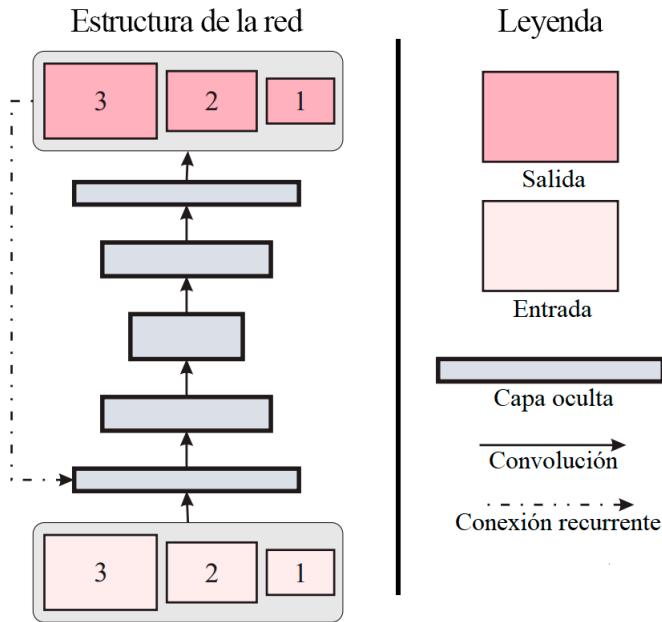


Figura 2.3: Estructura de la red propuesta por Tao et al. [3].³

2.5. Scale-recurrent Network for Deep Image Deblurring

En esta aproximación, los autores del método [3] trabajan con un red neuronal que presenta una estructura recurrente con varias escalas. En cada escala se genera una imagen latente nítida como resultado del subproblema de la estimación del emborronamiento dada una imagen borrosa. Como la red es recurrente, las características que se extraen en las capas ocultas se pueden volver a usar al volver a estimar en otra escala y así conseguir un mejor resultado. En la subsección 2.5.1 podemos ver más detalladamente cómo funciona la red.

En el artículo se prueba que aplicando una estructura codificadora-decodificadora ya existente no se pueden producir resultados óptimos. Es por esto que los autores amplifican el rendimiento de la red durante el entrenamiento usando las mejores estructuras de otras ya existentes. Lo que produce una estructura más consistente a la hora de tratar con imágenes que presentan mucho movimiento.

En la figura 2.3 podemos ver cómo se estructura la red.

³Imagen obtenida del artículo [3] que representa la estructura de la red SRN traducida al castellano.

2.5.1. Red recurrente

La estimación de la imagen latente en cada escala se realiza de la siguiente forma

$$\mathbf{X}^i, \mathbf{h}^i = \text{Net}_{SR}(\mathbf{Y}^i, \mathbf{X}^{i+1\uparrow}, \mathbf{h}^{i+1\uparrow}, \theta_{SR}) \quad (2.39)$$

donde i es el índice de la escala, con $i = 1$ representando la mejor escala. \mathbf{h}^i , \mathbf{X}^i e \mathbf{Y}^i son el estado oculto que puede contener información sobre los patrones de movimiento, la imagen latente estimada y la imagen borrosa en la escala denotada por i , respectivamente. Net_{SR} es la red propuesta por los autores con los parámetros de entrenamiento denotados por θ_{SR} . $(\cdot)^\uparrow$ representa al operador para adaptar las características o imágenes de la escala $(i + 1)$ a la escala i .

La ecuación 2.39 da información detallada de la red, pero en la práctica, hay mucha flexibilidad a la hora de diseñarla. Los autores optan por usar una red convolucional **Istm** (*long-short term memory*), dado que ofrecía un mejor resultado en los experimentos. Algunas posibilidades para el operador $(\cdot)^\uparrow$ incluyen una capa de deconvolución, una capa de convolución sub-pixel y el reescalado de la imagen, para el cual se usa interpolación bilineal debido a su simplicidad y suficiencia. Para obtener un resultado óptimo, la red tiene que estar bien diseñada para recuperar una imagen nítida en cada escala. Podemos ver los detalles del método a continuación.

2.5.2. Codificador/decodificador con ResBlocks

Red codificadora/decodificadora

Una red codificadora-decodificadora suele consistir en una conjunto de redes convolucionales simétricas que primero transforman progresivamente los datos de entrada en mapas de características con tamaños espaciales más pequeños y más canales (en el codificador), y luego los transforman de nuevo a la forma que tenían como en la entrada (en el decodificador). Al igual que en métodos anteriores, se omiten conexiones entre mapas de características se usa para combinar distintos niveles de información. Esto beneficia la propagación del gradiente y acelerar la convergencia debido a que conseguimos una transición más gradual. Normalmente, el codificador cuenta con varias etapas con capas convolucionales que tienen varios pasos, y el decodificador se implementa usando un conjunto de capas de deconvolución o de redimensionado. Después de cada nivel se insertan más capas convolucionales para aumentar aún más la profundidad. A pesar de que la estructura haya demostrado ser efectiva, usar directamente la red no es la mejor opción.

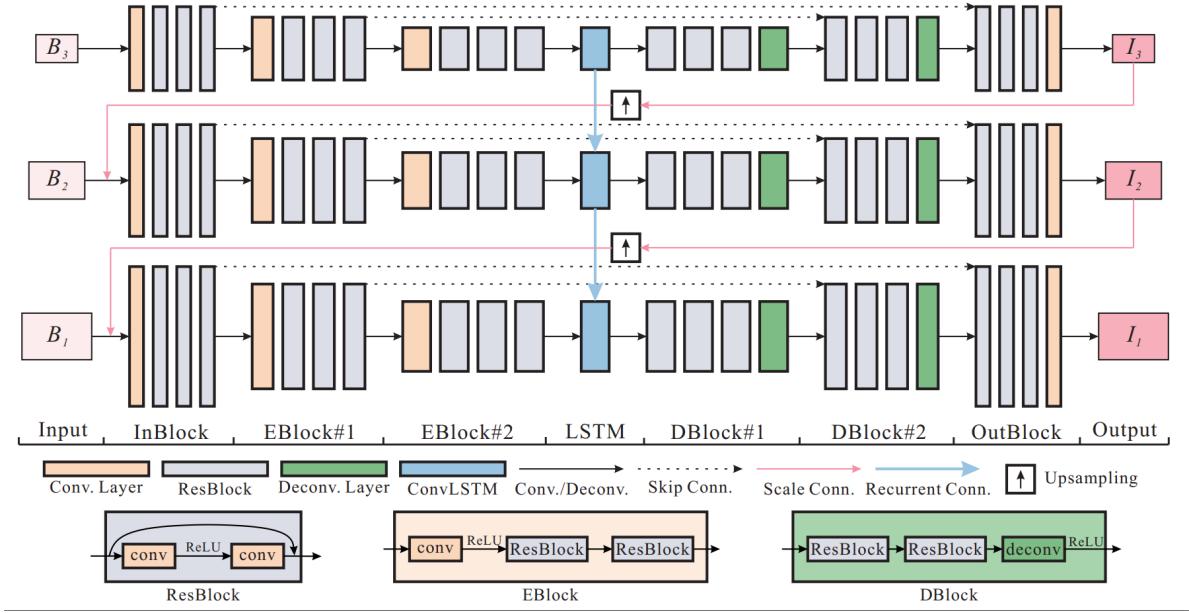


Figura 2.4: Estructura detallada de la red con las distintas etapas y capas. ⁴

Primero, para la tarea de estimación del movimiento, el campo receptivo debe ser grande para poder tratar con imágenes que presenten un movimiento severo, lo que resulta en tener que apilar más niveles en los módulos codificadores/decodificadores. Sin embargo, esta estrategia no se recomienda en la práctica ya que aumenta el número de parámetros rápidamente en las primeras escalas debido con el gran número de canales de características intermedios. Además, el tamaño espacial del mapa de características medio sería demasiado pequeño como para mantener la información espacial para la reconstrucción. En segundo lugar, agregar más capas de convolución en cada nivel de los módulos codificadores/decodificadores haría que la red convergiera lentamente. Es por todo esto que la estructura propuesta usa módulos recurrentes con estados ocultos en su interior.

Red codificadora/decodificadora con omisión de conexiones (Res-Block)

Se realizan varias modificaciones para adaptar las redes codificadoras-decodificadoras al método. Primero, se mejoran los módulos codificadores/decodificadores mediante la introducción de bloques de aprendizaje residual [35]. Según los resultados de [36] y los experimentos realizados por los propios autores, eligen usar ResBlocks en lugar del original en ResNet [35] (sin

⁴Imagen obtenida del artículo [3] que representa la estructura detallada de la red SRN y las conexiones entre las distintas escalas.

normalización por lotes). Como se ilustra en la figura 2.4, el codificador ResBlocks (**EBlocks**) propuesto cuenta con una capa de convolución seguida de varios ResBlocks. En la segunda capa de convolución se realiza un *stride* de 2 píxeles. Se duplica el número de núcleos de la capa anterior y se reducen los mapas de características a la mitad de tamaño. Cada uno de los siguientes ResBlocks contiene 2 capas de convolución. Además, todas las capas de convolución tienen el mismo número de núcleos. El decodificador ResBlock (**DBlocks**) es simétrico a **EBlock**, contiene varios ResBlocks seguidos de una capa de deconvolución. La capa de deconvolución se utiliza para duplicar el tamaño espacial de los mapas de características y reducir a la mitad los canales. En segundo lugar, la estructura de escala recurrente requiere módulos recurrentes dentro de las redes. Siguiendo la estrategia de [37] se insertan capas de convolución la última capa oculta para que así conecte escalas consecutivas. Finalmente, se utilizan núcleos de convolución grandes con tamaño 5×5 para cada capa de convolución. Se puede expresar la red modificada de la siguiente forma

$$\begin{aligned} \mathbf{f}^i &= \mathbf{Net}_E(\mathbf{Y}^i, \mathbf{X}^{i+1\uparrow}; \theta_E) \\ \mathbf{h}^i, \mathbf{y}^i &= \mathbf{ConvLSTM}(\mathbf{h}^{i+1\uparrow}, \mathbf{f}^i; \theta_{LSTM}) \\ \mathbf{X}^i &= \mathbf{Net}_D(\mathbf{y}^i; \theta_D) \end{aligned} \quad (2.40)$$

donde \mathbf{Net}_E y \mathbf{Net}_D son las redes codificadoras y decodificadoras con los parámetros θ_E y θ_D , respectivamente. Se usan 3 etapas de EBlocks y DBlocks en \mathbf{Net}_E y \mathbf{Net}_D . θ_{LSTM} son los parámetros de la red convolucional LSTM. h^i es el estado oculto que puede contener información útil sobre el resultado parcial y sobre los patrones de movimiento, por lo que se pasa a la siguiente escala para mejorar la solución al problema.

La red de los autores contiene 3 escalas. La escala $(i + 1)$ es de la mitad del tamaño de la escala $i - \text{ésima}$. Para la red ResBlock del codificador/-decodificador, hay 1 InBlock, 2 EBlocks, seguido de 1 bloque convolucional LSTM, 2 DBlocks y 1 OutBlock, como se muestra en la figura 2.4. InBlock produce un mapa de características de 32 canales. OutBlock toma el mapa de características anterior como entrada y genera una imagen de salida. El número de núcleos de todas las capas de convolución dentro de cada EBlock/DBlock es el mismo. Para EBlocks, el número de núcleos es 64 y 128, respectivamente. Para DBlocks, son 128 y 64. El tamaño de paso para la capa de convolución en EBlocks y capas de deconvolución es 2, mientras que el resto son 1. Las unidades lineales rectificadas (ReLU) se utilizan como la función de activación para todas las capas, y todos los tamaños del núcleo se establecen en 5.

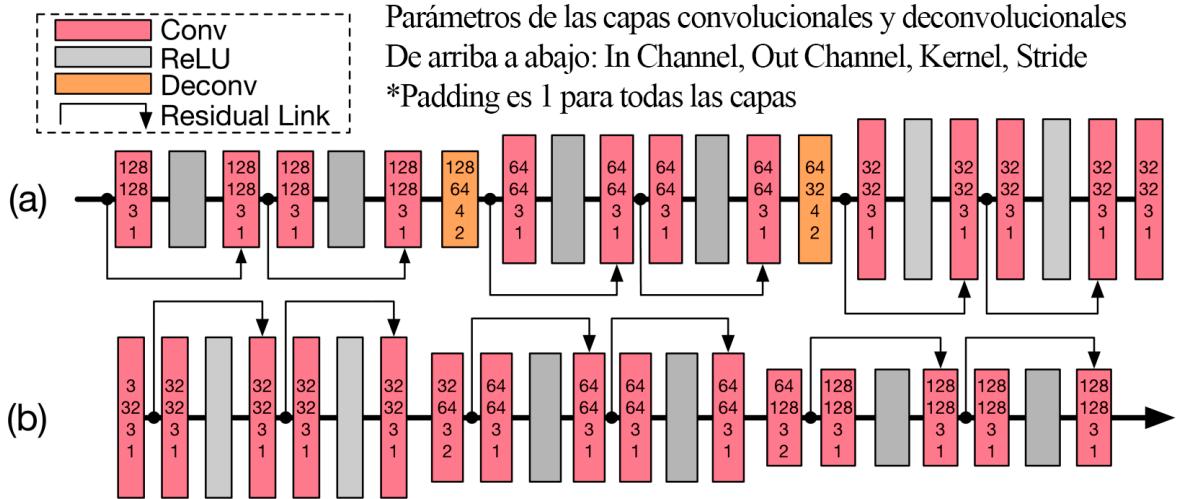


Figura 2.5: Estructura de la red codificadora-decodificadora propuesta por [4] donde (a) representa el decodificador y (b) el codificador. ⁵

2.6. Deep Stacked Hierarchical Multi-Patch Network for Image Deblurring

Para este método los autores de [4] se inspiran en una pirámide multiescala que se ha usado para reconocimiento de escenas para agregar parches que mejoran la imagen y proporcionan un mejor resultado. En comparación a otros modelos de inferencia a escala múltiple y recurrente, el método propuesto usa una arquitectura similar a la residual, por lo que hace uso de filtros más pequeños, lo que resulta en un procesamiento más rápido. A pesar de que este modelo utiliza una arquitectura muy simple (se han eliminado las conexiones omitidas y recurrentes), es muy eficaz. Se utilizan operaciones como las concatenaciones de mapas de características, que son posibles debido a la configuración de parches múltiples que se proponen.

2.6.1. Arquitectura codificadora/decodificadora

Cada nivel de la red DMPHN consta de un codificador y un decodificador cuya arquitectura se ilustra en 2.5. El codificador consta de 15 capas convolucionales, 6 enlaces residuales y 6 unidades ReLU. Las capas de decodificador

⁵Imagen obtenida del artículo [4] donde los autores presentan la estructura de la red codificadora/decodificadora traducida al castellano.

y codificador son idénticas excepto que dos capas convolucionales son reemplazadas por capas deconvolucionales para generar imágenes. Los parámetros del codificador y decodificador tan sólo tienen un tamaño de 3.6 MB debido a la naturaleza residual del modelo, lo que contribuye significativamente al rápido tiempo de ejecución de la estimación de la imagen final.

2.6.2. Arquitectura de la red

La red DMPHN usa el modelo 1-2-4-8, donde la notación 1-2-4-8 indica el número de parches desde el nivel más bajo hasta el más alto. Se denota la entrada de la imagen borrosa inicial como \mathbf{Y}_1 , mientras que \mathbf{Y}_{ij} denota el parche j -ésimo en el nivel i -ésimo. Además, \mathcal{E}_i y \mathcal{D}_i son el codificador y el decodificador en el nivel i -ésimo, \mathbf{C}_{ij} es la salida de \mathcal{D}_i para una imagen \mathbf{Y}_{ij} , y \mathbf{X}_{ij} representa los parches de salida de \mathcal{D}_i .

Cada nivel de la red consiste en un par codificador/decodificador. La entrada para cada nivel se genera dividiendo la imagen original \mathbf{Y}_1 entre varios parches no superpuestos. La salida del codificador y del decodificador en cada nivel se añade al nivel de arriba, de forma que conforme ascendemos por los niveles de la pirámide se infiere la información de las capas de los niveles inferiores. Hay que tener en cuenta que el número de parches de entrada y salida en cada nivel son diferentes, esto se debe a que los autores consiguen con este método que los niveles inferiores se centren en la información local con el objetivo de producir información residual que será usada en los niveles superiores.

El proceso de estimación del emborronamiento comienza en la base de la pirámide, en el nivel 4. En este nivel la imagen \mathbf{Y}_1 se divide en 8 parches no superpuestos $\mathbf{Y}_{4,j}$, con $j = 1, \dots, 8$, que tras introducir en el codificador \mathcal{E}_4 produce la siguiente representación de características:

$$\mathbf{C}_{4,j} = \mathcal{E}_4(\mathbf{Y}_{4,j}), \quad j \in \{1 \dots 8\} \quad (2.41)$$

después se concatenan las características adyacentes para obtener una nueva representación $\mathbf{C}_{4,j}^*$ que es del mismo tamaño que la representación en el nivel 3:

$$\mathbf{C}_{4,j}^* = \mathbf{C}_{4,2j-1} \oplus \mathbf{C}_{4,2j}, \quad j \in \{1 \dots 4\} \quad (2.42)$$

donde \oplus denota el operador de concatenación. Después, esta nueva representación se pasa por el decodificador para obtener $\mathbf{X}_{4,j} = \mathcal{D}_4(\mathbf{C}_{4,j}^*)$. Una vez tenemos la salida del decodificador subimos al nivel 3, donde la salida del codificador \mathcal{E}_3 se obtiene combinando $\mathbf{X}_{4,j}$ con los parches $\mathbf{Y}_{3,j}$. Ya teniendo

la salida del decodificador \mathcal{D}_3 se le añade la representación $\mathbf{C}_{4,j}^*$:

$$\mathbf{C}_{3,j} = \mathcal{E}_3(\mathbf{Y}_{3,j} + \mathbf{X}_{4,j}) + \mathbf{C}_{4,j}^*, \quad j \in \{1\dots4\} \quad (2.43)$$

Los autores siguen el mismo procedimiento en los distintos niveles (incluyendo algunas variaciones que aparecen explicadas en la sección 3.2 de [4]) hasta que en el nivel 1 se obtiene la imagen final X_1 nítida. En este último nivel la imagen final se obtiene siguiendo

$$\mathbf{C}_1 = \mathcal{E}_1(\mathbf{Y}_1 + \mathbf{X}_2) + \mathbf{C}_2^*, \quad \mathbf{X}_1 = \mathcal{D}_1(\mathbf{C}_1) \quad (2.44)$$

Al contrario que en otras aproximaciones, los autores optan por evaluar la función de pérdida MSE (*mean squared error*) tan sólo con la salida del nivel 1, con la imagen final. Para ello, la función de pérdida que usan es la siguiente:

$$\mathcal{L} = \frac{1}{2} \|\mathbf{X}_1 - \mathbf{X}'\|_F^2 \quad (2.45)$$

donde \mathbf{X}' denota la imagen nítida original. Debido a la arquitectura jerárquica, la red sigue el principio de aprendizaje residual en el que se recogen estadísticas de la imagen en cada escala de la pirámide. Por tanto, sólo se evalúa la pérdida una vez se tiene la imagen final. Los autores comentan que tras haber investigado aproximaciones en las que usan una función de pérdida en las diferentes escalas no obtienen ninguna mejora visible.

2.6.3. Red apilada de parches múltiples

Los autores proponen un nuevo paradigma para la estimación del embotamiento, en el que en lugar de hacer la red más profunda verticalmente proponen incrementar la profundidad de forma horizontal. En la sección 3.3 de [4] proponen varias formas de conectar la red en cascada. El objetivo principal para todas las propuestas es minimizar la función dada por

$$\mathcal{L} = \frac{1}{2} \sum_{i=1}^N \|\mathbf{X}_i - \mathbf{X}'\|_F^2 \quad (2.46)$$

donde N es el número de submodelos usados, \mathbf{X}_i es la salida del submodelo i -ésimo y \mathbf{X}' es la imagen nítida original.

Los experimentos realizados por los autores prueban que el uso de esas redes apiladas mejoran el rendimiento de la estimación del embotamiento. También comentan que a pesar de haber usado unidades DMPHN, su aproximación puede complementarse con otros métodos de aprendizaje profundo

con el objetivo de mejorar su rendimiento. Optan por usar sus unidades DMPHN debido a su eficacia y eficiencia, lo que hace que en conjunto sea un método capaz de trabajar con un conjunto de datos pesado en un tiempo razonable.

2.7. Efficient And Interpretable Deep Blind Image Deblurring Via Algorithm Unrolling

Los autores del artículo [38] proponen una estructura basada en DL que cuenta con una red neuronal interpretable que denominan DUBLID (*Deep Unrolling for Blind Deblurring*). En este artículo, los autores se centran principalmente en resolver el problema del desenfoque de movimiento, al igual que hemos visto en otras aproximaciones anteriores, aunque aseguran que su propuesta se puede extender a distintos tipos de emborronamiento. Para resolver el problema de la estimación del emborronamiento desarrollan la ecuación 1.1 y proponen un método basado en redes neuronales más rápido que los tradicionales. Presentan un algoritmo iterativo que trabaja usando un método de regularización en los gradientes de la imagen, después usan dicho algoritmo para construir la red. Los parámetros de penalización que usan para el algoritmo se consiguen mediante el uso de retro-propagación durante el entrenamiento. Los autores aseguran que la principal diferencia de su método respecto de otros que también se basan en DL se debe a que se trata de un modelo muy simple en el que el diseño interno de la red se ha diseñado cuidadosamente, lo que resulta en un modelo más rápido que necesita menos parámetros y que proporciona mejores resultados que otros métodos en el estado del arte de la deconvolución ciega de imágenes.

2.7.1. Estimación del emborronamiento en el dominio de los filtros

Una práctica común para la estimación del emborronamiento ha sido tratar de estimar el núcleo en el dominio de los gradientes de la imagen. El principal problema al resolver el problema usando esta aproximación se da a la hora de recuperar la imagen, pues el núcleo generalmente actúa como un filtro de paso bajo, haciendo que perdamos muchos detalles de la imagen a pesar de haber estimado correctamente el emborronamiento. El uso de este modelo para estimar el emborronamiento y la imagen final es insuficiente, ya que necesitamos regularizadores para ambas estimaciones. Para regularizar el núcleo, como hemos visto en la introducción, se suele asumir que sus coeficientes no son negativos y que en conjunto suman uno. Para solucionar este problema, los autores formulan el siguiente problema de optimización

$$\min_{\mathbf{h}, \{\mathbf{g}_i\}_{i=1}^C} \sum_{i=1}^C \left(\frac{1}{2} \|\mathbf{f}_i * \mathbf{y} - \mathbf{h} * \mathbf{g}_i\|_2^2 + \lambda_i \|\mathbf{g}_i\|_1 \right) + \frac{\epsilon}{2} \|\mathbf{h}\|_2^2 \quad (2.47)$$

sujeto a $\|\mathbf{h}\|_1 = 1$, $\mathbf{h} \geq 0$. Donde C es una colección de filtros $\{\mathbf{f}_i\}_{i=1}^C$, \mathbf{g}_i

es una estimación del gradiente de la imagen nítida \mathbf{x} , \mathbf{y} es la imagen borrosa y \mathbf{h} es el núcleo.

2.7.2. Minimización mediante HQS (*Half-quadratic Splitting*)

Para resolver 2.47 necesitan un algoritmo simple y rápido que converja en pocos pasos para así reducir el número de capas necesarias en la red. La idea principal es usar HQS para estimar las variables y alternar la minimización de la función de pérdida. Proponen el siguiente modelo

$$\min_{\mathbf{h}, \mathbf{g}_1, \mathbf{g}_2} \frac{1}{2} \left(\|D_x \mathbf{y} - \mathbf{h} * \mathbf{g}_1\|_2^2 + \|D_y \mathbf{y} - \mathbf{h} * \mathbf{g}_2\|_2^2 \right) + \lambda_1 \|\mathbf{g}_1\|_1 + \lambda_2 \|\mathbf{g}_2\|_1 + \frac{\epsilon}{2} \|\mathbf{h}\|_2^2 \quad (2.48)$$

sujeto a $\|\mathbf{h}\|_1 = 1$, $\mathbf{h} \geq 0$ e introduciendo las variables auxiliares $\{\mathbf{z}_i\}_{i=1}^C$, donde $\zeta_i, i = 1, \dots, C$ son parámetros de regularización. Entonces, resuelven alternativamente $\{\mathbf{g}_i\}_{i=1}^C$, $\{\mathbf{z}_i\}_{i=1}^C$ y \mathbf{h} e iteran hasta que converja minimizando cada ciertas iteraciones. Para realizar la estimación de los mapas de características hacen uso de la transformada discreta de Fourier (*Discrete Fourier Transform*, DFT). La estimación del núcleo es un problema cuadrático con restricción simple que se resuelve mediante la iteración de algoritmos numéricos. Para la estimación del emborronamiento proponen el algoritmo 4.

Algorithm 4 Algoritmo de división media cuadrática para la deconvolución ciega con continuación.

Require: Imagen borrosa y , bancos de filtros $\{\mathbf{f}_i^l\}_{i,l}$, parámetros constantes positivos $\{\zeta_i^l, \lambda_i^l\}_{i,l}$, número de iteraciones L y ε .

Output: núcleo estimado \tilde{h} , mapas de características estimados $\{\widetilde{fg}_i\}_{i=1}^C$.

- 1: Inicializar $fh^1 \leftarrow \delta; fz_i^1 \leftarrow 0, i = 1, \dots, C$.
 - 2: **for** $l = 1$ **to** L **do**
 - 3: **for** $i = 1$ **to** C **do**
 - 4: $y_i^l \leftarrow \mathbf{f}_i^l * y,$
 - 5: $g_i^{l+1} \leftarrow \mathcal{F}^{-1} \left\{ \frac{\zeta_i^l \widehat{h}^l * \widehat{y}_i^l + \widehat{z}_i^l}{\zeta_i^l |\widehat{h}^l|^2 + 1} \right\},$
 - 6: $z_i^{l+1} \leftarrow \mathcal{S}_{\lambda_i^l \zeta_i^l} \{g_i^{l+1}\},$
 - 7: **end for**
 - 8: $h^{l+\frac{1}{3}} \leftarrow \mathcal{F}^{-1} \left\{ \frac{\sum_{i=1}^C \widehat{z}_i^{l+1} * \widehat{y}_i^l}{\sum_{i=1}^C |\widehat{z}_i^{l+1}|^2 + \epsilon} \right\},$
 - 9: $h^{l+\frac{2}{3}} \leftarrow \left[h^{l+\frac{1}{3}} - \beta^l \log \left(\sum_i \exp \left(h_i^{l+\frac{1}{3}} \right) \right) \right]_+,$
 - 10: $h^{l+1} \leftarrow \frac{h^{l+\frac{2}{3}}}{\|h^{l+\frac{2}{3}}\|_1},$
 - 11: $l \leftarrow l + 1.$
 - 12: **end for**
-

Después de que el algoritmo 4 converja, se obtiene la estimación de los mapas de características $\{\tilde{g}_i\}_i$ y del núcleo \tilde{h} . Como hemos comentado en la introducción, debido a la naturaleza de filtro de paso bajo de \tilde{h} , usarlo para recuperar la imagen final no es adecuado. Por tanto, recuperan \tilde{x} resolviendo el siguiente problema de optimización

$$\tilde{x} \leftarrow \arg \min_x \frac{1}{2} \|\mathbf{y} - \tilde{h} * x\|_2^2 + \sum_{i=1}^C \frac{\eta_i}{2} \|\mathbf{f}_i^L * x - \tilde{g}_i\|_2^2 = \mathcal{F}^{-1} \left\{ \frac{\widehat{\tilde{h}}^* \odot \widehat{y} + \sum_{i=1}^C \eta_i \widehat{\mathbf{f}_i^L}^* \odot \widehat{\tilde{g}}_i}{\|\widehat{\tilde{h}}\|^2 + \sum_{i=1}^C \eta_i \|\widehat{\mathbf{f}_i^L}\|^2} \right\} \quad (2.49)$$

donde η_i son parámetros de regularización positivos.

2.7.3. Construcción de la red

Cada paso del algoritmo 4 se encuentra en forma analítica y se puede implementar usando una serie de operaciones funcionales básicas. En concreto, para los pasos 5 y 8 podemos ver su implementación en 2.6. La operación del paso 6 se puede resolver usando dos operaciones ReLU dado $S_\lambda(x) = [x - \lambda]_+ - [-x - \lambda]_+$. De igual modo, 2.49 se puede implementar de acuerdo al subgráfico (c) en 2.6. Entonces, el algoritmo 4 admite una representación en forma de diagrama, y repitiéndola L veces acabamos con una red neuronal de L capas.

Cuando el núcleo tiene un tamaño grande (cosa que puede ocurrir en imágenes que presenten el emborronamiento debido a un movimiento rápido), se debería cambiar el tamaño espacial de los bancos de filtros $\{\mathbf{f}_i\}_i$ en las diferentes capas. Como hemos podido ver en otros trabajos, en la restauración de imágenes con emborronamiento, la estimación del núcleo se realiza frecuentemente por etapas. Tratando con la imagen con poca resolución y aumentando su tamaño al final de cada etapa. Esto se puede integrar en el esquema del algoritmo 4 eligiendo filtros más grandes en las primeras iteraciones, de modo que sean capaces de capturar las características más importantes de la imagen, y disminuyendo gradualmente su tamaño para permitir obtener más detalles. Traduciendo esto a la red, podemos esperar la siguiente relación entre los tamaños del núcleo en diferentes capas:

$$\text{tamaño de } \mathbf{f}_i^1 \geq \text{tamaño de } \mathbf{f}_i^2 \geq \text{tamaño de } \mathbf{f}_i^3 \geq \dots$$

En la práctica, los filtros grandes pueden ser difíciles de entrenar debido a que tienen una mayor cantidad de parámetros. Para solucionar este problema, se crean filtros grandes conectando en cascada filtros de tamaño 3×3 . Formalmente, se establece $\mathbf{f}_i^L = w_{i1}^L$ donde $\{w_{i1}^l\}_{i=1}^C$ es una colección de filtros 3×3 y recursivamente se obtiene \mathbf{f}_i^l filtrando \mathbf{f}_i^{l+1} a filtros 3×3 con $\{w_{ij}^l\}_{i,j=1}^C$:

$$\mathbf{f}_i^l \leftarrow \sum_{j=1}^C w_{ij}^l * \mathbf{f}_j^{l+1}, \quad i = 1, 2, \dots, C.$$

Integrando lo anterior en la red, obtienen la estructura representada en la figura 2.7. Hay que tener en cuenta que \mathbf{y}^l se puede obtener ahora más

⁶Bloques de diagramas obtenidos del artículo [38].

⁷En esta figura se usa una notación distinta a la que se ha usado en las distintas ecuaciones. k denota el núcleo de emborronamiento.

⁸Imagen obtenida del artículo [38] en la que se presenta la estructura de la red DUBLID.

⁹En esta figura se usa una notación distinta a la que se ha usado en las distintas ecuaciones. S denota la imagen latente estimada.

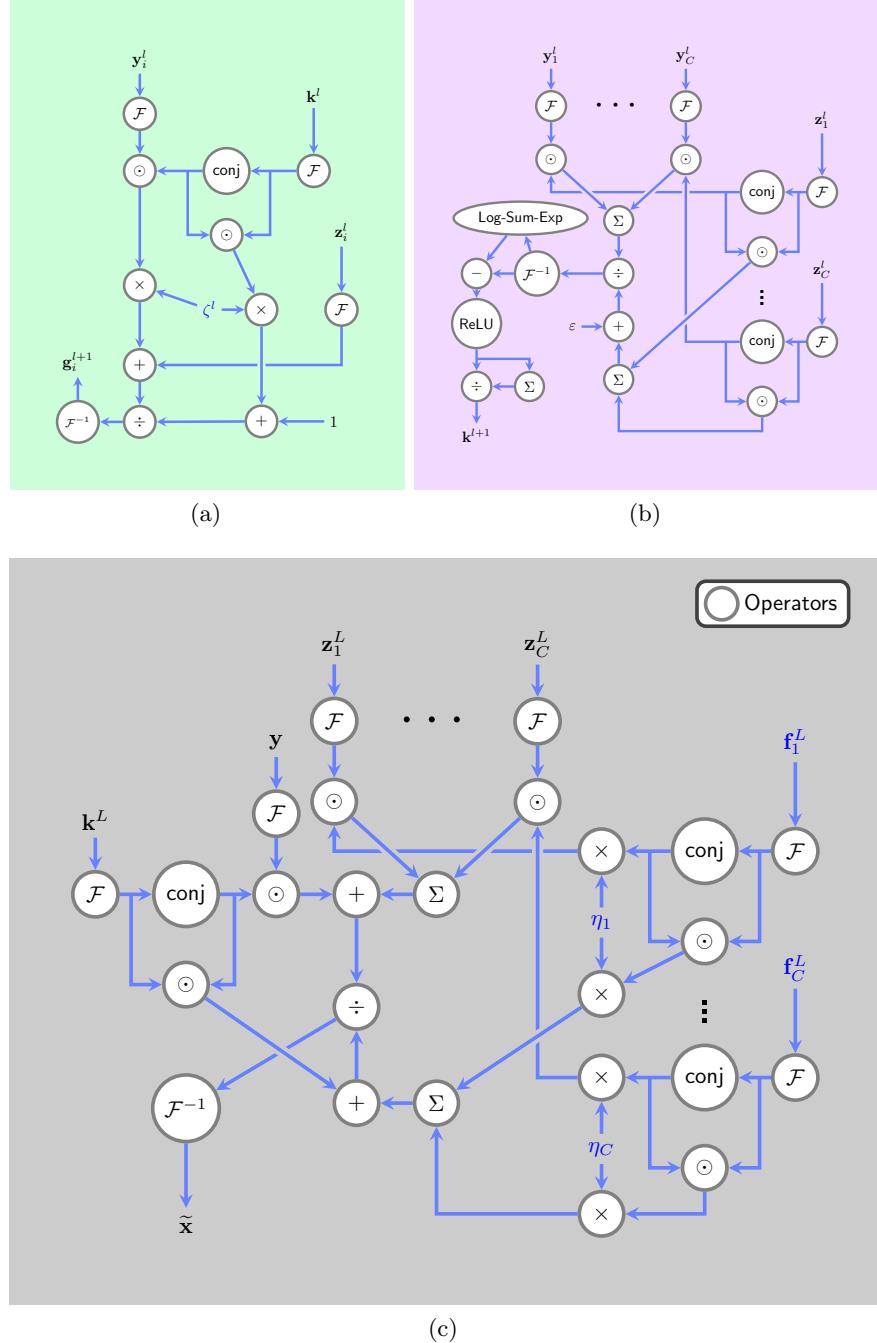


Figura 2.6: Bloque de diagramas representando (a) el paso 5 en algoritmo 4, (b) paso 8 en algoritmo 4 y (c) ecuación 2.49. Representando el algoritmo como una red multicapa, los diagramas se usan como bloques de construcción que se repiten en las capas. Los parámetros (ζ y η) se aprenden de conjuntos de datos reales y se colorean en azul.⁶⁷

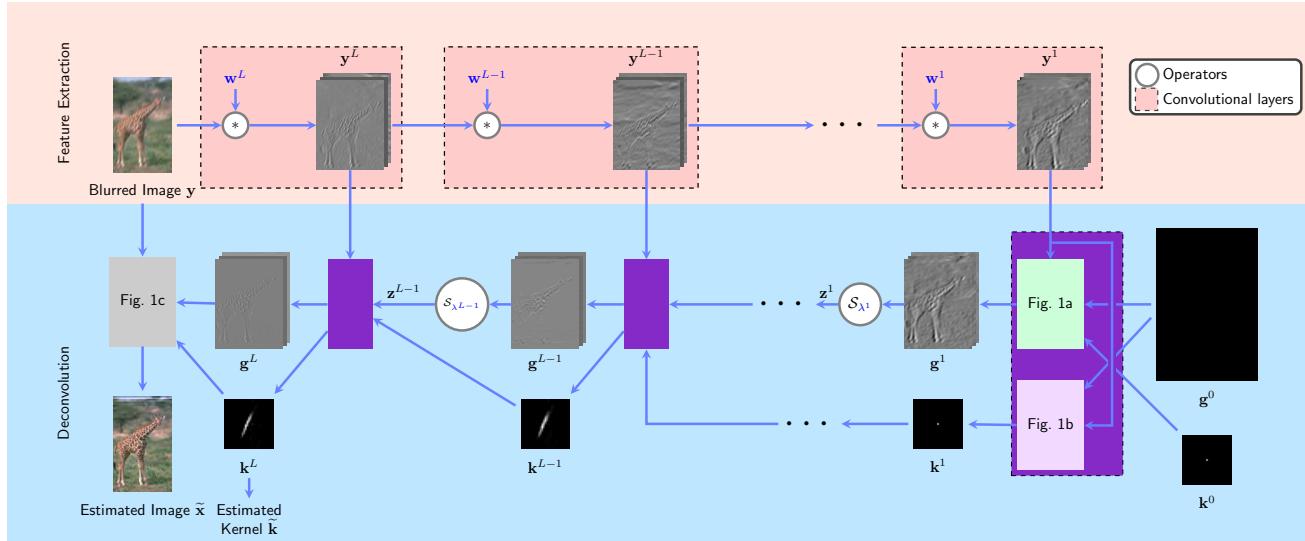


Figura 2.7: DUBLID usando en cascada filtros de tamaño 3×3 en lugar de filtros más grandes reduce el número de parámetros y la red puede ser más fácil de entrenar. También se muestran las capas ocultas. Se puede observar que conforme l incrementa, g^l e y^l tienen un resultado más detallado. Los parámetros que se aprenden de los datasets con imágenes reales están coloreados de azul.⁸⁹

eficientemente filtrando y^{l+1} a través de w^l . También se ha de tener en cuenta que $\{w_{ij}^l\}_{i,j=1}^C$ tiene que aprenderse como marca la figura 2.7.

Capítulo 3

Estudio comparativo de métodos analíticos y de aprendizaje profundo para la deconvolución ciega de imágenes

Como ya hemos indicado, el objetivo de este trabajo es el diseño de un modelo de postprocesamiento de imágenes obtenidas por métodos de deconvolución ciega para mejorarlas mediante un modelo basado en técnicas de DL. Para ello, en este capítulo realizamos una comparativa entre los distintos métodos que hemos visto en el capítulo anterior con el objetivo de escoger aquél que nos ofrezca un mejor resultado y que, en conjunto a nuestra propuesta, consiga una mejora en la salida.

3.1. Conjuntos de datos usados para el estudio comparativo

Para la selección del modelo de deconvolución ciega cuyas imágenes serán mejoradas y la comparación final de todos los métodos, hemos empleado cuatro datasets de imágenes, en los que podemos encontrar tanto las imágenes a restaurar como sus versiones originales, las cuales nos servirán para comparar los resultados usando distintos criterios.

Trabajaremos con diversos tipos de imágenes para asegurar que el método que escogemos es capaz de tratar con un conjunto variado de entradas y proporcionar un buen resultado independientemente de la entrada. En las

42 3.2. Representación de los métodos usados en la comparación

tablas 3.1 y 3.2 podemos ver la información más relevante de cada dataset.

Dataset	Sintético/Real	Tipo de imagen	Métrica de evaluación
Levin et al. 2009 [39]	Sintético	Natural	PSNR
Köhler et al. 2012 [40]	Real	Natural	PSNR
Sun et al. 2013 [41]	Sintético	Natural	PSNR
Nah et al. 2017 [42]	Real	Natural	PSNR

Cuadro 3.1: Información sobre del tipo de imágenes del dataset y la métrica que usaremos para compararlos

Dataset	Imágenes originales	Núcleos de emborronamiento	Imágenes borrosas
Levin et al. 2009 [39]	4	8	32
Köhler et al. 2012 [40]	4	12	48
Sun et al. 2013 [41]	80	8	640
Nah et al. 2017 [42]	1111	1	1111

Cuadro 3.2: Número de imágenes y núcleos de emborronamiento del dataset
¹

3.2. Representación de los métodos usados en la comparación

Antes de ver los resultados para los distintos métodos usaremos una serie de abreviaturas para una mejor distinción a la hora de observar las tablas. Para el método usado en la sección 2.1 usaremos como denotación **Li**, para el método usado en la sección 2.2 usaremos **Zhou**, para el método usado en la sección 2.3 usaremos **Fort**, para el método usado en la sección 2.4 usaremos **DEBLUR**, para el método usado en la sección 2.5 usaremos **SRN**, para el método usado en la sección 2.6 usaremos **DMPHN** y para el último método, que podemos encontrar en la sección 2.7 usaremos **DUBLID**. En las tablas de las próximas secciones se puede observar que en la columna de métodos pueden aparecer dos citas. En el caso de que aparezcan dos citas se ha de tener en cuenta que la primera de ellas indica el método usado para la estimación del núcleo de emborronamiento, y, la segunda, para la recuperación

¹Para el caso del dataset de Nah et al. [42] se indica más detalladamente cómo se generan las imágenes desenfocadas en la sección 3.6

de la imagen final usando el núcleo estimado. Así mismo, reportamos de forma separada los tiempos de estimación del emborronamiento y los de deconvolución. En los métodos que no aparecen los tiempos de estimación ni de deconvolución son aquellos que dada una imagen borrosa como entrada genera directamente la imagen final y tan sólo medimos el tiempo total en dar una salida.

Complementando la comparativa de las distintas métricas también mostramos una serie de imágenes correspondientes a los resultados que obtienen los distintos métodos para una misma imagen. En estas imágenes podremos encontrar en la esquina superior izquierda la estimación del núcleo de emborronamiento en aquellos métodos que lo proporcionen. En el caso de las imágenes que usan una combinación en la que para estimar el emborronamiento se use un método y para recuperar la imagen otro distinto, el núcleo usado corresponde al del método original. En este caso, las combinaciones que usan como estimación los métodos de **Li** [1] y **Zhou** [32], es decir, aquellas con la forma **Li - X** y **Zhou - X**. En la esquina inferior derecha de la imagen podremos ver un pequeño fragmento de la imagen ampliado para apreciar con más detalle el resultado que otorgan los distintos métodos.

3.3. Resultados para el dataset de Levin et al.

Este dataset cuenta con un total de 4 imágenes de tamaño 256x256 píxeles, en escala de grises y con 8 núcleos de emborronamiento generados sintéticamente. Para generar estos núcleos montan la cámara en un trípode, y usando como referencia la imagen original estiman un núcleo de emborronamiento a partir de una imagen que ha sido tomada mientras el trípode se movía durante la exposición. A partir de ese núcleo estimado pueden generar nuevas imágenes con el mismo emborronamiento.

Para este dataset podemos ver en el cuadro 3.4 que el PSNR más alto que obtenemos se corresponde a la combinación **Zhou** [32] - **Fort** [2] en la que usamos la estimación del emborronamiento de Zhou [32] y la deconvolución no ciega del método propuesto por Fortunato en [2]. Sin embargo, comparando el resto de métodos podemos observar que la propuesta **Li** [1] - **Fort** [2] también ofrece un buen resultado en comparación con los otros métodos.

En la figura 3.1 podemos comprobar visualmente que con los métodos de estimación de Li [1] y de Zhou [32] obtenemos un mejor resultado que con el resto de aproximaciones.

²Como hemos comentado en la sección 3.2, podemos ver métodos en los que algunas entradas aparecen vacías. En el caso de los modelos que usan tanto la estimación de Li [1] como la de Zhou [32] toman el valor de la primera aparición, en este caso **Li** [1] - **Li** [1] y **Zhou** [32] - **Li** [1]. Para el resto de métodos tan sólo se mide el tiempo total entre que se introduce la imagen y se genera una salida.

Método	T. estimación	T. deconvolución	T. total
Li [1] - Li [1]	37.8053 s	0.8858 s	38.6911 s
Li [1] - Zhou [32]	-	0.4459 s	38.2512 s
Li [1] - Fort [2]	-	0.0406 s	37.8459 s
Zhou [32] - Li [1]	7.2520 s	0.5576 s	7.8096 s
Zhou [32] - Zhou [32]	-	0.4956 s	7.7476 s
Zhou [32] - Fort [2]	-	0.0408 s	7.2928 s
DMPHN [4]	-	-	0.0139 s
SRN [3]	-	-	0.591 s
DUBLID [38]	-	-	0.0163 s
DEBLUR [43]	-	-	751.1702 s

Cuadro 3.3: Tiempos usando el dataset de Levin et al.²

Método	PSNR	SSIM
Li [1] - Li [1]	27.9298	0.8571
Li [1] - Zhou [32]	27.7415	0.85
Li [1] - Fort [2]	28.3216	0.8619
Zhou [32] - Li [1]	28.9213	0.8978
Zhou [32] - Zhou [32]	29.1484	0.9005
Zhou [32] - Fort [2]	29.6043	0.9017
DMPHN [4]	20.8282	0.6103
SRN [3]	25.3969	0.7873
DUBLID [38]	23.8777	0.7193
DEBLUR [43]	17.5240	0.4233

Cuadro 3.4: PSNR y SSIM del dataset de Levin et al. El mejor resultado aparece resaltado en negrita.

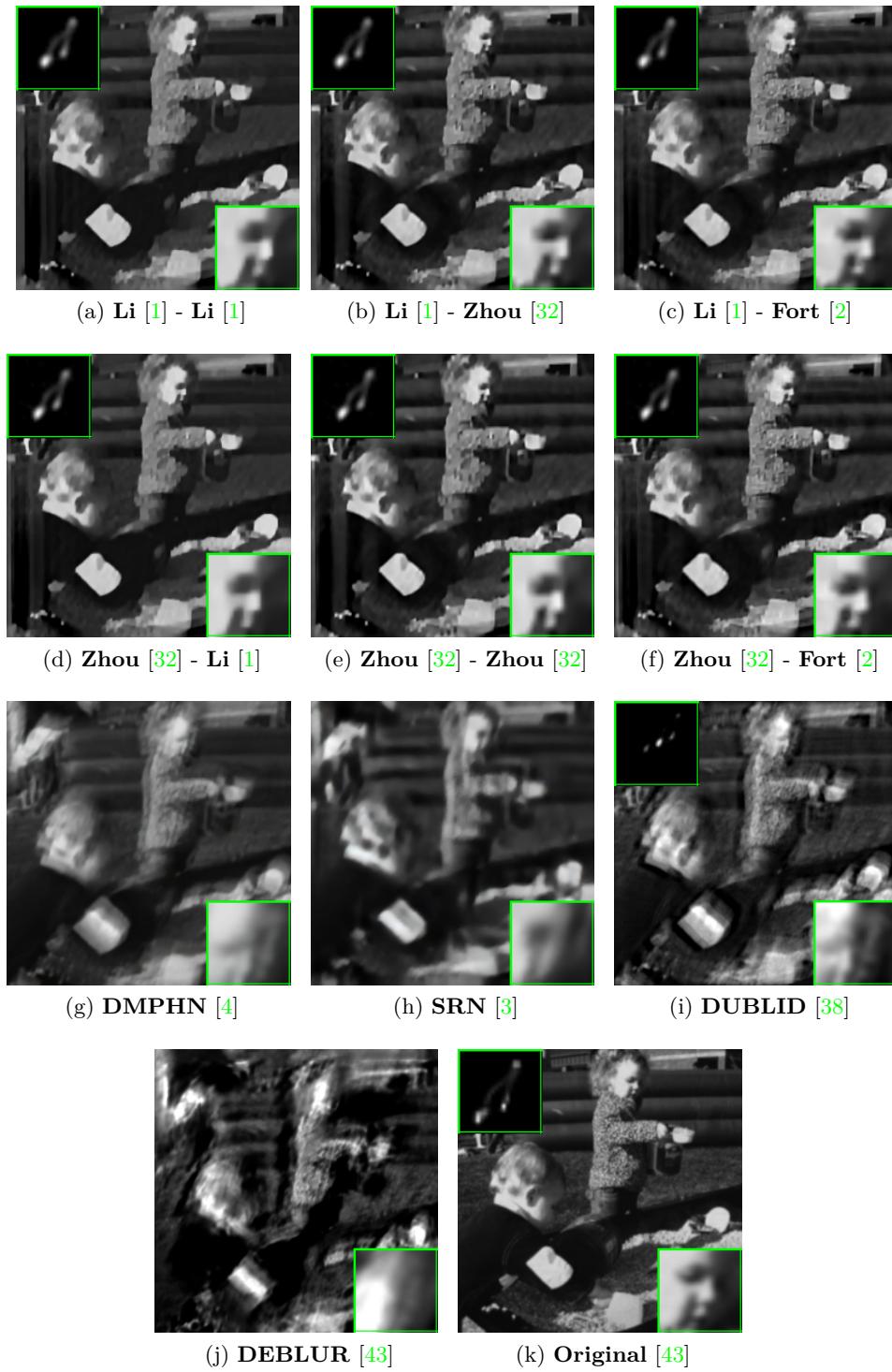


Figura 3.1: Resultados de la deconvolución del dataset de Levin et al.

3.4. Resultados para el dataset de Köhler et al.

Este dataset cuenta con un total de 4 imágenes de tamaño 800x800 píxeles, en color y con 12 núcleos de emborronamiento generados naturalmente. Para generar estos núcleos, las imágenes son tomadas con una cámara unida a un dispositivo de precisión que genera el movimiento durante la exposición de la toma. Al usar este tipo de dispositivo podemos generar el mismo movimiento durante la captura de distintas imágenes lo que equivale a usar un mismo núcleo sintético para generar la versión borrosa de la imagen original.

Método	T. estimación	T. deconvolución	T. total
Li [1] - Li [1]	253.5426 s	25.8774 s	279.42 s
Li [1] - Zhou [32]	-	6.6377 s	260.2203 s
Li [1] - Fort [2]	-	0.9439 s	254.4865 s
Zhou [32] - Li [1]	71.6673 s	25.2174 s	96.8847 s
Zhou [32] - Zhou [32]	-	4.9257 s	76.593 s
Zhou [32] - Fort [2]	-	1.0154 s	72.6827 s
DMPHN [4]	-	-	0.0162 s
SRN [3]	-	-	0.409 s
DUBLID [38]	-	-	0.0184 s
DEBLUR [43]	-	-	191.5712 s

Cuadro 3.5: Tiempos usando el dataset de Köhler et al. ³

En este dataset no contamos con los núcleos de emborronamiento con los que se han generado las imágenes borrosas, por lo que tan sólo podemos comparar los resultados usando nuestras propias métricas y visualizando las imágenes restauradas.

En este caso, a diferencia del anterior, obtenemos un mejor resultado usando la propuesta **Li [1] - Zhou [32]** que usa la estimación del emborronamiento de Li [1] y la deconvolución del método de Zhou [32]. En este ejemplo hemos optado por seleccionar una imagen que presenta un núcleo complejo, lo que nos ayuda a visualizar las diferencias que presenta cada método. Usando este núcleo y esta imagen, obtenemos una mejor estimación del núcleo usando el método de Li [1], aunque, al comparar el cuadro de resultados medios 3.6, el método de Zhou [32] presenta unos resultados bastante similares, y el resto métodos ofrecen unos resultados que no distan mucho en calidad de los

³Como hemos comentado en la sección 3.2, podemos ver métodos en los que algunas entradas aparecen vacías. En el caso de los modelos que usan tanto la estimación de Li [1] como la de Zhou [32] toman el valor de la primera aparición, en este caso **Li [1] - Li [1]** y **Zhou [32] - Li [1]**. Para el resto de métodos tan sólo se mide el tiempo total entre que se introduce la imagen y se genera una salida.

Método	PSNR	SSIM
Li [1] - Li [1]	25.5438	0.8236
Li [1] - Zhou [32]	25.8876	0.8371
Li [1] - Fort [2]	24.9299	0.7927
Zhou [32] - Li [1]	25.1272	0.7885
Zhou [32] - Zhou [32]	25.1419	0.7916
Zhou [32] - Fort [2]	25.0371	0.7902
DMPHN [4]	23.2170	0.7223
SRN [3]	24.3619	0.7603
DUBLID [38]	24.1566	0.7196
DEBLUR [43]	23.2433	0.7243

Cuadro 3.6: PSNR y SSIM del dataset de Köhler et al. El mejor resultado aparece resaltado en negrita.

mejores.

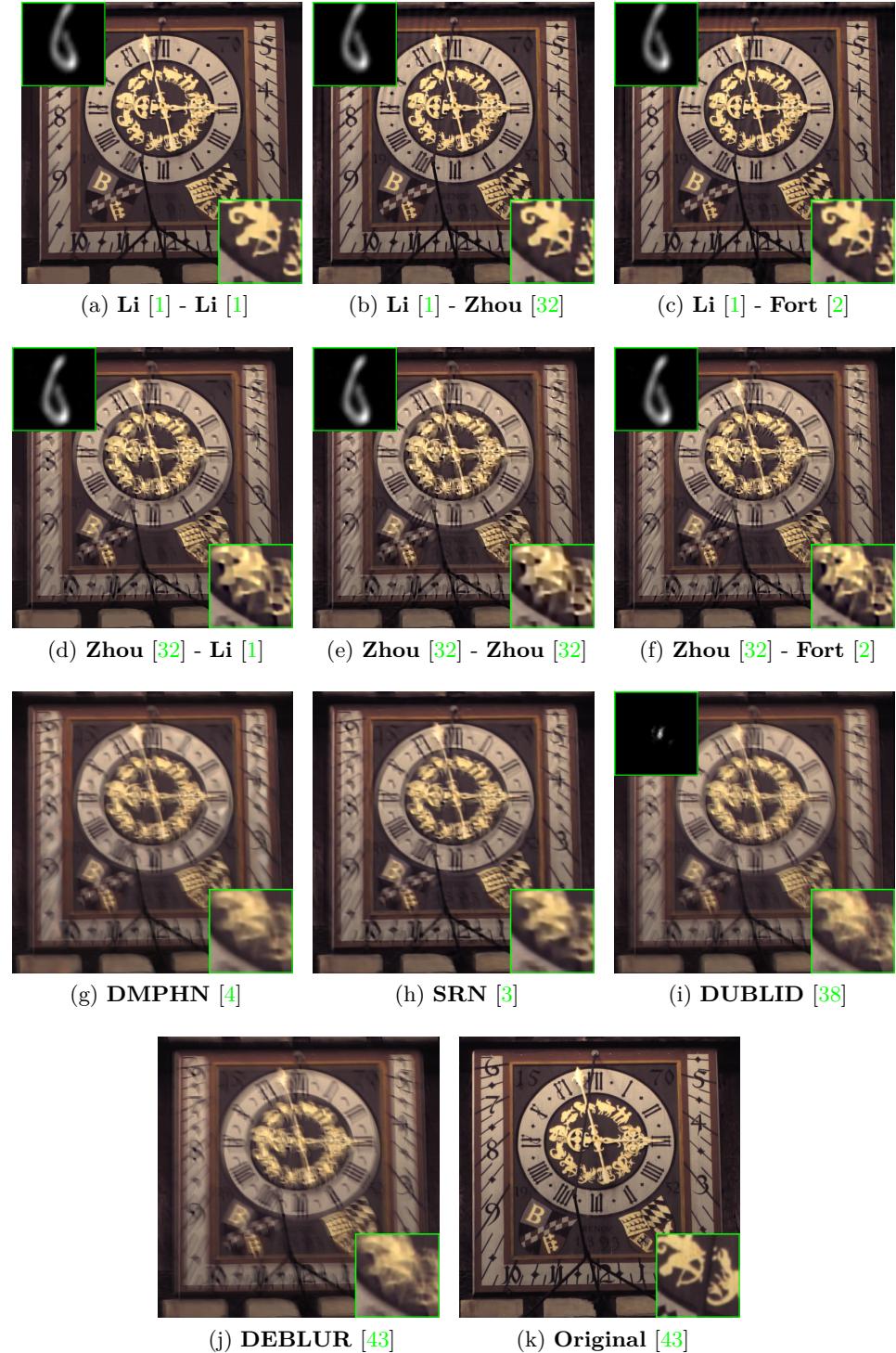


Figura 3.2: Resultados de la deconvolución del dataset de Kohler et al.

3.5. Resultados para el dataset de Sun et al.

Este dataset cuenta con un total de 80 imágenes de tamaño variado, con resoluciones comprendidas entre 1024×426 y 1024×928 , en escala de grises y con 8 núcleos de emborronamiento generados sintéticamente. Dichos núcleos corresponden a los usados en el dataset de Levin et al. [39] que hemos visto en la sección 3.3.

Método	T. estimación	T. deconvolución	T. total
Li [1] - Li [1]	280.5220 s	9.7427 s	290.2647 s
Li [1] - Zhou [32]	-	5.8907 s	286.4127 s
Li [1] - Fort [2]	-	0.5889 s	281.1109 s
Zhou [32] - Li [1]	81.5046 s	10.7616 s	92.2662 s
Zhou [32] - Zhou [32]	-	5.5341 s	87.0387 s
Zhou [32] - Fort [2]	-	0.5376 s	82.0422 s
DMPHN [4]	-	-	0.0154 s
SRN [3]	-	-	0.591 s
DUBLID [38]	-	-	0.02079 s
DEBLUR [43]	-	-	404.5345 s

Cuadro 3.7: Tiempos usando el dataset de Sun et al. ⁴

Método	PSNR	SSIM
Li [1] - Li [1]	27.3926	0.8139
Li [1] - Zhou [32]	27.5414	0.805
Li [1] - Fort [2]	28.5665	0.8222
Zhou [32] - Li [1]	29.0972	0.827
Zhou [32] - Zhou [32]	29.6492	0.8433
Zhou [32] - Fort [2]	29.5543	0.8287
DMPHN [4]	15.6852	0.3787
SRN [3]	23.9644	0.6050
DUBLID [38]	24.0708	0.5251
DEBLUR [43]	15.4006	0.2954

Cuadro 3.8: PSNR y SSIM del dataset de Sun et al. El mejor resultado aparece resaltado en negrita.

⁴Como hemos comentado en la sección 3.2, podemos ver métodos en los que algunas entradas aparecen vacías. En el caso de los modelos que usan tanto la estimación de Li [1] como la de Zhou [32] toman el valor de la primera aparición, en este caso **Li [1] - Li [1]** y **Zhou [32] - Li [1]**. Para el resto de métodos tan sólo se mide el tiempo total entre que se introduce la imagen y se genera una salida.

Al comparar los resultados de este dataset podemos apreciar bastante diferencia entre las propuestas que usan como estimación del emborronamiento los métodos de Li [1] y Zhou [32] respecto de todos los otros, y, comparando las propuestas que usan estos métodos para estimar el emborronamiento, la propuesta Zhou [32] - Zhou [32] es la que da el mejor resultado. Al igual que en el dataset anterior, hemos seleccionado un núcleo complejo y una imagen en la que se puede apreciar si el método consigue resolver el problema de la estimación del emborronamiento correctamente. En este caso, podemos ver en la figura 3.3 que obtenemos una mejor estimación usando la propuesta Zhou [32] - Zhou [32].

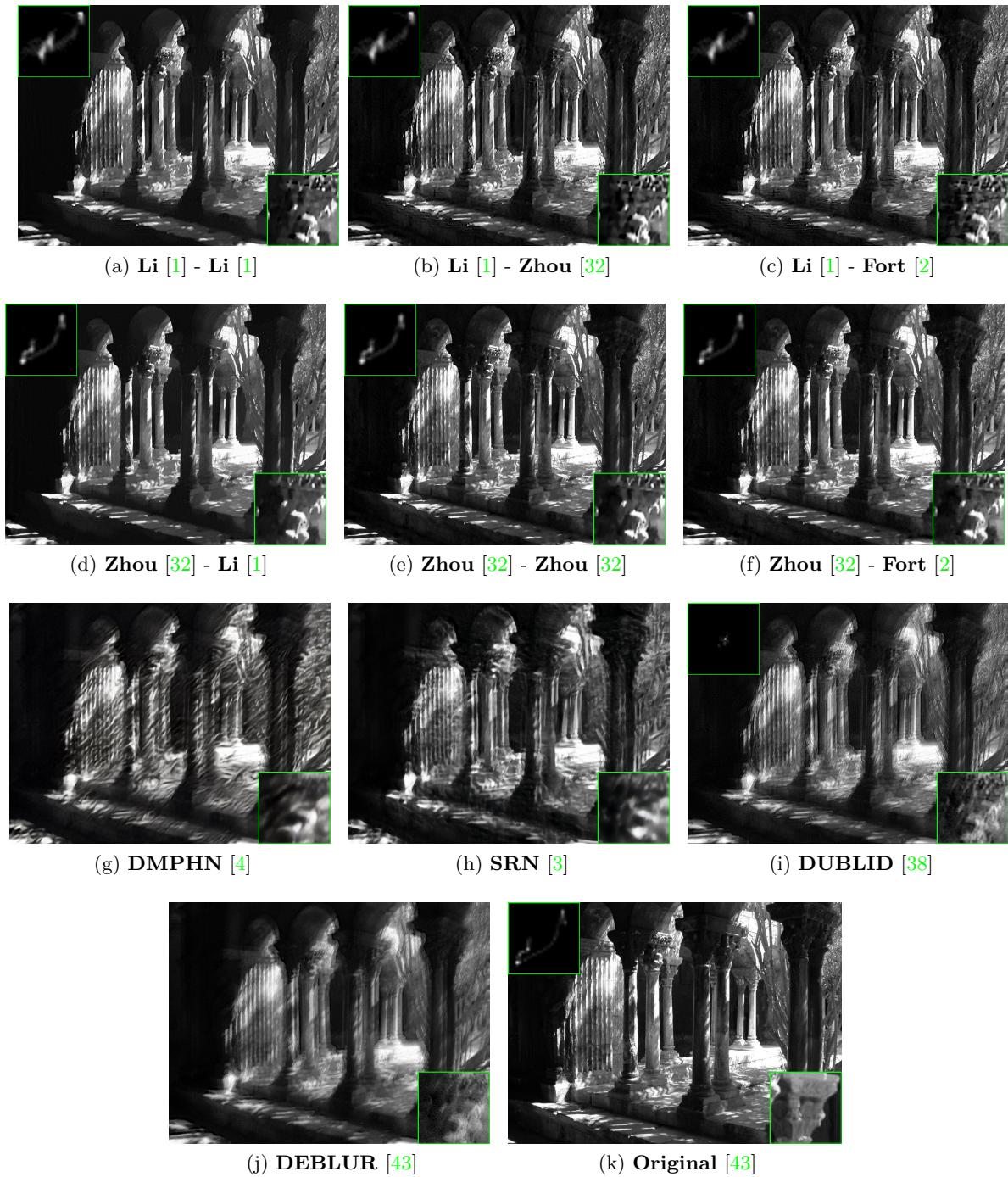


Figura 3.3: Resultados de la deconvolución del dataset de Sun et al.

3.6. Resultados para el dataset de Nah et al.

En este dataset podemos encontrar varios conjuntos distintos de datos. Para la realización de las pruebas usamos el sub-conjunto de imágenes de test de este dataset, el cuál está compuesto por varias imágenes emborronadas con distintas núcleos de emborronamiento. En este dataset contamos con imágenes en color con resolución 1280×720 en la que cada imagen tiene un emborronamiento distinto. Para generar estos núcleos de emborronamiento los autores de [42] parten de una filmación de vídeo nítida y generan la versión borrosa de la imagen mediante la unión de distintos fotogramas. Por ejemplo, una imagen tomada con un tiempo de obturación de $1/16\text{s}$ se podría simular mediante la media de 15 fotogramas tomados del vídeo, mientras que una imagen nítida correspondería a una imagen tomada con una cámara con un tiempo de obturación de $1/240\text{s}$.

Método	T. estimación	T. deconvolución	T. total
Li [1] - Li [1]	10069.1 s	34.2704 s	1103.3704 s
Li [1] - Zhou [32]	-	9.5812 s	1078.6812 s
Li [1] - Fort [2]	-	1.3780 s	1070.478 s
Zhou [32] - Li [1]	96.6834 s	34.2174 s	130.9008 s
Zhou [32] - Zhou [32]	-	6.6778 s	103.3612 s
Zhou [32] - Fort [2]	-	1.3278 s	98.0112 s
DMPHN [4]	-	-	0.0185 s
SRN [3]	-	-	0.596 s
DUBLID [38]	-	-	0.01808 s
DEBLUR [43]	-	-	216.9525 s

Cuadro 3.9: Tiempos usando el dataset de Nah et al.⁵

Para este dataset encontramos que los métodos basados en DL obtienen mucho mejor resultado que los analíticos. Siendo la mejor propuesta **SRN** [3], que presenta resultados mucho mejores que el resto de métodos en un tiempo ínfimo si comparamos, por ejemplo, con las propuestas que usan para estimar el emborronamiento el método propuesto por Li [1]. En la figura 3.4 se puede ver que las imágenes (g) [4] y (h) [3] son una buena propuesta a la hora de estimar el emborronamiento de las imágenes de este dataset. Para el ejemplo hemos escogido una imagen que presentaba un emborronamiento severo, y

⁵Como hemos comentado en la sección 3.2, podemos ver métodos en los que algunas entradas aparecen vacías. En el caso de los modelos que usan tanto la estimación de Li [1] como la de Zhou [32] toman el valor de la primera aparición, en este caso **Li [1] - Li [1]** y **Zhou [32] - Li [1]**. Para el resto de métodos tan sólo se mide el tiempo total entre que se introduce la imagen y se genera una salida.

Método	PSNR	SSIM
Li [1] - Li [1]	21.350	0.7190
Li [1] - Zhou [32]	21.7983	0.732
Li [1] - Fort [2]	21.176	0.698
Zhou [32] - Li [1]	22.7656	0.7561
Zhou [32] - Zhou [32]	22.980	0.764
Zhou [32] - Fort [2]	23.269	0.775
DMPHN [4]	31.5885	0.9390
SRN [3]	32.1931	0.9451
DUBLID [38]	25.6641	0.8363
DEBLUR [43]	23.9644	0.8310

Cuadro 3.10: PSNR y SSIM del dataset de Nah et al. El mejor resultado aparece resaltado en negrita.

aquí podemos ver cómo al usar un enfoque de DL donde contamos con un conjunto de datos enorme que cuenta con ejemplos similares al usado, las redes son capaces de recuperar la imagen a pesar de que la entrada presente un problema muy complejo.

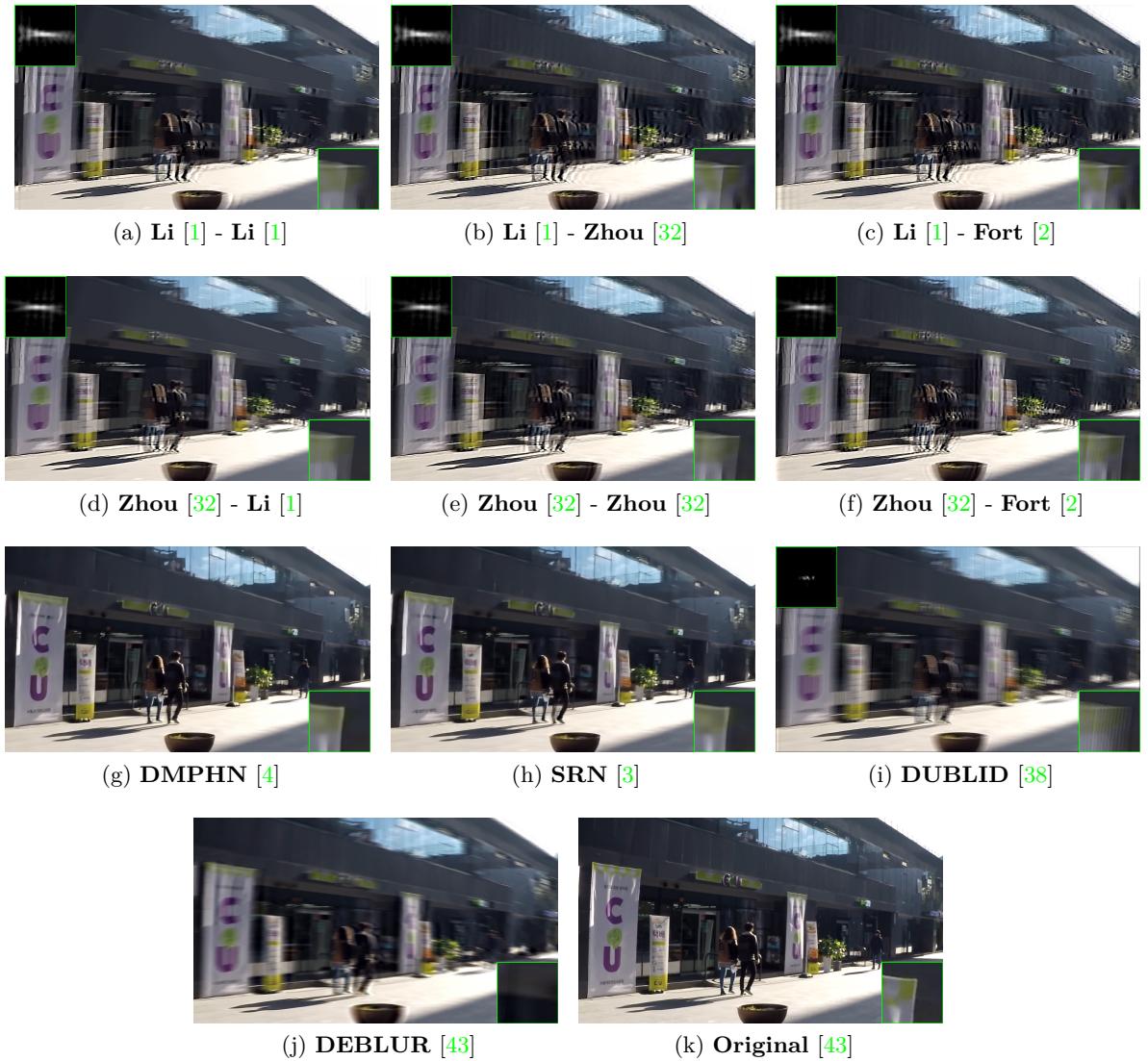


Figura 3.4: Resultados de la deconvolución del dataset de Nah et al.

Capítulo 4

Propuesta de combinación de métodos analíticos y de aprendizaje profundo para la deconvolución ciega de imágenes

Como hemos indicado anteriormente, en este trabajo fin de grado proponemos como aproximación a la deconvolución ciega de imágenes el postprocesamiento de la imagen obtenida por los métodos del estado del arte con el fin de lograr un resultado más natural que el que se obtiene haciendo uso de estos. Para ello, la mejora se realizará sobre la imagen que obtenemos tras aplicar un método de deconvolución ciega sobre una imagen borrosa inicial.

Nuestra aproximación consiste en usar una red CNN (*Convolutional Neural Network*, Red Neuronal Convolucional) para obtener la imagen latente a partir de dos imágenes iniciales: una estimación ruidosa con altas frecuencia obtenida con el filtro de Wiener y una imagen limpia pero con mucho menos detalles obtenida por otro método de deconvolución. Para asegurar que nuestra propuesta consigue ofrecer unos resultados aceptables debemos cerciorarnos de escoger aquél método (Zhou [32]) que nos proporcione una solución robusta y balanceada en distintos tipos de escenarios. Tras el estudio y comparación de distintos métodos del estado del arte hemos determinado que aquél que logra satisfacer estas necesidades es la aproximación propuesta por Zhou [32]. Combinando estas dos imágenes (figura 4.1) logramos paliar el problema del excesivo suavizado, provocado debido a la naturaleza de los filtros de paso bajo, que se da tras regularizar en deconvolución ciega a la par que eliminamos los artificios que se generan durante la convolución. Nuestra



Figura 4.1: En (a) tenemos la imagen estimada de Zhou y en (b) el resultado de aplicar el filtro de Wiener a esa imagen estimada

aproximación ofrece una salida similar a la de los métodos que añade detalles y resulta más natural al ojo humano, es por ello que para ofrecer una buena salida ha de presentar una buena entrada en la que el emborronamiento se haya estimado correctamente. En la figura 4.2 podemos ver la comparación entre la imagen observada que presenta emborronamiento, la original y la salida de nuestra propuesta introduciendo las imágenes de 4.1. Denominamos a nuestra propuesta *RDBNet enhancing for blind image deconvolution* (Red residual para la mejora de deconvolución ciega de imágenes) de ahora en adelante **RDBNet**.

4.1. Estructura de la propuesta

Para este algoritmo proponemos una red convolucional secuencial con la estructura de la figura 4.3. Tomamos como entrada la imagen estimada y su versión filtrada con una aproximación del filtro de Wiener, en la figura 4.1 podemos ver la clase de imágenes que usa nuestra propuesta. Podemos dividir la estructura en función del tipo de capas que usamos en cada etapa desde que introducimos los datos de entrada hasta que generamos la imagen final.

- Capa de reducción: En la primera etapa adoptamos la capa de reducción de [44] denominada *Pixel Shuffle Layer Downsampling*. En esta capa reducimos el tamaño de las imágenes de entrada a la mitad. En los trabajos de súper-resolución se parte de una imagen de baja resolución para generar otra de una resolución mayor que presenta más detalles que esta primera. En estos trabajos se asume que la imagen de baja resolución es una versión borrosa, con ruido y con una escala menor de

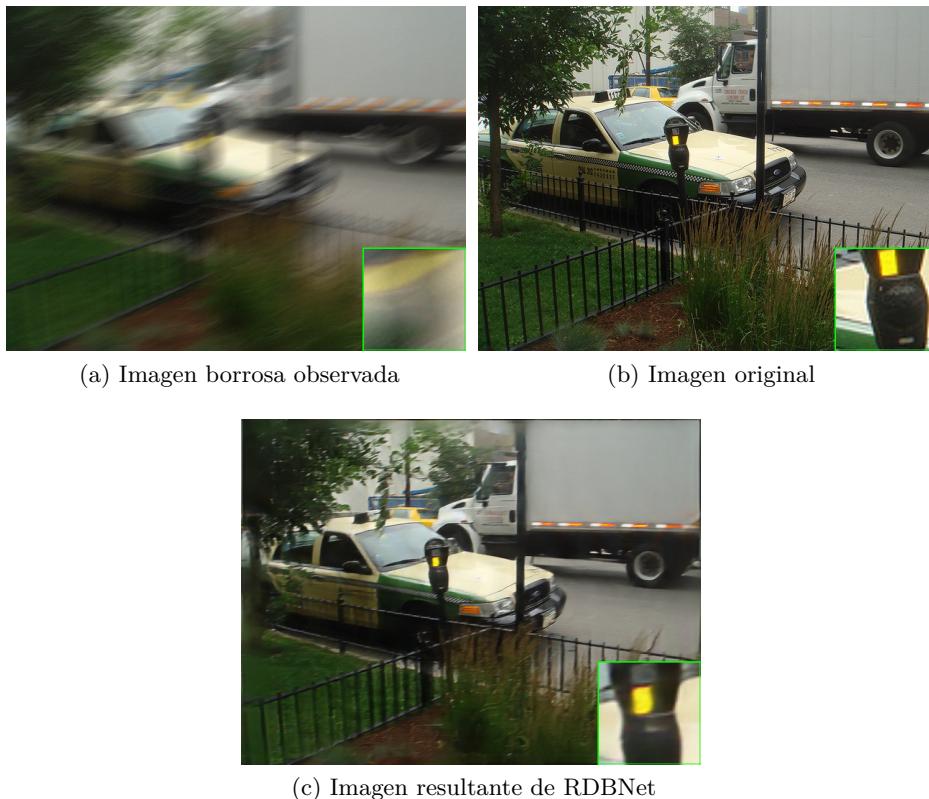


Figura 4.2: En (a) tenemos la imagen borrosa observada, en (b) la imagen original nítida y en (c) el resultado que obtenemos introduciendo las imágenes de 4.1 a nuestra propuesta

la imagen de alta resolución. En el trabajo [44] reescalan la imagen a su versión en la última capa de la red, lo que permite que se extraigan los mapas de características de la versión en baja resolución y se pueda trabajar con núcleos de menor tamaño. La reducción de la resolución y del tamaño del núcleo nos permite reducir el coste computacional y la complejidad del problema. Antes de llegar a la siguiente etapa contamos con una imagen con una resolución 4 veces menor a la resolución de la imagen original.

- **Capas de convolución:** En estas capas realizamos una operación de convolución sobre una señal compuesta por varios canales de entrada. Usamos este tipo de capas para agrupar la información que recogemos de capas anteriores y generar una salida en la que se combine esa información recogida previamente.
- **Capas ReLU:** Para evitar tener valores negativos en las neuronas añadimos capas ReLU junto a las capas de convolución, que actúan como una función lineal que se activa para los valores negativos y los evita. Esto resulta en un modelo que puede adaptarse a distintos tipos de problemas ya que ha aprendido de distintas entradas y tienen neuronas que se activan en función del tipo de problema. El uso de las capas ReLU puede resultar en un problema si tan sólo tenemos valores negativos y las capas no proporcionan ninguna información (*Dying ReLU*). Para evitar esto contamos con una variante llamada Leaky ReLU que soluciona el problema de las neuronas muertas que se quedan en el lado negativo y sólo proporcionan 0 como salida. En nuestro caso, las capas ReLU adoptan la función $y = 0.01x$ cuando $x < 0$. La variante Leaky ReLU es más balanceada, lo que resulta en un entrenamiento y aprendizaje más rápido.
- **Bloque residual:** En este bloque contamos 4 capas convolucionales (de convolución) en las que transmitimos la información de las 3 primeras a la última. En cada capa tomamos un número de canales de entrada distintos con el fin de recoger distinta información de la imagen y transmitirla a la capa final. En las 3 primeras capas usamos también una capa ReLU para evitar los negativos y en la última agrupamos toda la información. Repetimos este proceso 10 veces.
- **Capa de ampliación:** Al igual que en la primera etapa donde reducimos la resolución de la imagen, en esta etapa volvemos a reescalar la imagen a su tamaño original usando la capa de ampliación de [44] denominada *Pixel Shuffle Layer Upsampling*. Al realizar la reescala al tamaño original logramos corregir los errores que se producían durante la deconvolución y generaban artificios en la imagen final. Para la escala a la resolución original se usan los mapas de características que se han

extraído en las capas anteriores, resultando en una imagen con más detalle que la de entrada y que corrige aquellos fallos que la metodología clásica no puede tratar.

- Filtro dinámico: En esta etapa se trata la imagen resultante de la capa de ampliación y la imagen estimada de Zhou usando un filtro de Wiener. Se calcula la diferencia de las imágenes extrayendo la información de la entrada, lo que resulta en una imagen filtrada que contiene más información en las altas frecuencias y ha sido corregida previamente usando los mapas de características de los bloques anteriores.

En resumen, adoptamos las capas de reducción y ampliación de [44] denominadas *Pixel Shuffle Layer Downsampling/Upsampling* que nos permiten reducir las imágenes iniciales para un cálculo más rápido y después reescalar las imágenes a su tamaño original sin perder detalles y corrigiendo los artificios que pudieran tener. Reducimos a un cuarto el tamaño de las imágenes e iteran por un bloque residual de 4 capas convolucionales en el que transmitimos la información de las 3 primeras capas a la última. Repetimos este proceso 10 veces y agrupamos la información del bloque residual y la previa a este bloque (el resultado de las dos capas de reducción). Se reescalan las imágenes al tamaño original y calculamos la imagen latente mediante una capa de convolución y la de la diferencia usando la imagen obtenida con el filtro de Wiener y un filtro dinámico. La salida resulta de agrupar la información de la convolución de la imagen latente junto a la del filtro dinámico.

4.2. Entrenamiento de la red

Para entrenar la red hemos trabajado con un conjunto de más de 300.000 imágenes. Para seleccionar dicho conjunto hemos tratado las imágenes y hemos tratado como entrada aquellas que presenten un buen resultado inicial, en nuestro caso un PSNR mayor a 17.5 dB en las imágenes estimadas usando la aproximación de Zhou [32]. Como hemos visto anteriormente, generalmente el método propuesto por Zhou [32] presenta unos buenos resultados, el motivo de esta selección se debe a la propia naturalidad de las redes neuronales. Al tratar con imágenes que ya presentan un buen resultado inicial limitamos el dominio del problema y la red consigue ofrecer mejoras sustanciales dada una buena entrada. En caso de no haber filtrado el conjunto de datos para entrenar la red no podríamos asegurar que esta ofrece una mejora visible, dado a la pobre calidad de las imágenes con las que ha tratado. De misma forma que no es una buena idea seleccionar un conjunto muy grande para el entrenamiento tampoco lo es seleccionar uno muy pequeño con el objetivo de garantizar aún más una mejora sustancial. En este caso, al tratar con

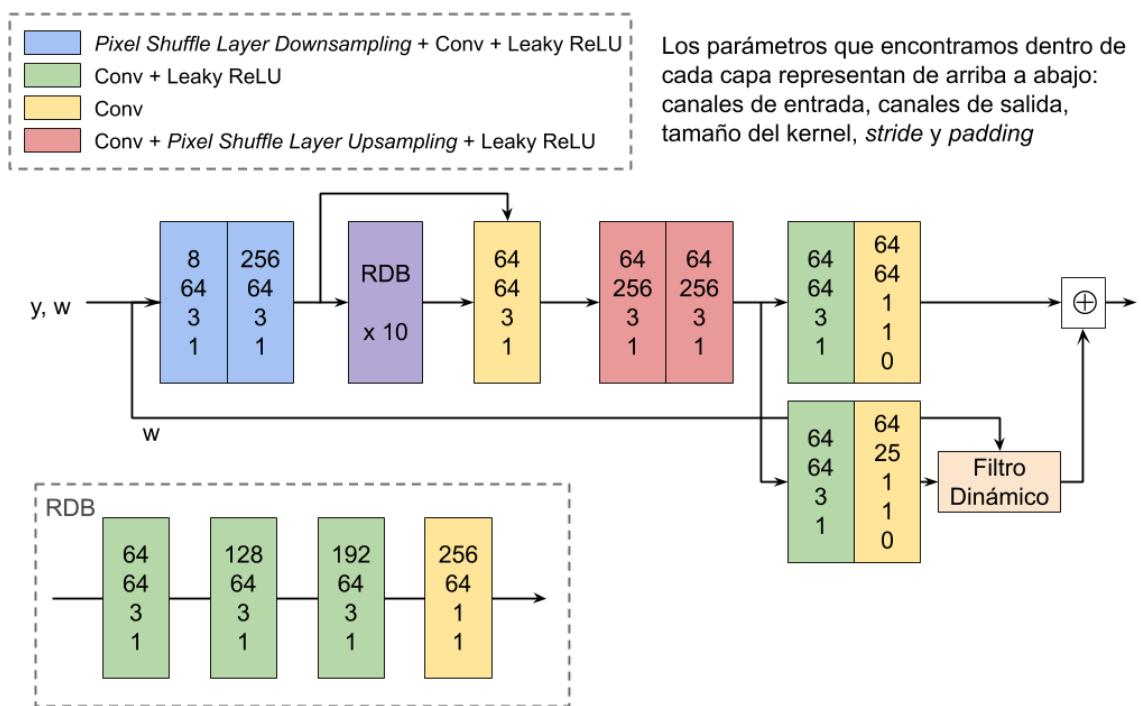


Figura 4.3: Estructura de la red convolucional propuesta donde y, w representan la imagen resultado de la estimación de Zhou [32] y su versión filtrada con una aproximación de Wiener.

Propuesta de combinación de métodos analíticos y de aprendizaje profundo para la deconvolución ciega de imágenes 61

conjuntos de datos muy pequeños que no garanticen que trabajamos con un conjunto de variables aleatorias independientes e idénticamente distribuidas puede llegar a darse sobreajuste. Esto es cuando la red tan sólo aprende a tratar con los ejemplos de entrenamiento, ofreciendo un muy buen resultado pero no siendo capaz de tratar nuevas entradas que no conoce. Tras varias pruebas encontramos que el conjunto escogido cuenta con la información necesaria para que el modelo sea capaz de tratar con una amplia cantidad de imágenes naturales llegando a ofrecer un muy buen resultado.

Capítulo 5

Resultados experimentales de la propuesta de combinación de métodos analíticos y de aprendizaje profundo

En este capítulo compararemos nuestra propuesta con el estado del arte en deconvolución ciega de imágenes. En las siguientes tablas podemos ver una media del tiempo total que tardan los métodos comparados en generar la salida para cada uno de los conjuntos de datos que hemos visto en el capítulo 3. En el caso de nuestro método, RDBNet, este tiempo es la suma del tiempo total del método de Zhou [32] con el tiempo empleado por la red en estimar la nueva imagen. Como se puede comprobar, en nuestra propuesta el tiempo viene condicionado por el método analítico que ofrece la estimación del emborronamiento. Esto es debido a la naturaleza iterativa de Zhou [32] y al hecho de que se ejecuta en la CPU, no en la GPU como RDBNet.

Estudiamos el comportamiento de nuestro método RDBNet y lo comparamos con otros métodos como ya hemos comentado. Para comparar cómo se comporta nuestra propuesta de mejora analizaremos los resultados en los mismos conjuntos de datos, aunque para esta comparativa escogeremos los 4 métodos más representativos del capítulo 3. Hemos optado por escoger los métodos analíticos, pues como pudimos estudiar, son los que otorgan un mejor resultado gracias a que la información que añadimos mediante las distribuciones a priori nos permiten limitar el espacio de soluciones y encontrar la óptima. También porque proponemos una mejora a las desventajas que presentan estos métodos a la hora de realizar la operación de deconvolución.

Dataset	Método	PSNR	SSIM	T. total
Levin [39]	Li [1] - Li [1]	27.9298	0.8571	38.6911 s
	Li [1] - Fort [2]	28.3216	0.8619	37.8459 s
	Zhou [32] - Zhou [32]	<u>29.1484</u>	<u>0.9005</u>	<u>7.7476 s</u>
	Zhou [32] - Fort [2]	29.6043	0.9017	7.2928 s
	Zhou [32] - RDBNet	26.1736	0.8262	7.7708 s
Köhler [40]	Li [1] - Li [1]	25.5438	0.8236	279.42 s
	Li [1] - Fort [2]	24.9299	<u>0.7927</u>	254.4865 s
	Zhou [32] - Zhou [32]	<u>25.1419</u>	0.7916	<u>76.593 s</u>
	Zhou [32] - Fort [2]	25.0371	0.7902	72.6827 s
	Zhou [32] - RDBNet	24.8910	0.7415	76.6169 s
Sun [41]	Li [1] - Li [1]	27.3926	0.8139	290.2647 s
	Li [1] - Fort [2]	28.5665	0.8222	281.1109 s
	Zhou [32] - Zhou [32]	29.6492	0.8433	<u>87.0387 s</u>
	Zhou [32] - Fort [2]	<u>29.5543</u>	<u>0.8287</u>	82.0422 s
	Zhou [32] - RDBNet	20.2020	0.3298	87.0529 s
Nah [42]	Li [1] - Li [1]	21.350	0.7190	1103.3704 s
	Li [1] - Fort [2]	21.176	0.698	1070.478 s
	Zhou [32] - Zhou [32]	22.980	<u>0.764</u>	<u>103.3612 s</u>
	Zhou [32] - Fort [2]	23.269	0.775	98.0112 s
	Zhou [32] - RDBNet	<u>23.015</u>	0.748	103.3773 s

Cuadro 5.1: PSNR, SSIM y tiempo medio total de cada método en los distintos datasets. El mejor resultado aparece resaltado en negrita y el segundo subrayado. Cada pareja **X - Y** denota el método usado para estimar el núcleo de emborronamiento y para recuperar la imagen respectivamente, excepto para nuestra propuesta, en la que la imagen usa la estimación del emborronamiento y la recuperación de la imagen de Zhou [32] y después la postprocesamos usando nuestro método.

En la tabla 5.1 podemos ver los resultados de medir PSNR, SSIM y el tiempo total para cada uno de los conjuntos de datos y 5 métodos, siendo uno de ellos nuestra propuesta, RDBNet. Para las pruebas hemos usado los métodos de estimación propuestos por Zhou [32] y Li [1], así como la deconvolución que proponen y añadiendo a estos últimos el método de deconvolución no ciega que ofrece Fortunato en el trabajo [2], a todo esto añadimos nuestra propuesta en combinación con el método de Zhou [32].

Tras estudiar los resultados de la tabla 5.1 podemos observar que la propuesta de mejora obtiene un resultado similar en algunos conjuntos de datos y en otros llega a tener una diferencia abismal respecto de otras propuestas. Si tan sólo nos fijamos en las métricas podemos ver que no se obtiene una mejora sustancial en comparación con el resto de métodos. También compararemos los resultados para imágenes específicas y veremos más detalladamente cómo actúa nuestro algoritmo. En estas imágenes, al igual que en el capítulo 3, mostraremos la estimación del núcleo que nos proporcionan los métodos de **Li** y **Zhou**. En este caso, al mostrar la propuesta **Zhou - RBNet**, no significa que la red es la encargada de deconvolucionar usando la estimación del emborronamiento para recuperar la imagen nítida, al contrario que en los otros casos. Para este ejemplo partimos de la imagen que nos otorga el método de Zhou [32] mediante la estimación del emborronamiento y su posterior recuperación de la imagen, con esa imagen y una nueva versión filtrada usando un filtro de Wiener generamos otra nueva, que es la que mostramos en la figura y resulta de la salida de la red propuesta.

En la figura 5.1 podemos ver un ejemplo para una imagen específica correspondiente al dataset de Sun [41]. Para este conjunto de datos, nuestro algoritmo obtenía el peor resultado midiendo el PSNR y SSIM medio, pero al ver la imagen más detalladamente se puede observar que no proporciona una mala salida. La imagen cuenta con más ruido que las estimaciones y que la original, pero si nos fijamos más detalladamente en los bordes y las texturas podemos ver que conseguimos recuperar más información. En el caso de la figura 5.2, analizamos los resultados con una imagen del dataset de Köhler [40]. En este caso, la imagen resultante de nuestra propuesta genera un muy buen resultado, añadiendo detalles que los otros métodos son incapaces de recuperar, esto lo podemos ver más claramente en la zona ampliada, donde se aprecian más detalles que en el resto de propuestas. A pesar de proporcionar un peor resultado, si tan sólo medimos PSNR y SSIM, a la hora de comparar con el resto de métodos o con la imagen original nuestro algoritmo presenta una imagen que ofrece detalles que no podemos recuperar mediante los métodos clásicos de deconvolución, debido a la naturaleza de los mismos. Conseguimos así una imagen agradable al ojo humano, una imagen que no resulta muy artificial y que presenta las características propias de una imagen observada sin emborronamiento.

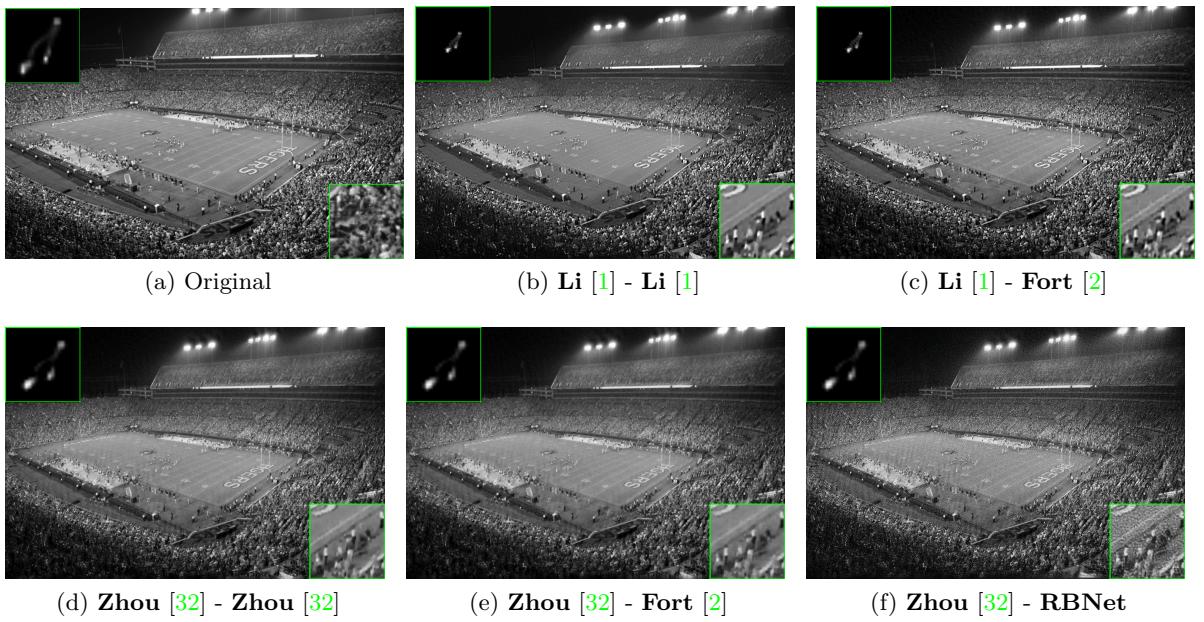


Figura 5.1: Comparativa de los métodos analíticos y de la propuesta de mejora para el dataset de Sun et al. En (a) podemos ver la imagen original nítida y en el resto de imágenes una pareja $\mathbf{X} - \mathbf{Y}$ que representan la combinación de los métodos usados para estimar el núcleo de emborronamiento y recuperar la imagen final, salvo para nuestra propuesta en la que postprocesamos la imagen resultante de usar el método de Zhou [32] para estimar el emborronamiento y la imagen final.

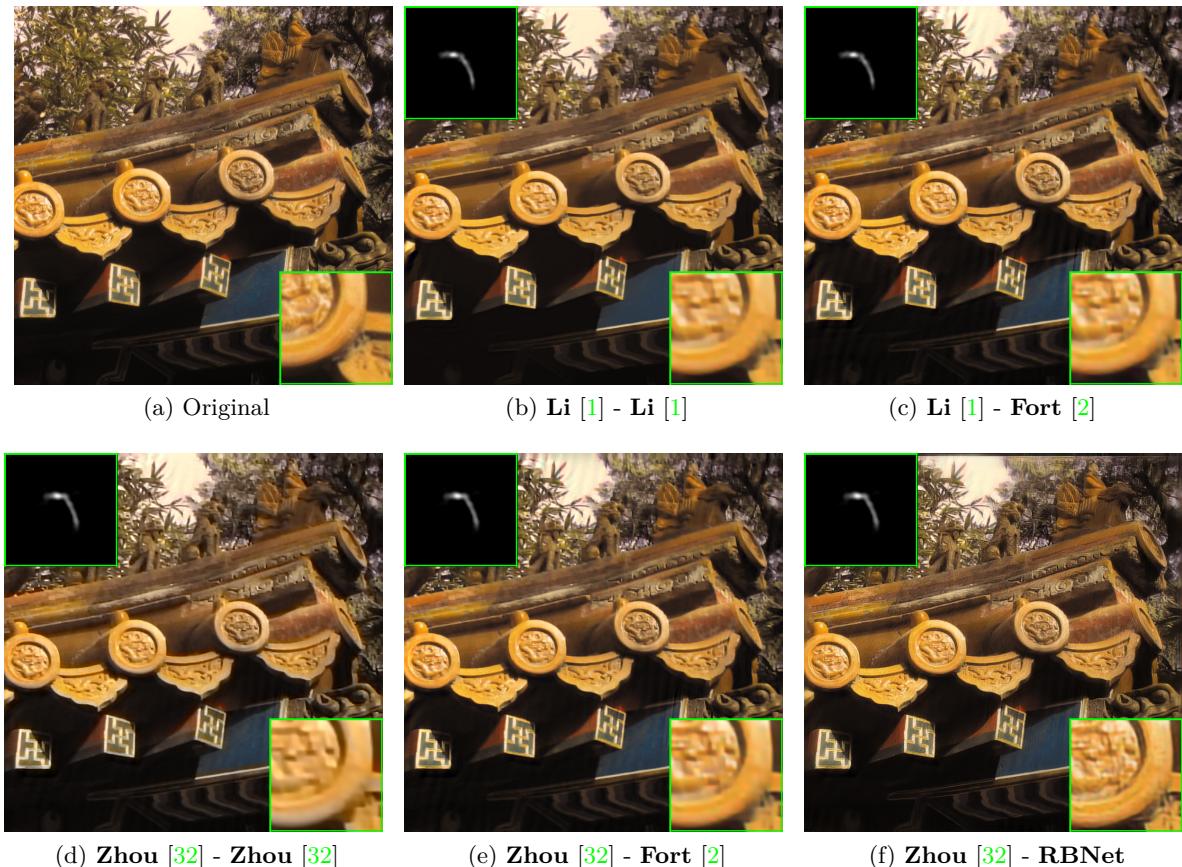


Figura 5.2: Comparativa de los métodos analíticos y de la propuesta de mejora para el dataset de Köhler et al. En (a) podemos ver la imagen original nítida y en el resto de imágenes una pareja $\mathbf{X} - \mathbf{Y}$ que representan la combinación de los métodos usados para estimar el núcleo de emborronamiento y recuperar la imagen final, salvo para nuestra propuesta en la que postprocesamos la imagen resultante de usar el método de Zhou [32] para estimar el emborronamiento y la imagen final.

Capítulo 6

Conclusiones del trabajo realizado

En este trabajo hemos abordado el problema de la deconvolución ciega de imágenes, un problema del campo de la visión por computador que pretende recuperar una imagen nítida a partir de una que presenta un emborronamiento y ruido desconocidos. Para resolverlo hemos propuesto en el mismo una combinación de métodos analíticos junto a modelos basados en aprendizaje profundo que logra paliar los problemas que han sufrido estos primeros a lo largo de los años. Recientemente, gracias al campo de la inteligencia artificial, encontramos una nueva forma de resolver los problemas y que junto a la metodología clásica logra ofrecer un resultado que hace unos años era impensable.

Para concluir, en este trabajo se ha podido ver una mejora a la metodología clásica usando técnicas basadas en DL, aunque no tenemos por qué quedarnos ahí. Hemos probado que nuestro método, RDBNet, es capaz de mejorar las imágenes que presentan un suavizado excesivo y artificial debido a la propia naturaleza de los métodos analíticos, pero podríamos llevar nuestra propuesta a un dominio en el que contamos con imágenes que ya presentan un buen resultado inicial con el objetivo de mejorárlas aún más. Gracias a la estructura usada contamos con un método rápido y fiable que podría ser añadido a cualquier sistema con el fin de mejorar las imágenes para un posterior análisis, como podría ser el caso de problemas en el que necesitemos reconocer objetos usando visión por computador, o simplemente para mejorar una imagen, por ejemplo, en el caso de querer recuperar una imagen antigua que no contase con mucha calidad debido a las limitaciones de la época. En un futuro, gracias al avance en el campo del DL y a la potencia de cómputo, podríamos presentar una propuesta más compleja que lograse refinar aún más el resultado supliendo aquellos pequeños defectos que tiene actualmente y en un tiempo mucho menor. Los resultados de este trabajo fin de grado serán

enviados al congreso *IEEE International Conference on Image Processing* 2021.

Bibliografía

- [1] Lerenhan Li, Jinshan Pan, Wei-Sheng Lai, Changxin Gao, Nong Sang, and Ming-Hsuan Yang. Blind image deblurring via deep discriminative priors. *International Journal of Computer Vision*, 127(8):1025–1043, 2019.
- [2] Horacio E. Fortunato and Manuel M. Oliveira. Fast high-quality non-blind deconvolution using sparse adaptive priors. *The Visual Computer*, 30(6-8):661–671, 2014.
- [3] Xin Tao, Hongyun Gao, Xiaoyong Shen, Jue Wang, and Jiaya Jia. Scale-recurrent Network for Deep Image Deblurring. *CVPR*, 2018.
- [4] Hongguang Zhang, Yuchao Dai, Hongdong Li, and Piotr Koniusz. Deep Stacked Hierarchical Multi-Patch Network for Image Deblurring. *CVPR*, 2019.
- [5] P. Ruiz, X. Zhou, J. Mateos, R. Molina, and A. K. Katsaggelos. Variational Bayesian blind image deconvolution: A review. *Digital Signal Processing*, 47:116–127, 2015.
- [6] Victor Powell. Image kernels explained visually.
- [7] X. Zhou, J. Mateos, F. Zhou, R. Molina, and A. K. Katsaggelos. Variational Dirichlet blur kernel estimation. 24:5127–5139, 2015.
- [8] D. Perrone, R. Diethelm, and P. Favaro. Blind deconvolution via lower-bounded logarithmic image priors. In *International Conference on Energy Minimization Methods in Computer Vision and Pattern Recognition (EMMCVPR)*, 2015.
- [9] R. Fergus, B. Singh, A. Hertzmann, S. T. Roweis, and W. T. Freeman. Removing camera shake from a single photograph. *ACM Trans. Graph*, 25(3), 2006.
- [10] X. Zhou, M. Vega, F. Zhou, R. Molina, and A. K. Katsaggelos. Fast Bayesian blind deconvolution with Huber super Gaussian priors. *Digital Signal Processing*, 60:122–133, 2017.

- [11] Lei Chen, Quansen Sun, and Fanhai Wang. Adaptive blind deconvolution using generalized cross-validation with generalized l_p/l_q norm regularization. *Neurocomputing*, 399:75 – 85, 2020.
- [12] Li Xu, Shicheng Zheng, and Jiaya Jia. Unnatural L0 sparse representation for natural image deblurring. In *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition, CVPR '13*, pages 1107–1114, Washington, DC, USA, 2013. IEEE Computer Society.
- [13] J. Pan, D. Sun, H. Pfister, and M. Yang. Blind image deblurring using dark channel prior. In *2016 IEEE Conf. Comput. Vision and Pattern Recognition*, pages 1628–1636, 2016.
- [14] Y. Yan, W. Ren, Y. Guo, R. Wang, and X. Cao. Image deblurring via extreme channels prior. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6978–6986, 2017.
- [15] B. Zhang, R. Liu, H. Li, Q. Yuan, X. Fan, and Z. Luo. Blind Image Deblurring Using Adaptive Priors. In *Internet Multimedia Computing and Service, Communications in Computer and Information Science*, pages 13–22. Springer, Singapore, August 2017.
- [16] V. Jain and S. Sebastian. Natural image denoising with convolutional networks. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 769–776. Curran Associates, Inc., 2009.
- [17] Junyuan Xie, Linli Xu, and Enhong Chen. Image Denoising and Inpainting with Deep Neural Networks. *Conference on Neural Information Processing Systems*, pages 1–9, 2012.
- [18] 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*, 38(2):295–307, February 2016.
- [19] M. V. Afonso, J. M. Bioucas-Dias, and M. A. T. Figueiredo. Fast image recovery using variable splitting and constrained optimization. *IEEE transactions on image processing*, 19(9)(3):2345–2356, 2010.
- [20] T. Meinhardt, Michael Möller, C. Hazirbas, and D. Cremers. Learning proximal operators: Using denoising networks for regularizing inverse imaging problems. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 1799–1808, 2017.
- [21] K. Zhang, W. Zuo, S. Gu, and L. Zhang. Learning deep cnn denoiser prior for image restoration. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3929–3938, 2017.

- [22] N. Parikh and S. Boyd. Proximal algorithms. *Found. Trends Optim.*, 1(3):127–239, January 2014.
- [23] Liqing Huang and Youshen Xia. Joint blur kernel estimation and cnn for blind image restoration. *Neurocomputing*, 396:324 – 345, 2020.
- [24] D. Gong, Z. Zhang, Q. Shi, A. van den Hengel, C. Shen, and Y. Zhang. Learning deep gradient descent optimization for image deconvolution. *IEEE Transactions on Neural Networks and Learning Systems*, in press, 2020.
- [25] S. Nah, T. H. Kim, and K. M. Lee. Deep multi-scale convolutional neural network for dynamic scene deblurring. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 257–265, July 2017.
- [26] O. Kupyn, V. Budzan, M. Mykhailych, D. Mishkin, and J. Matas. Deblurgan: Blind motion deblurring using conditional adversarial networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8183–8192, 2018.
- [27] O. Kupyn, T. Martyniuk, J. Wu, and Z. Wang. Deblurgan-v2: Deblurring (orders-of-magnitude) faster and better. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 8877–8886, 2019.
- [28] Li Xu, Cewu Lu, Yi Xu, and Jiaya Jia. Image smoothing via l0 gradient minimization. *ACM Transactions on Graphics (SIGGRAPH Asia)*, 2011.
- [29] S. Cho and S. Lee. Fast motion deblurring. *ACM Trans. Graph.*, 28(5), 2009.
- [30] Zhixun Su Jinshan Pan, Zhe Hu and Ming-Hsuan Yang. Deblurring text images via l0-regularized intensity and gradient prior. 2014.
- [31] Jinshan Pan, Deqing Sun, Hanspeter Pfister, and Ming-Hsuan Yang. Blind image deblurring using dark channel prior. pages 1628–1636, 06 2016.
- [32] Xu Zhou, Miguel Vega, Fugen Zhou, Rafael Molina, and Aggelos K. Katsaggelos. Fast bayesian blind deconvolution with huber super gaussian priors. *Digital Signal Processing: A Review Journal*, 60:122–133, jan 2017.
- [33] Robert Fergus, Barun Singh, Aaron Hertzmann, Sam Roweis, and William Freeman. Removing camera shake from a single photograph. *ACM Trans. Graph.*, 25:787–794, 07 2006.

- [34] James Miskin and David J.C. MacKay. Ensemble learning for blind image separation and deconvolution, 2000.
- [35] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. 2015. cite arxiv:1512.03385Comment: Tech report.
- [36] Seungjun Nah, Tae Hyun Kim, and Kyoung Mu Lee. Deep multi-scale convolutional neural network for dynamic scene deblurring, 2016.
- [37] Xin Tao, Hongyun Gao, Renjie Liao, Jue Wang, and Jiaya Jia. Detail-revealing deep video super-resolution. *CoRR*, abs/1704.02738, 2017.
- [38] Yuelong Li, Mohammad Tofighi, Junyi Geng, Vishal Monga, and Yonina Eldar. Efficient and interpretable deep blind image deblurring via algorithm unrolling. *IEEE Transactions on Computational Imaging*, PP:1–1, 01 2020.
- [39] Anat Levin. Understanding and evaluating blind deconvolution algorithms.
- [40] Rolf Köhler. Recording and playback of camera shake: benchmarking blind deconvolution with a real-world database.
- [41] Libin Sun. Recording and playback of camera shake: benchmarking blind deconvolution with a real-world database.
- [42] Seungjun Nah. Deep multi-scale convolutional neural network for dynamic scene deblurring.
- [43] O. Whyte, J. Sivic, A. Zisserman, and J. Ponce. Non-uniform deblurring for shaken images. *International Journal of Computer Vision*, 98(2):168–186, 2012.
- [44] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P. Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. *CoRR*, abs/1609.05158, 2016.

