Estudio de arquitecturas de Autoencoders, aplicados a la generación de Espectrogramas ficticios de Ballenas Barbadas.

Marco Carnaghi y María C. Cebedio
ICYTE, Depto. de Electrónica y Computación, Facultad de Ingeniería - UNMDP
Mar del Plata, 7600, Argentina
{mcarnaghi,celestecebedio}@fi.mdp.edu.ar

***Resumen*—En este trabajo se presenta la síntesis de espectrogramas de registros de ballenas barbadas, a partir de utilizar Autoencoders Convolucionales simples. El estudio abarca el acondicionamiento de los datos, el diseño de diferentes arquitecturas, el entrenamiento y la evaluación, el análisis de la mejor opción y la generación de los espectrogramas ficticios. Los resultados obtenidos demuestran que es posible generar espectrogramas de elevada correspondencia con sonidos provenientes de ballenas barbadas, a partir de una arquitectura de pocas capas convolucionales.**

***Palabras Clave*—Autoencoders convolucionales, espectrogramas, sonidos subcuáticos, síntesis.**

I. Introducción

El estudio de los registros vocales de los mamíferos subacuáticos es un área de interés e importancia biológica. Los sonidos que emiten representan el lenguaje natural de estas especies y reflejan su comportamiento. En la actualidad, el creciente interés por la conservación de las especies marinas ha propiciado un avance científico en esta área. En este sentido, las nuevas técnicas de Machine Learning resultan prometedoras y se han desarrollado trabajos muy interesantes [1], [2], [3]. Prueba de esto, en [4] se utilizan para detectar llamadas de ballenas en peligro de extinción. La ausencia del volumen requerido de datos de audio submarino, es uno de los principales problemas que se enfrentan. Esta baja disponibilidad de recursos condiciona en gran medida la utilización de técnicas, para las cuales la cantidad y calidad de los datos juega un papel central.

La generación automática de espectrogramas ficticios con las características de una señal de audio de interés puede ser beneficioso para expandir la cantidad de datos. Estos datos, generados artificialmente, podrían solucionar problemas de desbalanceo en el conjunto de entrenamiento [5].

Investigaciones actuales han desarrollado modelos generativos basados en técnicas de aprendizaje profundo no supervisado, como el Autoencoder Convolucional (AE-CNN) y el Autoencoder Variacional (VAE), para reconstruir señales. Estos Autoencoders (AE) se caracterizan por generar una representación compacta de los datos de entrada denominada "espacio latente", la cual es posteriormente utilizada para generar imágenes de salida similares a las de entrada. En este sentido, el análisis realizado y los resultados obtenidos en [6] para la síntesis de la risa pueden ampliarse y aplicarse a sonidos proveniente de fuentes biológicas.

Por otro lado, la generación ficticia de espectrogramas en tiempo real puede ser útil en bancos de prueba destinado al análisis de sonido submarino. En este sentido, sería de mayor utilidad aún, contar con un equipamiento dedicado a la generación de espectrogramas de manera de independizarse de la PC. Previo a una implementación física sobre algún sistema embebido, es necesario desarrollar un modelo de arquitectura simple y de pocas capas, de manera que el costo computacional asociado a las operaciones matemáticas y a la cantidad de parámetros se reduzca. Trabajos en el área han demostrado que con grandes redes convolucionales se pueden obtener excelentes resultados, pero estas redes poseen una gran cantidad de parámetros y las operaciones matemáticas se incrementan con cada capa que se añade. El problema reside actualmente en minimizar el modelo de red para que luego puedan ser implementados en sistemas embebidos.

En este trabajo se propone el análisis y evaluación de diferentes arquitecturas sencillas de AE, para el proceso de síntesis de espectrogramas de sonidos de ballenas barbadas. El fuerte hincapié en obtener un modelo simple radica en la posible implementación del mismo sobre sistemas que disponen de una potencia computacional muy limitada. Esta limitación no es un gran impedimento en redes neuronales sencillas sin mucha complejidad o sin muchos cálculos, pero en redes más complejas sí.

II. Metodología

La metodología de síntesis consiste en transformar el espectrograma logarítmico de alta dimensión, en un vector latente de baja dimensión. Este vector contiene información valiosa que se utiliza para reconstruir un espectrograma de magnitudes sintéticas. El proceso se desarrolla siguiendo los siguientes pasos: adecuación de los datos de entrada, entrenamiento del modelo y obtención de métricas, selección del mejor modelo a entrenar y generación de espectrogramas ficticios. Los pasos enumerados anteriormente no son estáticos. La selección de características y evaluación es iterativa.

II-A. Adecuación de los datos de entrada

El objetivo de esta etapa es obtener una matriz bidimensional que represente el espectrograma de magnitud logarítmica para cada registro de audio. Esta representación matricial de la energía del contenido frecuencial de una señal puede pensarse como una imagen en escala de grises.

El conjunto de todos los espectrogramas normalizados, correspondiente a registros de audio de diferentes tipos de ballenas barbadas, se agrupa en un tensor. Originalmente, estos datos provienen de diferentes fuentes y para su tratamiento se normalizan bajo las siguientes características: Frecuencia de remuestreo= 44.1KHz, Duración temporal del registro de audio=1s, Tipo de ventana para espectrogramas=Tukey, Cantidad de puntos por bloque para STFT =256, Normalización de Amplitud: -150 a 150 dB.

Para automatizar el proceso de normalización, se diseña un programa [7] que permite customizar estas características. De esta manera se obtiene un tensor de 3 dimensiones (Sx,Sy,N), siendo 'N' la cantidad de muestras de 1 seg., Sx y Sy los valores de la matriz de espectrograma. En este punto, los datos aún no son adecuados para el entrenamiento de los AE. Por lo tanto, se realiza: la adecuación de la dimensión de los datos de entrada a (N,Sx,Sy,1); la división de los datos en entrenamiento (D_{Train}), validación (D_{Val}) y test (D_{Test}); y una normalización entre 0 y 1.

II-B. Arquitecturas de Autoencoders

Se generan dos tipos de arquitecturas: AE-CNN y VAE. Como se muestra en la Fig. 1, el AE-CNN posee una sección de codificación basada en redes convolucionales, en la que se realiza una sección de extracción de características, una capa flatten (utilizada para uni-dimensionalizar la entrada) y una última red densamente conectada. A la salida se obtiene un vector representativo (espacio latente), a partir del cual y replicando a la inversa el proceso de codificación se intenta recuperar la imagen de entrada.

La arquitecturas VAE es similar al AE-CNN con la diferencia de la capa de parámetros probabilísticos después de la capa Flatten [8]. Por lo tanto, en lugar de aprender directamente las características latentes de las muestras de entrada, aprenden la distribución de las características latentes. Los VAE aseguran que los puntos que están muy cerca uno del otro en el espacio latente representan muestras de datos muy similares [9].

Las arquitecturas propuestas poseen en la etapa de extracción de características 4 capas. En todas las etapas la función de activación es tipo RELU y el Padding (asociado a la etapa de filtrado) se re-acomoda de forma automática. Se realiza la operación de Batch Normalization como capa Pooling. Esta capa acelera el proceso de entrenamiento, mejora las propiedades de normalización de la red y la vuelve más robusta a diferentes esquemas de inicialización y tasas de aprendizaje [10]. Resumiendo la arquitectura empleada: 1° capa: 32 Filtros de 3x3x1 con stride= 1; 2° capa: 64 filtros de 3x3x1 con stride= 2; 3° capa: 64 filtros de 3x3x1 con stride= 2; 4° capa: 64 filtros de 3x3x1 con stride= 1.

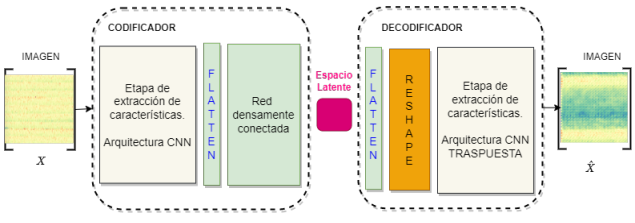


Fig. 1: Arquitectura de un Autoencoder CNN genérico.

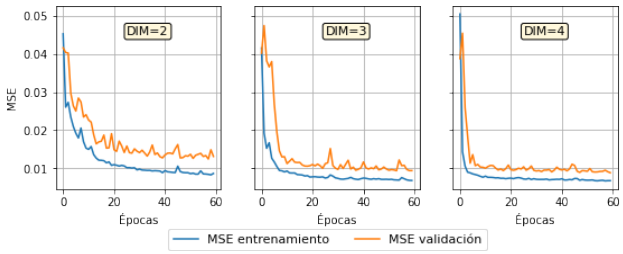
Para el caso del AE-CNN, se diseña una sola capa densa en la sección de capa profunda y para el VAE dicha capa está representada por capas densamente conectadas de varianza y media. Respecto a la dimensión del espacio latente, esta es una consideración de diseño que hace a la arquitectura de la red. En este caso se decide obtener diferentes arquitecturas, a partir de cambiar únicamente esta dimensión.

II-C. Entrenamiento

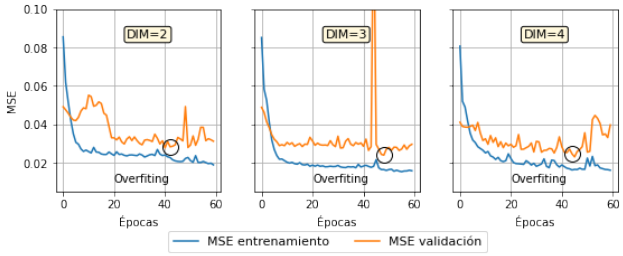
El objetivo de entrenar la red es obtener un Autoencoder que reconstruya el espectrograma a su entrada. Los Hiperparámetros asociados al entrenamiento se establecen en: función de pérdida=MSE; optimizador=Adam; taza de aprendizaje: 0.0005; épocas=60(CNN) y 48(VAE); tamaño de Minibatch=100; D_{Train}=6043 con $\%D_{Val}$=0.2.

II-D. Elección de las arquitecturas

Para la elección de la dimensión del espacio latente, se entrenan la AE-CNN y la VAE, con el set de datos correspondiente a las ballenas Barbadas, variando la dimensión entre tres valores posibles: 2, 3 y 4. A partir del análisis de los resultados del entrenamiento que se muestran en la Fig. 2, se seleccionan las arquitecturas AE-CNN con D=4 y VAE con D=3.



(a) AE-CNN



(b) VAE

Fig. 2: Datos de Entrenamiento para arquitecturas AE-CNN y VAE con diferentes dimensión del espacio latente.

En la Fig. 3 se observa que, aún en el mejor caso de VAE, AE-CNN posee un orden de magnitud menos de error y éste cae notablemente con pocas épocas de entrenamiento.

El MSE es un indicativo de buen desempeño, pero es conveniente evaluar con una imagen de Test para corroborar el significado de dicho valor. Además, a modo de obtener otro parámetro de medición, se calcula el error de similitud estructural (SSIM), que ofrece buena precisión de evaluación y simple formulación [11]. En la Fig. 4 se presentan los resultados, indicando que el mejor modelo a implementar es el AE-CNN.

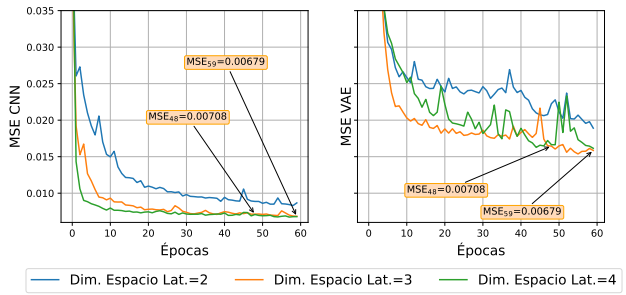


Fig. 3: MSE de entrenamiento para diferentes dimensiones del espacio latente en las arquitecturas AE-CNN y VAE.

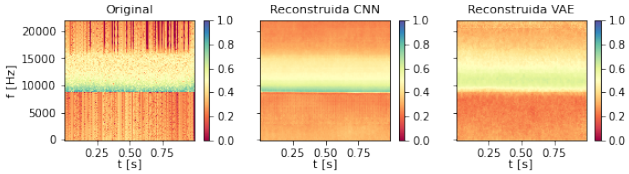


Fig. 4: Imagen reconstruída utilizando un AE-CNN y VAE. Datos: MSE_{CNN}=0.0052, $SSIM_{CNN}$=0.580, MSE_{VAE}=0.011, $SSIM_{VAE}$= 0.497.

III. Generación de espectrogramas ficticios

Para generar espectrogramas ficticios se utiliza el bloque decodificador del AE-CNN. El problema radica en definir los valores del espacio latente de entrada. Con el fin de obtener muestras representativas, se realizan los siguientes pasos:

1. Con los vectores de espacio latente generados en el entrenamiento, se obtiene una matriz de dim.[N_{Train},4].
2. Se analiza la interdependencia entre los datos de cada vector de espacio latente (fila de la matriz). En 1, la matriz de correlación muestra que no existe una marcada relación intra-vector.

$$\begin{bmatrix} 1 & 0{,}125 & -0{,}071 & -0{,}074 \\ 0{,}125 & 1 & 0{,}073 & 0{,}143 \\ -0{,}071 & 0{,}073 & 1 & -0{,}142 \\ -0{,}074 & 0{,}143 & -0{,}142 & 1 \end{bmatrix} \quad (1)$$

3. Se obtiene el valor medio y la desviación estándar de la distribución de los valores de cada vector.
4. Se genera un vector de código aleatorio [X_0,X_1, X_2,X_3] siguiendo una distribución Gaussiana con la media y desviación estándar encontradas en el paso 2.

De esta manera se busca que el vector generado sea representativo.

Las Figs. 5 y 6 muestran ejemplos de imágenes reales y sintetizadas, con los modelos entrenados en [12].

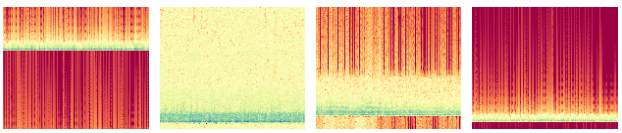


Fig. 5: Espectrogramas Reales, obtenidos a partir de registros aleatorios de ballenas Barbadas.

IV. Conclusión y trabajo a futuro

En esta trabajo se demuestra la posibilidad de generar registros de espectrograma ficticios, a partir de un codificador automático reducido y a partir de pocas muestras. Se

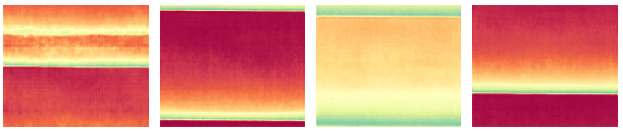


Fig. 6: Espectrogramas sintetizados aleatoriamente, obtenidos con un decodificador entrenado, a partir de registros de ballenas Barbadas.

En ambos casos, se generan imágenes aleatorias, no hay correspondencia entre ellas.

observa que un modelo general simple permite aprender características latentes efectivas para diferentes espectrogramas de sonidos subacuático de baja frecuencia.

En general, las implementaciones de redes profundas se realiza sobre CPU's debido a la gran carga computacional que conlleva realizar la cantidad de operaciones asociadas a cada capa de la red. Un modelo simple como el mostrado puede ser entrenado en una PC personal (sin recurrir a GPU o entrenamiento en la nube) y permite, a futuro posibles implementación sobre sistemas embebidos que posean limitada potencia computacional.

Dado que el modelo propuesto posee una cantidad reducida de capas convolucionales y de parámetros asociados, se reducen las operaciones involucradas y los errores asociados a ellas debido a la limitación física en la cantidad de bits que presentan las arquitecturas de sistemas embebidos simples.

V. Agradecimientos

Al Dr. Diego Comas y al Dr. Gustavo Meshino por los conocimientos impartidos sobre la temática.

Referencias

[1] E. Tejero, "Aplicaciones de Machine Learning a la Bioacústica Marina," Ph.D. dissertation, 07 2020.

[2] D. Tuia and E. Al, "Perspectives in machine learning for wildlife conservation," *Nature Communications*, vol. 13, no. 792, 2022.

[3] A. Lamba, P. Cassey, R. Raja Segaran, and L. Koh, "Deep learning for environmental conservation," *Current Biology*, vol. 29, pp. R977–R982, 10 2019.

[4] A. Ibrahim and et. al, "A multimodel deep learning algorithm to detect North Atlantic right whale up-calls," *The Journal of the Acoustical Society of America*, vol. 150, 08 2021.

[5] H. Ali, M. Salleh, R. Saedudin, K. Hussain, and M. Mushtaq, "Imbalance class problems in data mining: A review," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 14, 03 2019.

[6] N. Mansouri and Z. Lachiri, "Laughter synthesis: A comparison between Variational autoencoder and Autoencoder," in *5th International Conference on Advanced Technologies for Signal and Image Processing (ATSIP)*, 2020, pp. 1–6.

[7] M. C. Cebedio and M. Carnaghi, "Matriz de Espectrogramas," GitHub, 2022. [Online]. Available: https://acortar.link/MKnoYk

[8] Q. Xu, Z. Wu, Y. Yang, and L. Zhang, "The difference learning of hidden layer between autoencoder and variational autoencoder," in *29th Chinese Control And Decision Conference*, 2017, pp. 4801–4804.

[9] C. Doersch, "Tutorial on Variational Autoencoders," 2016. [Online]. Available: https://arxiv.org/abs/1606.05908

[10] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," 2015. [Online]. Available: https://arxiv.org/abs/1502.03167

[11] A. León-Batallas, J. Bermeo-Paucar, Paredes-Quevedo, and H. Torres-Ordoñez, "Una revisión de las métricas aplicadas en el procesamiento de imágenes," *RECIMUNDO*, pp. 267–273, 2020.

[12] M. Carnaghi and M. C. Cebedio, "Autoencoders," GitHub, 2022. [Online]. Available: https://acortar.link/YgWCOY