Compresión de Imagines con Redes Neuronales Profundas Basadas en la Calidad Visual Perceptual

Sebastian Guarnaccio†

†Campus Universitario, Buenos Aires, Argentina

sguarnaccio@gmail.com

‡Dto Electrónica UTN FRBA

**Resumen—** El objetivo del actual documento es presentar un framework de compresión de imágenes basado en redes neuronales el cual esta optimizado en función de la percepción visual de la imagen. Para esto se utilizó una red neuronal convolucional para obtener una representación más compacta de la imagen, la cual será posteriormente tratada para obtener la codificación correspondiente de la misma. Finalmente, a partir de la representación binaria del código, con una red neuronal deconvolucional se construye el decoder responsable de reconstruir la imagen tratando de perder la menor cantidad de información posible y maximizando la calidad percibida. Este trabajo busca entonces un complemento sencillo a la solución existente para el problema de la compresión de imágenes, cuidando la calidad percibida de la misma para producir resultados que sean, visualmente, lo más agradables posible.

**Palabras Clave—** Deep Learning, Convolutional Neural Network (CNNs), Autoencoder, MS-SSIM, Image Quality Assessment.

# I. INTRODUCCIÓN

La compresión de imágenes es una de las aplicaciones de bajo nivel en el procesamiento de imágenes por la cual los investigadores han mostrado un creciente interés, con el objetivo de reducir la dimensionalidad o extraer características significativas [2] que puedan facilitar el procesamiento posterior en tareas de visión artificial, o simplemente realizar la transmisión a bajas tasas de bits.

Si pensamos en el problema de la compresión de imágenes a gran escala, este lo podemos resumir en tres partes donde tenemos un encoder responsable de reducir la imagen de entrada a una representación más compacta, el proceso de binarización el cual genera los códigos binarios, y un decodificador cuya función es realizar la síntesis de la imagen a sus dimensiones originales a partir de los códigos generados. Muchos trabajos se han realizado actualmente para enseñarle a una red neuronal a aprender una representación reducida de una imagen, y finalmente reconstruirla a partir de la misma, siendo estas redes conocidas como autoencoders [1].

Los autoncoders han estado latentes desde hace rato en el panorama de las redes neuronales, siendo hoy en día bastante amplia su clasificación [3], pudiendose encontrar entre ellos los convolutional autoencoders, denoising autoencoders, sparse autoencoders y los variational autoencoders [4], donde estos últimos han recibido un impulso muy grande debido a su utilidad en modelos generativos [5].

El problema principal de estas redes es que operan bajo ciertas restricciones que las hacen inviables para reemplazar a los codecs de imágenes existentes. Algunas de estas limitaciones son la imposibilidad de variar la tasa de compresión del encoder (sin re-entrenar previamente la red) y el hecho de que estas redes se entrenan para una escala en particular, por lo que solo capturan las redundancias presentes a esa escala. En [10], [18] y [19] buscan resolver estas limitaciones obteniendo resultados prometedores, pero ninguno de estos enfocan la optimización de la red desde el punto de vista de la calidad perceptual de la imagen y utilizan métricas que no se correlacionan adecuadamente con este tipo de medidas, las cuales buscan emular el comportamiento del sistema visual humano (HVS).

Por lo tanto, durante el desarrollo de este documento haremos uso de alguna de las arquitecturas presentadas en [10] pero agregando en el cálculo de las perdidas una termino que tenga en cuenta la calidad de la imagen como función de la percepción visual.

# II. Desarrollo

## Convolutional Autoencoder

Como mencionamos, La estructura básica del autoencoder se puede dividir en tres funciones fundamentales donde tenemos el encoder para reducir la representación de la entrada, la función de binarización para incrementar la tasa de compresión al deshacernos de los datos de tipo flotante, y por ultimo encontramos el decoder el cual es responsable de reconstruir la imagen a partir de los códigos. Es entonces que la predicción de la red puede ser vista como el encadenamiento de estas tres funciones base definidas.

(1)

Siguiendo estos principios se construye una red convolucional[6][7] la cual se encontrará separada en tres redes más pequeñas, pero las cuales deben ser entrenadas juntas de punta a punta antes de poder ser utilizadas por separado ya que el criterio de optimización estará basado en reducir el promedio de la diferencia de los cuadrados entre el segmento de entrada y la predicción realizada por el decoder. Entonces lo que se busca en definitiva es minimizar el residuo entre la imagen y su predicción.

(2)

Si abrimos la arquitectura del autoencoder, un encoder consiste en una serie de operaciones convolucionales que se llevan a cabo hasta alcanzar la reducción de las dimensiones buscada. Es de notarse que cuanta más chica es esta representación, más difícil será regenerar la imagen y más información se perderá en el proceso. En el otro extremo del modelo tenemos la red correspondiente al decoder, en donde a diferencia del encoder, las operaciones que se llevan a cabo de forma sucesiva corresponden a deconvoluciones [8], pero que a su vez posee una última etapa correspondiente a una convolución con tres filtros de 1x1 para convertir la decodificación en valores RGB.

La técnica de binarización usada para generar los códigos es aquella propuesta en [17] para simplificar las operaciones de la red, y utilizada en [9] para forzar a la red a aprender representaciones eficientes que no presenten patrones de bits redundantes y no influyan en la salida, las cuales pueden llegar a darse con los datos de tipo punto flotante. Este proceso consta de dos etapas en donde en una primer instancia se obtiene la cantidad de bits deseados por pixel de la representación reducida, pero cuyos valores se encuentran en el intervalo continuo [-1;1]. Esto es realizado mediante una convolución con una ventana de 1x1 y cuya cantidad de filtros dependen de la cantidad de bits deseados por cada pixel. En la segunda parte ocurre la binarización propiamente dicha, donde se toma esta representación en el dominio real y se produce una salida discreta cuyos posibles valores se encuentran en el set {-1 ; 1}. Aquí se utiliza la regularización provista por la cuantización aleatoria la cual nos permite que, durante la etapa de backpropagation, los gradientes pasen por esta etapa de binarización sin sufrir cambios.

(3)

(4)

Esto se debe a que la esperanza E[b(x)] = x para todo x ∈ [-1 ,1], al ser la media del ruido de cuantización nula.

(5)

(6)

Finalmente, para tener una única representación para una entrada en particular, una vez que la red se ha entrenado, para b(x) solamente se considera el valor más probable por lo que no se tiene en cuenta la cuantización aleatoria y se utiliza la función signo para realizar la binarización de los códigos.

Esta arquitectura fue presentada para introducir el bloque básico, junto a las respectivas funciones que se desarrollan dentro de este, y el cual formara parte de un modelo más elaborado que en definitiva constituirá la red utilizada en el presente trabajo.

## Residual Convolutional Autoencoder

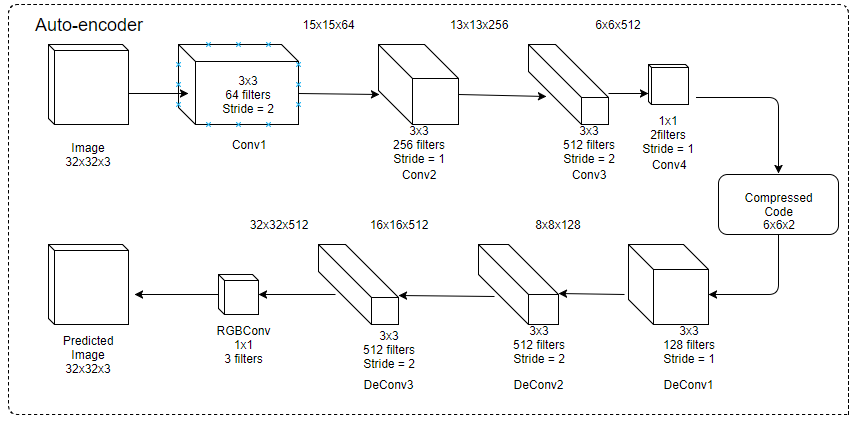
Este modelo es una iteración sobre el bloque ya definido, y extensamente desarrollado en [10] para varias arquitecturas. Sobre una imagen de entrada se encadenan una serie de autoencoder convolucionales con el objeto de obtener una tasa de compresión variable. Esto dará lugar a una relación de compromiso entre la tasa de compresión deseada y la calidad final de la imagen obtenida.

En esta estructura, la salida del primer bloque intenta replicar la entrada mientras que las predicciones de los bloques restantes realizan la predicción de una sucesión de residuos que a su vez oficiaran de entrada para el bloque siguiente en la cadena. De esta forma la reconstrucción de la imagen se llevara a cabo realizando la suma de cada una de las predicciones, o lo que es lo mismo q decir la suma de los residuos.

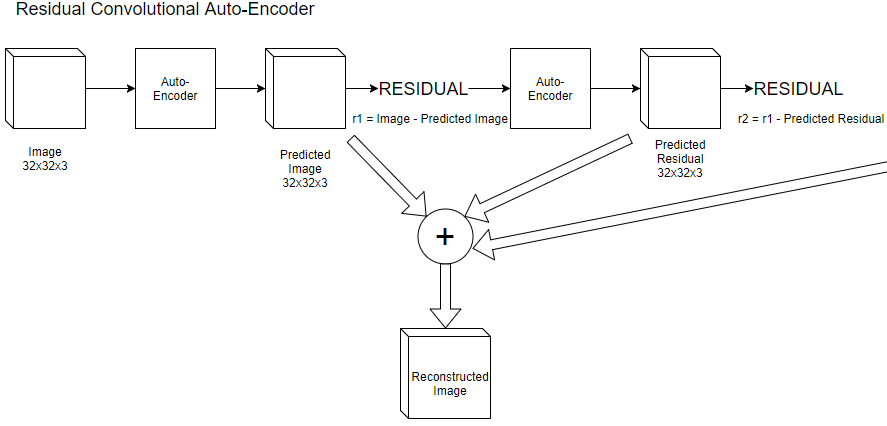
(7)

La optimización de este modelo, según se presenta en [10], se lleva a cabo mediante la reducción del promedio de los cuadrados de los residuos de cada una de las etapas.

(8)



*Fig.1 Arquitectura de una red Autoencoder convolucional. La red está compuesta por 3 capas convolucionales para llevar la imagen de entrada a una expresión reducida de 6x6x2 el cual representa el código resultante de la compresión. Finalmente, con 3 capas deconvolucionales mas una capa convolucional de conversión RGB reconstruimos la imagen a partir del código generado en el cuello de botella de la red.*



*Fig.2 Arquitectura de la red autoencoder convolucional residual. La misma está compuesta por una sucesión en serie de bloques autoencoders convolucionales los cuales van generando los distintos residuos de la imagen. Finalmente la reconstrucción de la imagen se realiza mediante la suma de dichos residuos.*

## Función de Perdidas

Hasta ahora se habló de reducir los cuadrados medios de la diferencia entre la imagen de entrada y su predicción, o de los residuos, ya que esta es una medida ampliamente aceptada debida a su gran cantidad de propiedades deseadas en el contexto de la optimización[11], sumado a la buena correlación que esta medida presenta con respecto a los índices convencionales para la evaluación de la calidad de imagen, como es el PSNR, los cuales operan directamente sobre la intensidad de la imagen.

(9)

Pero la realidad es que el MSE dispone de una serie de limitaciones y suposiciones que hacen que su rendimiento sea pobre respecto a la calidad de imagen según como esta es percibida por el sistema visual humano (HVS) [11][12].

Esto nos da la intuición de que aún hay lugar de mejora para el rendimiento de la red, sin hacer un cambio dramático en la arquitectura de la misma, si adaptamos la función de perdida a algún algoritmo basado en el HSV para minimizar el error de salida y maximizar la calidad de la imagen percibida.

En esta misma dirección de trabajo Hang Zhao [13] propone utilizar el MS-SSIM[15] en conjunto con el error L1, como criterio para reducir las pérdidas para el problema puntual de super-resolución, remoción de artefactos en compresión JPEG y demosaicking. Esta particular elección para la función de pérdidas se debe a que por sí solo el MS-SSIM parece no ser suficientemente sensible a las variaciones uniformes, pudiendo causar cambios de brillo o colores más apagados. Sin embargo Zhao demuestra que el MS-SSIM preserva el contraste, en las regiones de alta frecuencia, mejor que cualquiera de las otras funciones de perdidas con las que experimentan, mientras que el error L1 preserva mejor los colores y la luminancia, ya que el error es pesado de la misma forma sin importar la estructura local de la imagen. Es por esta razón que se combinan ambas funciones de error en una única para preservar las mejores características de ambas:

(10)

(11)

(12)

Donde α es un coeficiente para pesar el aporte de cada una de las pérdidas al error final y es el coeficiente gaussiano con desvío estándar correspondiente a la escala M, donde este último se encuentra multiplicando punto a punto a la diferencia ya que el MS-SSIM propaga el error del pixel q basado en su contribución al índice respecto del pixel central . Sin embargo se hace notar que, en la implementación del MS-SSIM como función de perdidas, realizar el re-scaling sobre todo el batch de las imágenes y predicciones para M escalas distintas en cada iteración, es una operación muy costosa por lo que la forma de resolver este problema es, en su lugar, calcular M filtros Gaussianos con un desvío estándar para cada filtro igual a la mitad del anterior. Al hacer esta aproximación vemos que, para una dada ventana del kernel, al disminuir el del filtro la cantidad de pesos que presentan un valor significativo disminuye por lo que hay una menor cantidad de pixeles tenidos en cuenta en el cálculo de la media del segmento. De esta forma entonces se simula la idea de escalar la imagen, desde el punto de vista que operamos sobre un segmento más grande o más chico con el filtro, según el valor de asignado.

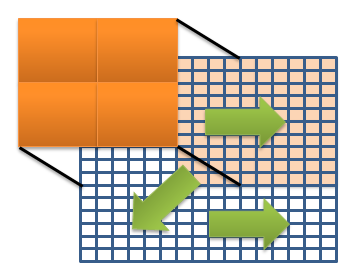




*Fig.3 Pesos de un filtro Gaussiano para una ventana de 9x9 y desvío estándar de0.5 y 0.25*

No obstante, la implementación realizada en [13] implica que previamente al entrenamiento se debe re-escalar las imágenes a una dimensión impar ya que el filtro gaussiano tiene dimensiones impares y el cálculo lo realizan sobre el pixel central para toda la imagen. El problema es que para una imagen de dimensiones 31x31, la ventana del filtro es grande y, por lo tanto, los valores de los píxeles que se encuentran no muy lejos del centro están mucho más atenuados o completamente anulados. Esto reduce considerablemente la contribución que estos pixeles aportan a la estimación del índice. Si bien esta atenuación es un efecto deseado a fines de simular las distintas escalas del filtro, una atenuación excesiva puede anular completamente demasiada información presente en la imagen, por lo cual los valores del del filtro se deben elegir con cuidado.

Es por tal motivo que el trabajo actual considera la función de perdidas propuesta para reducir el error en la regeneración de la imagen, pero encara el problema de una forma diferente. Se obtiene una matriz de 22x22x3 utilizando cuatro filtros gaussianos idénticos de 11x11 repetidos para los tres canales de la imagen. A su vez, esta matriz se convoluciona con la predicción de la red y la salida esperada, cuyas dimensiones son de 32x32x3, usando un stride de 10 pasos para obtener la implementación efectiva de 4 filtros idénticos sobre 4 segmentos diferentes. Esto es realizado para preservar la mayor cantidad de detalles en distintas áreas de la imagen. Por otro lado, la convolución debe ser realizada para cada canal por separado ya que cada uno de estos presenta su propia distribución y, por lo tanto, en la normalización cada canal debe ser afectado por una media y un desvío estándar diferentes. Una vez realizada la convolución, se promedian los resultados para obtener un único valor de por canal, y en última instancia un único valor del índice MS-SSIM. Esta misma matriz es también aplicada en el término de las pérdidas L1 para respetar las contribuciones que hace cada pixel en la estimación de las pérdidas.



*Fig.4 Kernel window conteniendo 4 filtros se desplaza 4 veces sobre la imagen obteniendo 16 estimaciones efectivas sobre cada canal. Finalmente promediamos los resultados para obtener un único valor.*

Finalmente a la función de pérdidas se incorpora el término del error correspondiente a los residuos de cada una de las etapas del autoencoder.

(13)

(14)

Podemos decir entonces que la función de pérdidas que se usara finalmente para entrenar la red tiene dos finalidades, las cuales consisten en reducir el error producido en la generación de cada uno de los residuos en las distintas etapas del autoencoder como también preservar la calidad perceptual de la imagen en su reconstrucción final.

# III. Métrica de Evaluación

Para realizar la evaluación se utilizara el algoritmo SSIM [14], el cual mide la similitud perceptual [16], y como tantos otros índices de similares características, se basa en el sistema visual humano. La razón de su adopción se debe a su amplia aceptación y utilización por lo que facilitara su comparación con trabajos previos o posteriores. Este algoritmo devuelve una puntuación en el rango entre 0 y 1, donde 1 representa la reconstrucción perfecta.

Se realizara la comparación de los resultados obtenidos con las dos arquitecturas detalladas, realizando para todos los casos la optimización de la red con las dos funciones de pérdidas más usadas (L1 y L2) además de la función propuesta en este trabajo. Adicionalmente, para una tasa de compresión dada, se realizara la comparación de la calidad obtenida respecto a aquella obtenida con el códec de compresión JPEG.

La forma de realizar la evaluación consistirá en dividir la imagen en distintos segmentos y calcular el puntaje de cada uno de estos, considerando cada canal RGB por separado. El valor final corresponderá al promedio del SSIM calculado para todos los segmentos y canales.

Para poder realizar la comparación de las tres métricas de optimización bajo las mismas condiciones, lo más sencillo es utilizar un β=0, luego la función de pérdidas resultará:

(15)

(16)

Debido a que lo que buscamos es reducir la perdida de la percepción visual de la calidad de la imagen, no tiene sentido realizar la evaluación del desempeño del PSNR en la comparación de los resultados.

# IV. Resultados

Para las pruebas, experimentamos con una red autoencoder la cual dispone de un cuello de botella cuyas dimensiones son de 6x6x8, permitiéndonos de esta forma controlar la tasa de compresión en pasos de 288 bits. El objetivo de este trabajo es comparar la métrica introducida, para disminuir las pérdidas en función de la calidad visual, respecto a las funciones de perdidas más conocidas y ampliamente utilizadas como son el error cuadrático medio (L2) y el error absoluto medio (L2). Para esto se entrenaron tres modelos distintos, donde en cada uno de ellos se integraron 12 etapas y la salida corresponde a la sumatoria de los residuos obtenidos en ellos. Finalmente se realizara la optimización de la red siguiendo la ec. 14, donde y tomará el valor de cada una de las funciones de la ec. 16. En el caso del , el es aquel utilizado en [13].

Para el entrenamiento se empleó la base de datos CIFAR-10 la cual consiste en 60k imágenes, de dimensiones 32x32, de las cuales se utilizaron 50k para entrenar la red y las 10k restante se usaron para realizar el benchmark de los resultados. El learning rate utilizado es de 0.0005, se usó un batch size de 64 y se realizó el entrenamiento durante 300 epochs. Si bien estos valores no son los más óptimos, la memoria de las dos GPUs disponibles nos imposibilito trabajar con batchs más grandes en el entrenamiento como tampoco fue posible utilizar una mayor cantidad de autoencoders en cascada, reduciendo la cantidad de canales del cuello de botella y así poder controlar la tasa de compresión con pasos más chicos.

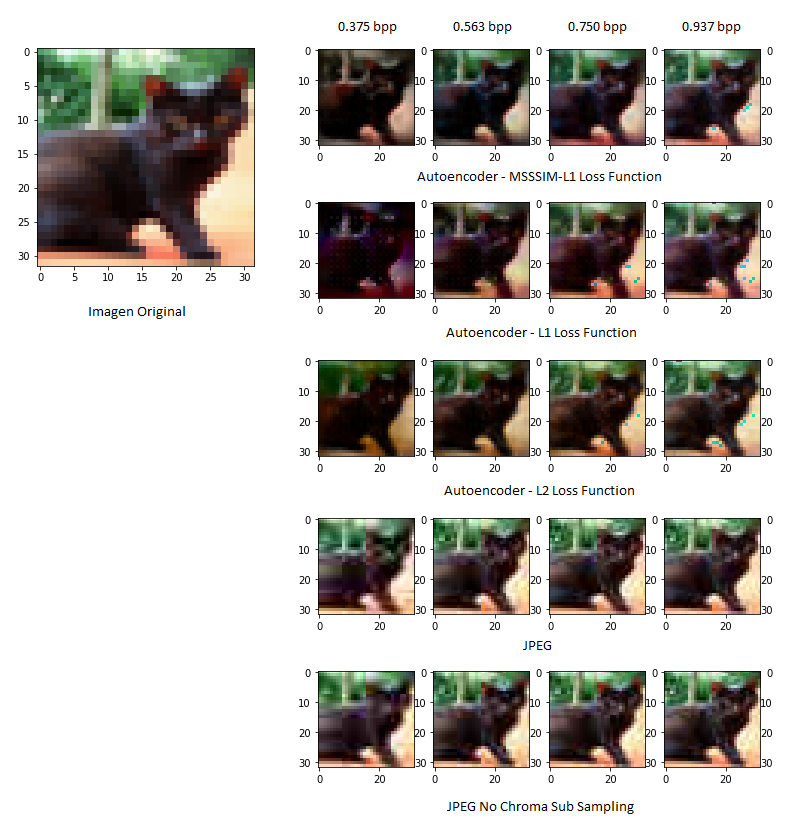


*Fig.5 Decodificación de 10 imágenes del set de testeo para distintos codecs, habiendo realizado una compresión a 0.75bpp.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SSIM / 0.75bbps | JPEG | L1 Autoencoder | L2 Autoencoder | MS-SSIM Autoencoder |
| Img. 1 | 0.8988 | 0.8970 | 0.8738 | 0.9339 |
| Img. 2 | 0.9507 | 0.8954 | 0.9092 | 0.9195 |
| Img. 3 | 0.9347 | 0.9012 | 0.8997 | 0.9070 |
| Img. 4 | 0.9515 | 0.8636 | 0.8757 | 0.8955 |
| Img. 5 | 0.9113 | 0.8698 | 0.8754 | 0.9100 |
| Img. 6 | 0.9564 | 0.8825 | 0.9054 | 0.9135 |
| Img. 7 | 0.8927 | 0.8262 | 0.8113 | 0.8620 |
| Img. 8 | 0.9190 | 0.8783 | 0.8788 | 0.9098 |
| Img. 9 | 0.9104 | 0.7769 | 0.7824 | 0.8584 |
| Img. 10 | 0.8061 | 0.7795 | 0.7816 | 0.8527 |

*Tabla 1. La siguiente tabla presenta los resultados obtenidos para 10 imágenes del set de testeo, comparando los valores obtenidos por los tres modelos y las mismas imágenes comprimidas en JPEG. Para estas dimensiones de imágenes, el encabezado de la compresión JPEG es considerable por lo que estos fueron eliminados antes de realizar las respectivas comparaciones.*

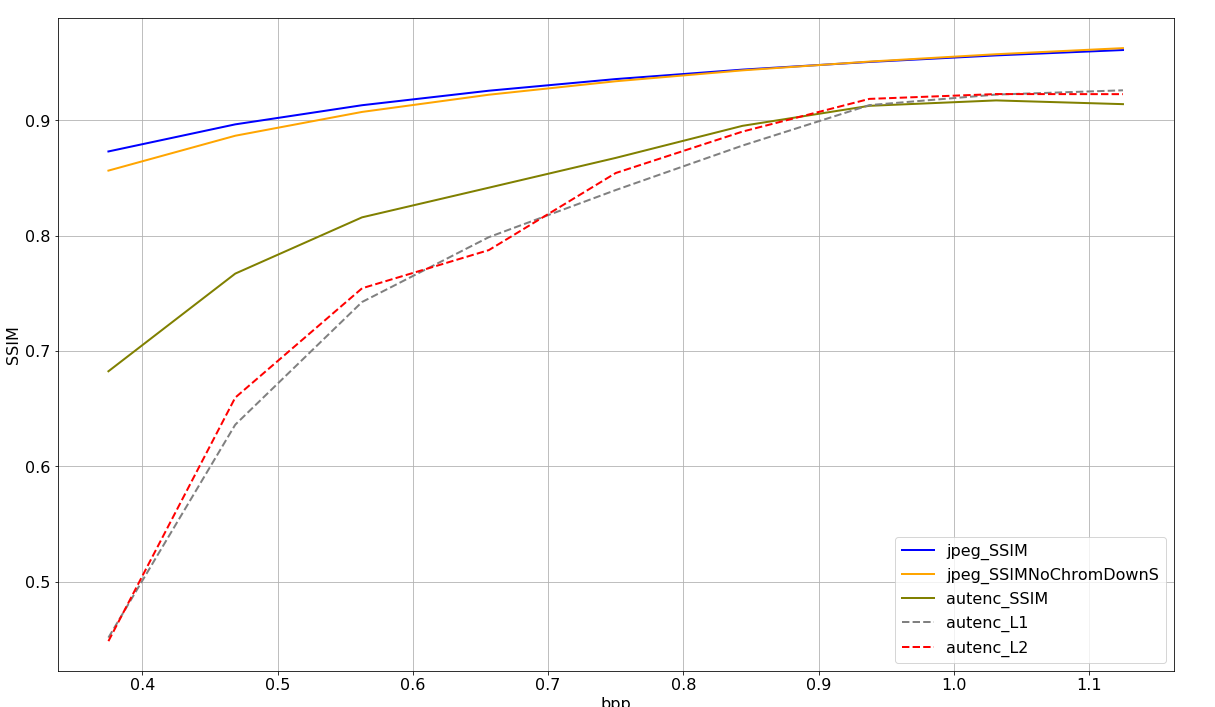
No todas las imágenes tienen la misma distribución por lo que los resultados serán distintos para cada una de estas. Esto hace que la compresión JPEG no sea la misma para una calidad dada, por lo tanto se debe buscar en cada imagen el parámetro de calidad que produce una tasa de compresión igual a la producida por la red. Si nos ponemos a analizar el set de imágenes podemos ver como la compresión JPEG suele ser la dominante en términos de resultados pero en lo que respecta a las imágenes producidas por los autoencoders, aquella que fue entrenada con la función de MSSSIM-L1 suele producir una mejor conservación de los colores y una disminución considerable de los blocking artifacts presentes. Respecto a este último punto, si observamos las imágenes 3, 6, 9 y 10, aun en los casos donde la compresión JPEG tuvo un valor de SSIM más alto, la cantidad de artefactos presentes es menor en la imagen producida por la red autoencoder entrenada con esta métrica.



*Fig.7 Aperturta de la imagen 8 a distintas tasas de compresión para 5 modelos distintos.*

La imagen de la figura 7 es de especial interés analizar. En ella se puede apreciar como si bien las imágenes JPEG obtuvieron una mejor evaluación según el índice SSIM, podemos decir que esto corresponde prácticamente a la variación en la luminancia de las imágenes. En términos generales los resultados observados concuerdan con lo presentado en [13], donde la red cuya función de perdidas utilizadas es la MSSSIM+L1 presenta colores un poco más apagados y a su vez, si miramos con detalle la región verde que se encuentra en la parte superior de la imagen, las variaciones en las tonalidades parecen ser más uniformes respecto al patrón que presenta la imagen original. Esto último probablemente se deba a la influencia de la función L1, el mismo efecto se pudo apreciar en los modelos que comparten esta función de pérdidas. Probablemente esto pueda corregirse si se ajustan los valores del y a su vez hacemos distinto de 0 para penalizar un poco más los errores en las regiones donde hay una variación más significativa de la intensidad. Más allá de estas consideraciones, podemos destacar que la función propuesta logra disminuir considerablemente los blocking artifacts propios de la compresión JPEG como también conserva mejor otros aspectos morfológicos del animal como son las extremidades rojas en los extremos de las orejas y el rostro en general.

En el siguiente grafico presentamos el valor obtenido del SSIM para un determinado nivel de compresión o mejor dicho bits por pixeles resultantes. Estos valores fueron obtenidos a partir de promediar los resultados obtenidos para las 10k de imágenes presentes en el set de testeo.



*Fig.7 SSIM para distintos codecs a diferentes bits rates. La curva se realizó tomando el promedio de las 10k imágenes del set de testeo para cada nivel de compresión determinado.*

Se recuerda que este nivel de compresión, para la red dada, dependerá de la cantidad de bloques residuales que se integren en la reconstrucción de la imagen. En la curva podemos ver como a medida que la cantidad de bbps aumentan o mejor dicho la cantidad de residuos que incorporamos aumentan, los resultados para los distintos autoencoders se van juntando ya que los errores presentes son cada vez más insignificantes y la diferencia de calidad ya no es tan perceptible. En definitiva, surge naturalmente la necesidad de evaluar cuantos bloques significativos hacen realmente falta para recuperar una imagen con una determinada calidad visual sin incrementar innecesariamente la cantidad de bbps, ya que el aporte de los últimos bloques residuales puede resultar insignificante frente al incremento marginal de la tasa de compresión. Este mismo efecto puede observarse durante el entrenamiento de la red donde, si buscamos reducir las pérdidas de cada uno de los residuos, vemos como el descenso en el error se va haciendo despreciable a medida que nos alejamos de los bloques iniciales y esto tiene que ver con que el error tiende a hacerse nulo cuando la cantidad de residuos considerados tiende a infinito. Determinar la cantidad optima de bloques para un determinado nivel de calidad excede los objetivos del presente trabajo pero lo que podemos decir es que esta cantidad está íntimamente ligada con la cantidad de canales como también las dimensiones de la matriz resultante de la etapa de binarización de la red, siendo que a medida que se disponga de una menor cantidad de canales o bits para codificar un segmento de una imagen, se requerirá entonces una mayor cantidad de bloques o residuos para reconstruirla con una determinada calidad. Esto se debe a que el error en la codificación tiende a incrementarse y por lo tanto aumenta el valor de los residuos. De la misma forma, a medida que aumentamos la cantidad de canales presentes en la matriz de codificación, la cantidad de bloques necesarios para alcanzar una determinada calidad será más chica.

# V. Conclusiones

El objetivo principal de este trabajo era demostrar como utilizando métricas más cercanas a la calidad visual como criterio de optimización de la red, se pueden lograr mejores resultados a tasas de compresión más bajas respecto a aquellos logrados con las funciones de perdidas tradicionales. En la figura x se puede apreciar como al utilizar el índice MS-SSIM en el entrenamiento de la red, los resultados obtenidos se encuentran entre 5 y 10 puntos porcentuales por encima de los alcanzados con los modelos donde se usaron las funciones L1 y el L2.

 Por otro lado es evidente como a medida q se incorporan más etapas en la reconstrucción de la imagen, y por lo tanto empeorando la tasa de compresión, esta brecha se disminuye debido a la reducción considerable del error.

Cabe destacar que en este trabajo no se realizó una exploración profunda de los híper-parámetros involucrados en el entrenamiento, limitándonos a utilizar, en algunos casos, los valores cercanos a los óptimos reportados en otros papers, y en otros valores que nos permitieran realizar una comparación justa cuidando los tiempos necesarios para entrenar las redes. Por tal motivo mejoras significativas podrían surgir de la optimización de los mismos.

Lo mismo podemos decir respecto a la arquitectura de la red, en nuestro caso se entrenó una red con 12 bloques y 8 canales en la matriz de codificación para realizar la evaluación de los resultados, pero resulta conveniente entrenar una red con una mayor cantidad de bloques y una menor cantidad de canales en el cuello de botella de cada uno de los autoencoders, disponiendo de esta forma de pasos discretos más chicos para manejar la tasa de compresión de la red. La contrapartida de realizar esto es que a medida que aumentamos la cantidad de etapas en nuestra red, la cantidad de parámetros a entrenar también lo hace en forma proporcional.

En [9] Se puede observar como obtienen excelentes resultados con una red residual 16 pasos, donde las dimensiones de la matriz de codificación de cada bloque son 8x8x2.

# VI. Referencias

1. *Using very deep autoencoders for content-based image retrieval*. Krizhevsky, A. and Hinton, G. E., pp. 1-7, Department of Computer Science, University of Toronto, Jan. 2011.
2. *Transforming Auto-Encoders*. Krizhevsky, A., Hinton, G. E. and S. D. Wang , pp. 1-8, Department of Computer Science, University of Toronto, Jun. 2011.
3. *Deep Learning*. Ian Goodfellow, Yoshua Bengio and Aaron Courville, pp. 500-523, MIT Press, Nov. 2016. https://www.deeplearningbook.org/contents/autoencoders.html
4. *Tutorial on Variational Autoencoders*. Carl Doersch, pp. 1-23, arXiv preprint arXiv:1606.05908v2, Aug. 2016.
5. *A Review of Learning with Deep Generative Models from perspective of graphical modeling*. Zhijian Ou, pp. 5-29, arXiv preprint arXiv:1808.01630v3, Sep. 2018.
6. *Gradient-based learning applied to document recognition*. Yan LeCun, L. Bottou, Y. Bengio and P. Haffner, pp. 5-11, IEEE, Nov. 1998.
7. *An Introduction to Convolutional Neural Networks*. Keiron O’Shea and Ryan Nash, pp. 4-10, arXiv preprint arXiv:1511.08458v2, Dec. 2015
8. *A guide to convolution arithmetic for deep learning*. Vincent Dumoulin and Francesco Visin, pp. 1-28, arXiv preprint arXiv:1603.07285v1, Mar. 2016.
9. *Variable Rate Image Compression with Recurrent Neural Networks*. George Toderici and Sean M. O’Malley, pp. 4, arXiv preprint arXiv:1511.06085v5, Mar. 2016.
10. *Variable Rate Image Compression with Recurrent Neural Networks*. George Toderici and Sean M. O’Malley, pp. 3-6, arXiv preprint arXiv:1511.06085v5, Mar. 2016.
11. *Mean squared error: Love it or leave it? A new look at Signal Fidelity Measures*. Zhou Wang and Alan C. Bovik, pp. 1-4, IEEE, Jan. 2009.
12. *A comprehensive evaluation of full reference image quality assessment Algorithms.* Zhang, L., Zhang, L., Mou, X., Zhang, D., pp. 1-3, IEEE International Conference on Image Processing. 2012
13. *Loss Functions for Image Restoration With Neural Networks.* Hang Zhao, Orazio Gallo, Iuri Frosio and Jan Kautz, pp. 1-6, arXiv preprint arXiv:1511.08861v3, Apr. 2018.
14. *Image Quality Assessment: From Error Visibility to Structural Similarity.* Zhou Wang and Alan C. Bovik, pp. 602-607, IEEE Transactions on Image Processing, Vol. 13, No. 4, Apr. 2004.
15. *Multi-Scale Structural Similarity For Image Quality Assessment.* Zhou Wang, Eero P. Simoncelli and Alan C. Bovik, pp. 1-3, IEEE, Nov. 2003.
16. *Perceptual Visual Quality Metrics: A survey*. Weisi Lin and C. –C. Jay Cuo, pp. 7-10, Elsevier, Jan. 2011.
17. *BinaryConnect: Training deep neural networks with binary weights during propagations*. Courbariaux, M., Bengio, Y., and David, J.-P. pp. 1-4, arXiv preprint arXiv:1511.00363v3, Nov. 2015.
18. *Full ResolutionImage Compression with RecurrentNeural Networks*. George Toderici, Damient Vincent, Nick Johnston, Sung Jin Hwang, David Minnen, Joel Shor and Michele Cowell, pp. 1-6, arXiv preprint arXiv:1608.05148v2, Jul. 2017.
19. *Deep Convolutional AutoEncoder-based Lossy Image Compression*, Zhengxue, Heming Sun, Masaru Takeuchi and Jiro Katto, pp. 1-6, arXiv preprint arXiv:1804.09535v1, Apr. 2018.