Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising Comprensión y análisis

Karlo Jair Guevara Diaz
José Ángel Alejandro Soto
Angel Gilberto Ayala Perez
Universidad de Guanajuato - DEMAT
11 de abril de 2023

Resumen

Durante la realización de está tarea estuvimos trabajando con el paper Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising, se mostrarán comentarios acerca de la arquitectura de la red así como el proceso de entrenamiento y los resultados obtenidos

1. Sobre la arquitectura de la red

Estructura

Buscamos analizar el comportamiento y la estructura del aprendizaje residual de redes neuronales convolucionales para la eliminación de ruido (DnCNN).

Iniciamos viendo la estructura general de la red (reduciendo el numero de profundidad a 2) es de la siguiente forma (figura 1).

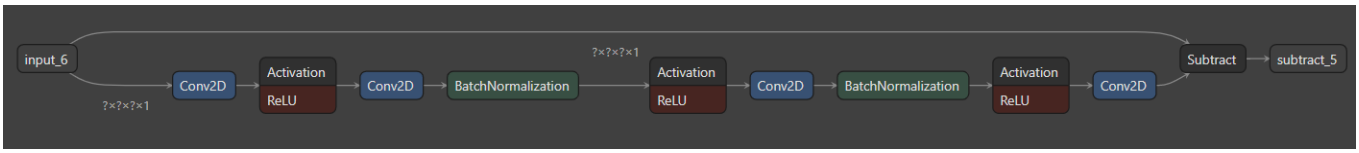


Figura 1

Tenemos que al entrar los datos a la red se separan en dos partes la inferior y la superior, en la inferior es donde se aproxima el ruido de la imagen y en la superior se mantiene la

imagen original, luego en 'Subtract' se restan estos dos resultados (el ruido y la imagen original) para obtener la imagen sin ruido.

Podemos notar que la red consta de las capas normalizadoras por lotes, las capas Relu y las capas convolucionales con kerneles 3x3 (en este orden). Procederemos a explicar la importancia de cada capa.

Capas convolucionales

Una capa convolucional consiste en aplicar un kernel de $n \times m$ a nuestra imagen, lo que nos genera una nueva imagen (la cual seria la salida de la capa convolucional). Esto de la siguiente forma, tomando una matriz de 3×3 como nuestro kernel por ejemplo la matriz $\begin{pmatrix} 1/2 & 1 & 1 \\ 1 & 1/2 & 1 \\ 1 & 1 & 1/2 \end{pmatrix}$ y al aplicarla a nuestra imagen la cual es otra matriz obtenemos lo siguiente.

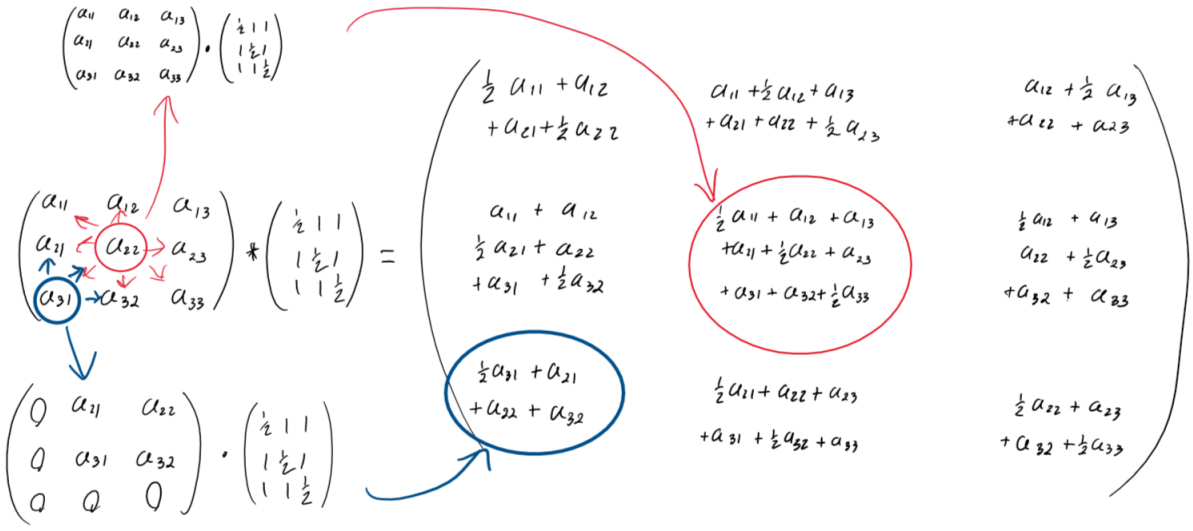


Figura 2

Para aplicar la matriz kernel a la entrada $a_{3,1}$ se tiene que agregar ceros al rededor (por que son los valores de la frontera de la imagen) y hacer producto punto con la matriz generado (flechas azules), por otro lado al aplicar la matriz kernel a la entrada $a_{2,2}$ es hacer el producto punto con la matriz de los vecinos de $a_{2,2}$ una matriz de 3×3 con la matriz kernel (flechas rojas).

La importancia de dicho kernel en el análisis de imágenes es que captura la información espacial de la imagen. Por ejemplo aplicando la siguiente matriz kernel $\begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$ (conocida

como el operador Sobel) a una imagen obtendríamos los bordes verticales de la imagen (figura 3).

Figura 3

Tenemos que todas las capas convulucionales en nuestra red DnCNN son de kernel 3x3, esto ya que esta dimensión es la mas pequeña en un kernel capas de captar características en una imagen como bordes horizontales o verticales.

Normalización por lotes

Otra capa importante de nuestra red es la capa de normalización por lotes. Esta capa lo que hace es normalizar los datos de salida de las capas convolucionales antes de entrar a nuestra función de activación Relu para normalizar las activaciones de salida. La utilidad de dicha capa es que al normalizar los datos tenemos menos varianza en nuestros datos entre las capas de la red, esto ayuda al descenso de gradiente estocástico para obtener una convergencia mas rápida.

Capa Relu

Tenemos que nuestra función de activación es la función Relu (unidad lineal rectificada) que esta dada por $f(x) = \text{máx}\{0, x\}$. El tener esta función nos ayuda a ajustar nuestra red al problema que no es lineal. El motivo por el cual se suele usar Relu es por su bajo costo computacional tanto en el calculo de la función como el de su derivada y por su gradiente constante tenemos una aprendizaje mas rápido a comparación de otras funciones como tangente hiperbólica o logística.

Pruebas del modelo

Probaremos y entrenaremos la red DnCNN, después modificaremos algunas características de la red como quitar las capas normalizadoras por lotes para comparar el rendimiento del modelo resultante como su comportamiento durante el entrenamiento.

Modelo DnCNN

Ya vimos que la red del modelo DnCNN tiene la siguiente estructura (figura 4),

Figura 4: Solo ponemos 2 de 15 capas para ver toda la red

Al entrenar la red obtenemos la siguiente función de costo en cada época.

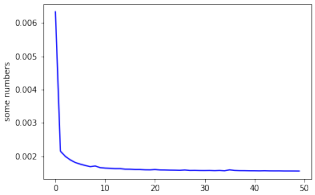


Figura 5

Luego aplicando la red para limpiar el ruido de una imagen obtenemos lo siguiente.

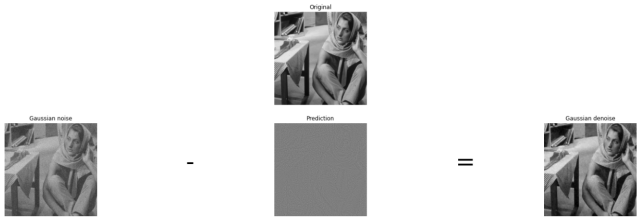


Figura 6

Modelo DnCNN sin capa normalizadoras

Al quitar las capas normalizadoras del modelo original obtenemos la siguiente estructura de la red (figura 7),

Figura 7: Solo ponemos 2 de 15 capas para ver toda la red

Durante el entrenamiento obtuvimos la siguiente gráfica de la función de costo.

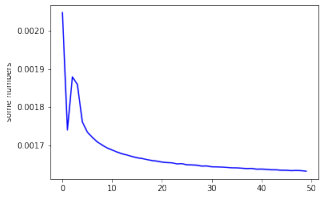


Figura 8

Podemos ver como esperábamos que tiene mas varianza, lo que hace que la convergencia sea mas lenta. Al aplicar la red entrenada para limpiar el ruido de una imagen obtenemos lo siguiente.

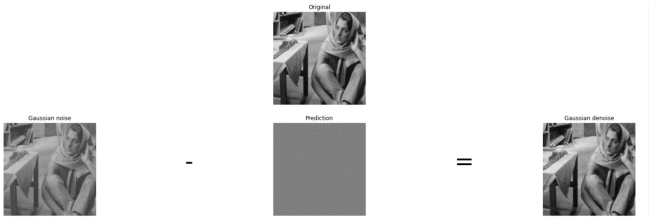


Figura 9

Modelo DnCNN cambiando Relu por Swish

Al cambiar las funciones de activación Relu por Swish obtenemos la siguiente estructura de la red (figura 6).

Figura 10: Solo ponemos 2 de 15 capas para ver toda la red

Al entrenar la nueva red obtenemos los siguientes resultados.

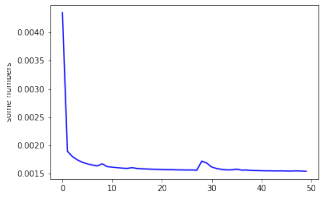


Figura 11

Obtenemos un comportamiento estable y una convergencia rápida (salvo el salto que da al rededor de la etapa 28). Al aplicar la red entrenada para limpiar el ruido de una imagen obtenemos lo siguiente.

Modelo DnCNN con 2 capas en lugar de 15

Al tomar 2 capas en lugar de 15 tenemos que la red tiene la siguiente forma, solo que esta ves la imagen si es toda la red.

Figura 12: Toda la red con 2 capas

Al entrenar la red obtenemos la siguiente función de costo en cada época, es un comportamiento muy parecido al de 15 capas solo que la función de costo converge a un valor un poco mayor.

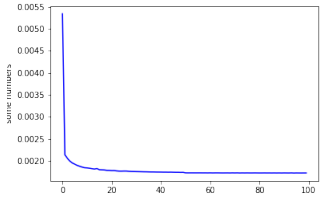


Figura 13

Luego aplicando la red para limpiar el ruido de una imagen obtenemos lo siguiente.

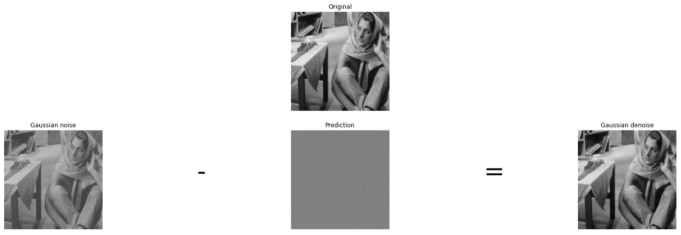


Figura 14

Modelo DnCNN sin capas normalizadoras y 2 capas en lugar de 15

Al quitar las capas normalizadoras del modelo original y poner 2 capas en lugar de 15 obtenemos la siguiente estructura de la red (figura 7),

Figura 15: En este caso es toda la red

Tenemos que tiene un comportamiento mas aleatorio que el modelo de 2 capas pero con normalización. Luego aplicando la red para limpiar el ruido de una imagen obtenemos lo siguiente.

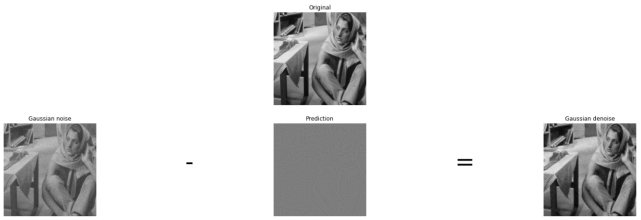


Figura 16

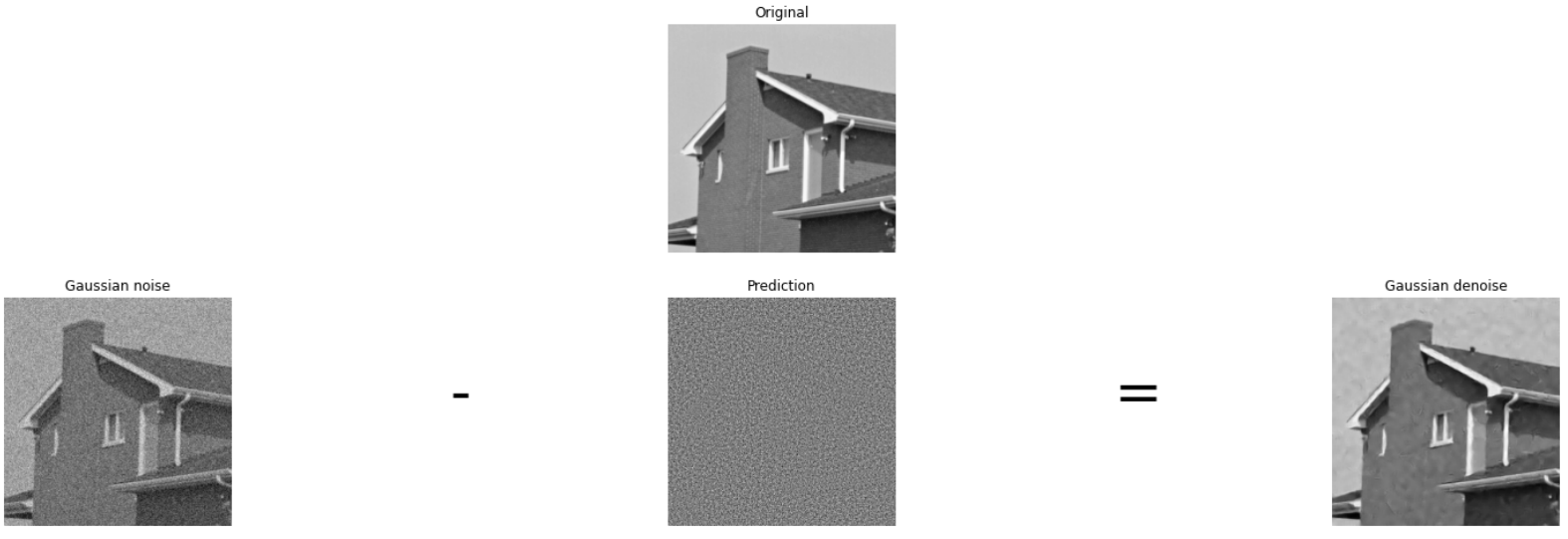
2. Resultados en el Test

El dataset para el test está divido en 2, el Set12 y el Set68 podemos ver el desempeño de cada modelo en estos test en la siguiente tabla

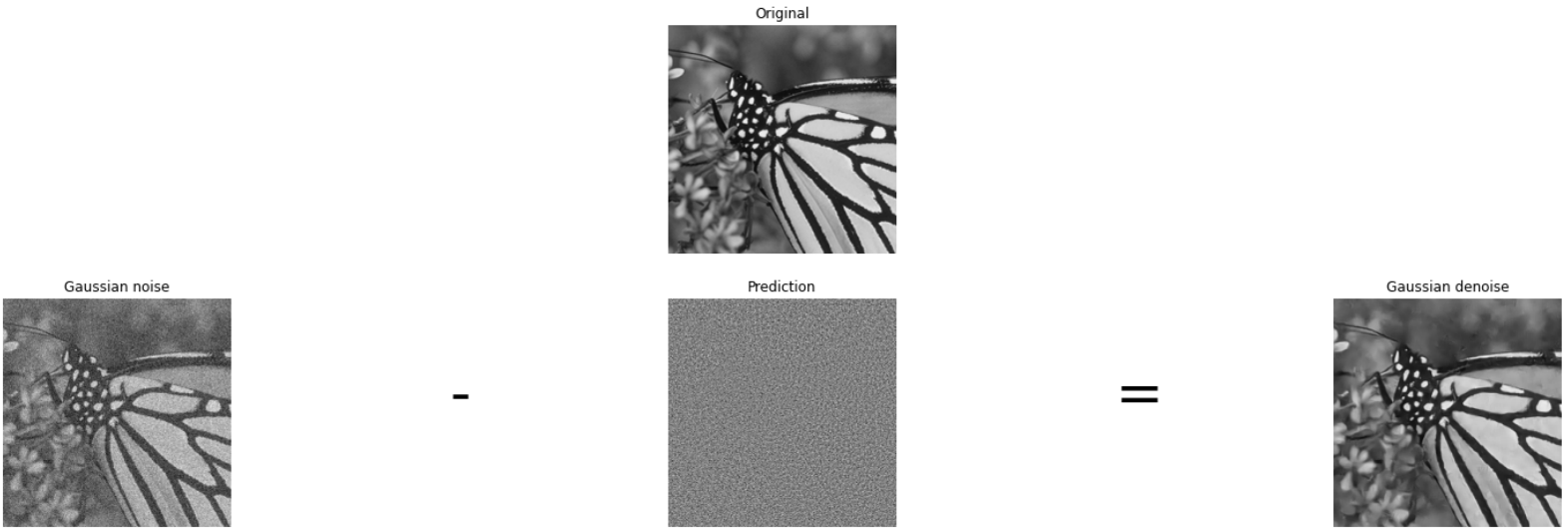
Modelo	Set 12		Set 68	
	Promedio PSNR	Promedio SSIM	Promedio PSNR	Promedio SSIM
Original	29.5868	0.91283	28.6183	0.8855
2 capas en lugar de 15	29.5835	0.9127	28.6158	0.8854
Sin normalización por lotes	**29.6183**	**0.9135**	**28.6222**	**0.8857**
Sin normalización lotes 2 capas	29.5985	0.9133	28.6160	0.8855
Swish	29.6082	0.9133	28.6209	0.8855

Podemos notar que el modelo sin normalización por lotes es el que obtuvo un mejor desempeño (aunque con poca diferencia) en ambos test y con ambas métricas

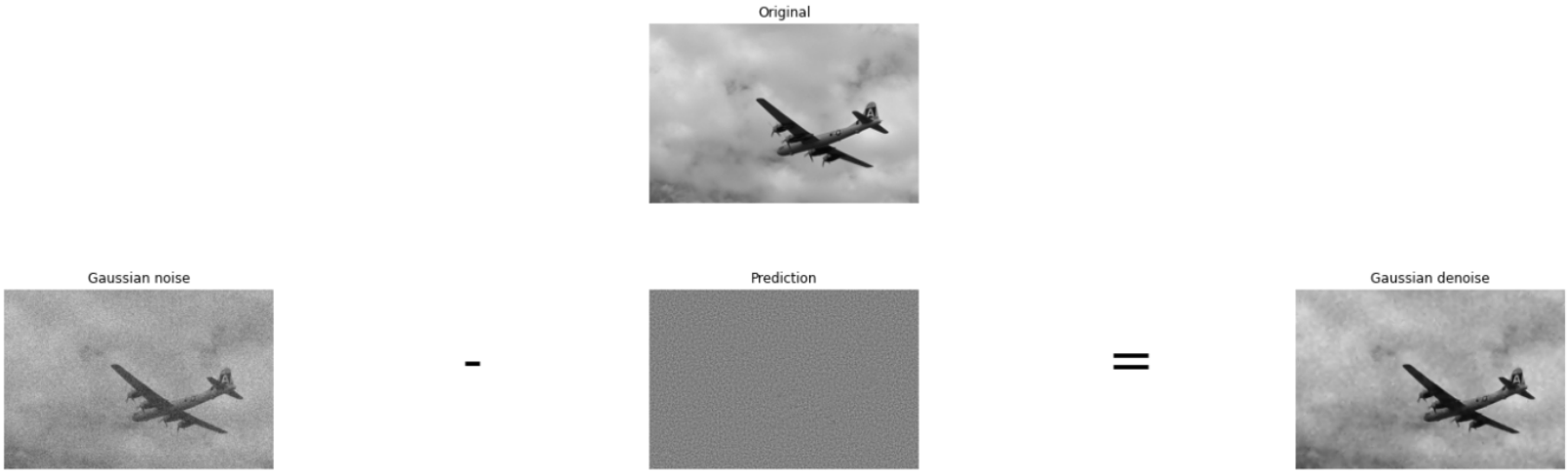
En el test 12 lo hizo peor según la métrica PSNR en la siguiente imagen con un valor de 31.6837



Por otro lado, según la métrica SSIM con un valor de 0.9413 lo hizo peor en



En el test 68 lo hizo peor en la misma imagen con una métrica de 35.2150 en PSNR y de 0.9615 en SSIM para la siguiente imagen



3. Conclusión

La normalización por lotes permite tiempos de entrenamiento menores, y a pesar de obtener ligeramente peores resultados, estos son muy similares.

Igualmente, el entrenamiento con 2 capas también obtiene resultados muy similares al de 15 capas. Por lo que si no disponemos de tiempo suficiente para entrenar la red, el entrenamiento con 2 capas y normalizado prevee resultados satisfactorios.

Referencias

[1] Zhang K, Zuo W, Chen Y, Meng D, Zhang L. Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising. IEEE Transactions on Image Processing. 2017 jul;26(7):3142-55. Available from: `https://doi.org/10.1109%2Ftip.2017.2662206`.

[2] husqin. Implemention of the paper Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising;. Available from: `https://github.com/husqin/DnCNN-keras`.