elektron ámbar

Original Article

Revista elektron, Vol. 6, No. 2, pp. 129-134 (2022)
https://doi.org/10.37537/rev.elektron.6.2.167.2022

Espectrogramas de registros de Ballenas Barbadas sintetizados a partir de arquitecturas de Autoenconders: CAE, VAE y CAE-LSTM

Spectrograms of baleen whale records synthesized from Autoenconder architectures: CAE, VAE and CAE-LSTM

María Celeste Cebedio[†1], Marco Carnaghi[†2]

[†]*ICYTE, Depto. de Electrónica y Computación, Facultad de Ingeniería - UNMDP*
Mar del Plata, 7600, Argentina
[1]celestecebedio@fi.mdp.edu.ar
[2]mcarnaghi@fi.mdp.edu.ar

Recibido: 30/10/22; Aceptado: 30/11/22

***Resumen*—En este trabajo se analizan diferentes arquitecturas de redes convolucionales sencillas para generar espectrogramas sintéticos correspondientes a registros de audio de ballenas barbadas.**

La sencillez en el modelo juega un rol importante en las implementaciones de este tipo de redes sobre sistemas embebidos. Además, existe una necesidad de generar modelos eficientes frente a la escasez de datos disponibles para este tipo de aplicaciones. Con tal fin, se presentan arquitecturas de Autoencoders simples y de baja cantidad de parámetros asociados, se entrenan los modelos, se obtienen métricas adecuadas y se realizan las correspondientes comparaciones.

Los resultados obtenidos demuestran que la arquitectura con una implementación más directa es, a su vez, la más conveniente. Finalmente, a partir de estos modelos, se generan espectrogramas sintéticos a partir de pocos datos de muestra, empleando una arquitectura de baja complejidad y asumiendo una distribución normal de los vectores reales.

***Palabras clave*—: Autoencoders convolucionales; Capas recursivas; espectrogramas; sonidos subcuáticos; síntesis.**

***Abstract*— In this paper, different architectures of simple convolutional networks are analyzed to generate synthetic spectrograms corresponding to baleen whales.**

Simplicity in these models plays an important role in the implementations of these type of networks on embedded systems. In addition, the scarcity of available data requires the generation of efficient models. With this aim in mind, simple Autoencoder architectures with a low number of associated parameters are presented and trained in this paper. Then, adequate metrics are obtained and the corresponding comparison among the architecture alternatives is made.

The obtained results show that the more straightforward architecture is, in turn, the most convenient. Finally, from these models, synthetic spectrograms are generated from few data samples are generated, employing a low complexity architecture and assuming a normal distribution of the latent space vectors from the training data.

***Keywords*—: Convolutional autoencoders; recursive layers; spectrograms; underwater sound; synthesis.**

I. Introducción

Las iniciativas por la preservación de especies marinas han actuado como disparador de numerosas investigaciones asociadas a dicha problemática [1], [2]. En particular, uno de los ejes temáticos se relaciona con el análisis de registros de audio provenientes de mamíferos subacuáticos. Estos registros son de interés debido a que brindan información respecto a los comportamientos y mecanismos de comunicación empleados por tales especies.

Dada la complejidad que trae asociada el análisis de tales datos, su correlación en largos períodos de tiempo y la contaminación de la información disponible con ruido en el entorno submarino; las técnicas basadas en inteligencia artificial han surgido como una alternativa atractiva para llevar a cabo los estudios mencionados [3], [4], [5]. En Ibrahim [6], las técnicas mencionadas se utilizan para detectar llamadas de ballenas en peligro de extinción.

Sin embargo, existe un volumen reducido de datos disponibles, lo cual impacta en forma directa sobre la calidad de los modelos que pueden lograrse mediante técnicas basadas en datos, condicionando en gran medida la utilización de estos. Una potencial solución es la generación automática de registros de audio en forma artificial, donde las muestras artificiales o sintéticas conserven las principales características de las muestra originales. De esta forma, se puede expandir la cantidad de datos disponibles y mitigar los problemas asociados al desbalanceo de los conjuntos de entrenamiento que enfrentan otros modelos.

Estudios actuales desarrollan modelos generativos basados en técnicas de aprendizaje profundo, como el Autoencoder Convolucional (CAE) y el Autoencoder Variacional (VAE), para reconstruir señales [7], [8], [9], [10]. Estos Autoencoders (AE) generan una representación reducida o codificada de los datos de entrada denominada espacio latente y se entrenan para que la salida imite a la entrada. Este espacio latente se utiliza posteriormente para generar imágenes de salida que conservan las características de las imágenes de entrenamiento sin ser idénticas. Los Autoencoders son especialmente atractivos para la generación de datos sintéticos debido a que en su entrenamiento no se consiguen réplicas exactas. Estas características que han llevado a la adopción de otros modelos para recrear los datos de entrada es, en estas situaciones, aprovechada para la síntesis de nuevos datos.

Los modelos de redes convolucionales, que pueden emplearse en la etapa de extracción de características de los Autoencoders, han demostrado presentar excelentes resultados pero poseen el inconveniente de ser arquitecturas complejas. Cuando se dispone de una PC de alto rendimiento esto no representa un problema, pero es una limitante importante para la aplicación de estos modelos sobre sistemas de bajo rendimiento. A modo de ejemplo, en el caso de la red AlexNet (Una red ampliamente contrastada) presenta 62,3 millones de parámetros [11], lo que implica igual orden de operaciones aritméticas y necesidad de trabajar en paralelo con unidades de procesamiento gráfico.

La complejidad asociada a cualquier implementación de estos modelos típicos sobre un sistema embebido, es muy grande. Por este motivo, es interesante estudiar la posibilidad de que estos datos sintéticos puedan ser generados en tiempo real, por dispositivos de baja potencia computacional. De esta manera, por ejemplo, podrían generarse bancos de prueba donde se generen estas muestras artificiales, lo cual sería muy beneficioso para el estudio del medio al contar con equipos que integren estas funcionalidades.

El presente artículo parte de los resultados obtenidos en un trabajo previo [12], donde se realizó un estudio sobre la aplicación de las técnicas de aprendizaje profundo para la generación de espectrogramas sintéticos. En dicho artículo, se propusieron y analizaron arquitecturas de Autoencoders simples que parten de una representación bidimensional, en particular, el espectrograma de magnitud logarítmica, para la codificación y decodificación de registros de audio provenientes del ámbito submarino. En concreto, se comparó el desempeño de Autoencoders convencionales (es decir, con un mapeo discreto al espacio latente) y variacionales, basados en capas convolucionales para la extracción de características (CAE y VAE, respectivamente). La principal ventaja de estas arquitecturas, radica en la sencillez del modelo. Es decir, se logra un desempeño aceptable manteniendo una reducida complejidad en el mismo. Esta característica es de destacar, debido a que se busca implementar modelos generativos sobre sistemas embebidos de bajos recursos.

Dado que la arquitectura de los modelos ha demostrado tener un gran impacto en la respuesta del mismo, en el presente trabajo se propone estudiar una arquitectura de red híbrida basada en capas convolucionales y capas recursivas, donde se conserven las virtudes logradas en las arquitecturas CAE y VAE, bajo la premisa de baja complejidad.

La red propuesta se contrasta con las estudiadas en [12] y se extraen conclusiones. De la comparación surge la mejor opción, que luego se utiliza para la generación de espectrogramas artificiales.

II. Objetivos

En base a Carnaghi [12] se puede indicar que un modelo general simple, permite recuperar las características latentes correspondientes a diferentes espectrogramas de sonidos subacuático de baja frecuencia. Además, queda en evidencia que la utilización de AE de baja dimensionalidad puede ser un camino viable para la obtención de datos sintéticos.

El objetivo del presente estudio radica en extender los resultados mencionados, a partir de plantear una nueva arquitectura que conserve las cualidades de las CAE y VAE presentadas en [12]. Es importante mencionar que la arquitectura propuesta debe continuar con la premisa de baja complejidad, asociada a una posible implementación sobre un sistema embebido de bajo costo.

La arquitectura que se plantea en este trabajo utiliza una primera etapa convolucional, al igual que las anteriores, e incorpora recursividad en la siguiente etapa. Por último, con el objetivo de explotar las virtudes presentes en las arquitecturas VAE, se realiza un mapeo continuo del espacio latente.

III. Metodología

En la presente sección, se describe el proceso establecido para el entrenamiento de los modelos propuestos y su posterior utilización para la síntesis de nuevos espectrogramas. La metodología consisten en: adecuación de los datos de entrada, diseño de arquitecturas de AE, entrenamiento del modelo y obtención de métricas, comparación de resultados entre arquitecturas, selección del mejor modelo y generación de espectrogramas sintéticos.

III-A. Adecuación de los datos de entrada

La metodología de síntesis inicia con un acondicionamiento de los datos, cuyo objetivo es obtener una representación bidimensional de las muestras de audio originales. Para ello, se generan matrices bidimensionales que corresponden al espectrograma de magnitud logarítmica para cada muestra de audio. Dicha representación matricial combina la energía existente en un rango de frecuencia contemplado, y su variación en el tiempo del registro. A su vez, permite una directa re-interpretación como imagen en escala de grises, apta como entrada al modelo.

Los datos disponibles fueron originalmente obtenidos a partir de diversas fuentes y pertenecen a diferentes subfamilias dentro de la familia de Ballenas Barbadas [13]–[16]. Para constituir un conjunto correctamente estructurado, los datos deben normalizarse. Los parámetros considerados para este proceso de normalización se listan a continuación:

- 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.

Una vez calculado el espectrograma asociado a cada registro de audio, se adecuan sus dimensiones para coincidir con las dimensiones de datos en formato imagen, es decir [número de filas, número de columnas, número de canales de color]. Luego se los agrupa conformando un conjunto de datos destinado al entrenamiento del modelo. El mismo, es, a su vez, dividido en: datos en entrenamiento (D_{Train}), validación (D_{Val}) y evaluación (D_{Test}).

En la Tabla I se resumen la cantidad de datos disponibles y la forma del tensor de entrada correspondiente. Por otro lado, en [17] se encuentran disponibles: el tensor de entrada correspondiente a los diferentes registros

normalizados de Ballenas Barbadas, el vector de muestras temporales y el vector de frecuencia necesarias para la realización de los espectrogramas posteriores.

DATOS	N	Dimensión del tensor de entrada	
		Datos Entrenamiento	Datos Evaluación
B. Barbadas (todas)	6715	[6043,128,196,1]	[672,128,196,1]

Tabla I: Datos disponibles para el entrenamiento de los modelos: cantidad y forma del tensor de entrada

III-B. Arquitecturas de Autoencoders

Las arquitecturas de AE que se estudian en este trabajo son: CAE, VAE y CAE-LSTM (CAE with Long Short Term Memory layers).

En la Fig. 1 se ilustra la arquitectura CAE, la cual emplea capas convolucionales para el proceso de codificación y extracción de características. Además, cuenta con una capa flatten y una capa densamente conectada para mapear las características extraídas, a un vector representativo (espacio latente). Posteriormente, el vector de espacio latente obtenido se emplea para recuperar la imagen de entrada mediante un proceso de decodificación que espeja el procesamiento previo.

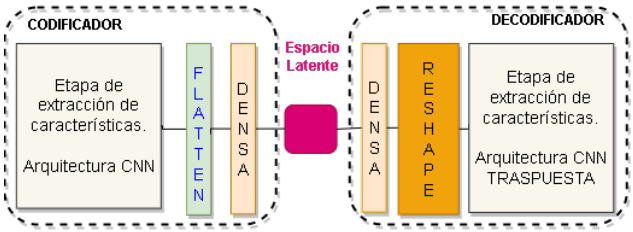


Fig. 1: Arquitectura de un CAE genérico.

La arquitecturas VAE, por su parte, es ilustrada en la Fig. 2 y es similar a la CAE con la diferencia de realizar un mapeo a un espacio latente continuo. Para ello, el aprendizaje tiene por objetivo descubrir la distribución de las características latentes, en lugar del mapeo a vectores individuales [7]. Con esta finalidad, se agregan capas densamente conectadas que extraen el valor medio (μ) y la varianza (σ) de los valores obtenidos en capas previas.

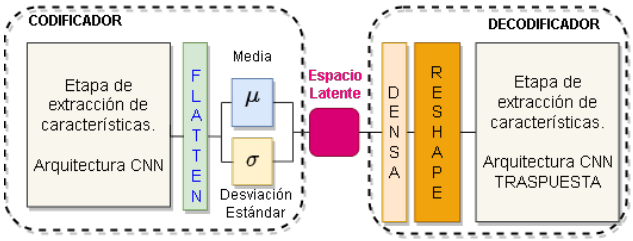


Fig. 2: Arquitectura de un VAE genérico.

Por último, la arquitectura CAE-LSTM es presentada en la Fig. 3, donde se observa que tanto la etapa de codificación como la etapa de decodificación están compuestas por una etapa convolucional y una etapa recursiva. De esta forma, se divide el procesamiento interno de los datos empleando, en primer instancia, capas convolucionales para extraen las características de las imágenes o matrices del espectrograma y, al mismo tiempo, conservar la correlación local de los datos mediante el uso de kernels; y en segunda instancia, capas recursivas cuyo propósito es descubrir patrones temporales en los datos, es decir, las diferentes características obtenidas de las capas convolucionales. El objetivo de la incorporación de etapas recursivas es brindar memoria a la red, lo que permite aprender y aprovechar la naturaleza ordenada de las observaciones de las secuencias de entrada [18]. De esta manera, la etapa recurrente se encarga de la extracción de características temporales. Finalmente, para obtener un mapeo continuo a un espacio latente, se utiliza el enfoque presentado en la arquitectura VAE.

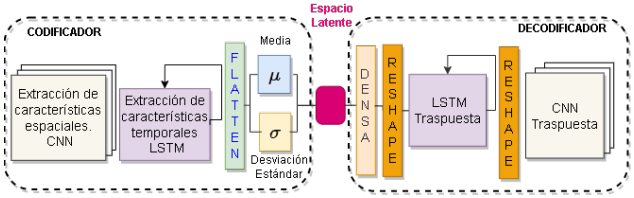


Fig. 3: Arquitectura de un CAE-LSTM genérico.

Las tres arquitecturas propuestas poseen en la etapa de extracción de características 4 capas convolucionales con los siguientes parámetros:

- 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.

Además, todas las etapas emplean función de activación tipo RELU y auto-padding. A su vez, aplican instancias de Batch Normalization, con el fin de acelerar el proceso de entrenamiento, mejorar las propiedades de normalización de la red, volverla más robusta frente a diferentes esquemas de inicialización y tasas de aprendizaje [19]. En la Fig. 4, se realiza una descripción gráfica de la etapa CNN, las transformaciones involucradas y las dimensiones asociadas a las matrices hasta la salida de la capa Flatten.

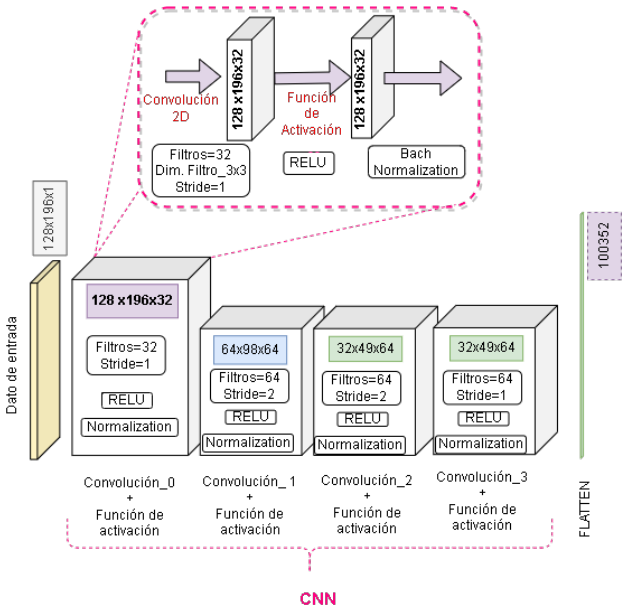


Fig. 4: Descripción de la etapa de extracción de características.

Al igual que en los aspectos anteriormente listados, la dimensión del espacio latente es un hiperparámetro.

En este caso se decide obtener diferentes arquitecturas, a partir de variar únicamente esta dimensión. Además, en los modelos CAE-LSTM se analiza el impacto de incorporar más de una etapa de recursividad y de variar el número de celdas de memoria en cada una de las mismas.

III-C. Entrenamiento y métricas

El objetivo del entrenamiento de un AE es lograr una apropiada reconstrucción de los datos presentados. Los hiperparámetros durante el entrenamiento se establecieron en:

- Función de pérdida: MSE,
- Optimizador: Adam [20],
- Taza de aprendizaje: 0.0005,
- Tamaño de Minibatch: 100,
- D_{Train}: 6043 con $\%D_{Val}$: 0.2,
- Épocas: 60(CAE y CAE-LSTM) y 48(VAE).

Se busca mantener estos hiperparámetros a fin de realizar una comparación objetiva entre los modelos propuestos. Notar que, en el caso de la cantidad de épocas, el valor cambia debido al punto en que produce sobreajuste (overfitting).

III-D. Comparación y selección de arquitectura

Para la elección de la arquitectura a utilizar en la síntesis, se entrenan las redes variando la dimensión del espacio latente. A fin de mantener la premisa inicial de sencillez en el modelo, las posible dimensiones de este espacio latente se mantendrán en 2, 3 y 4.

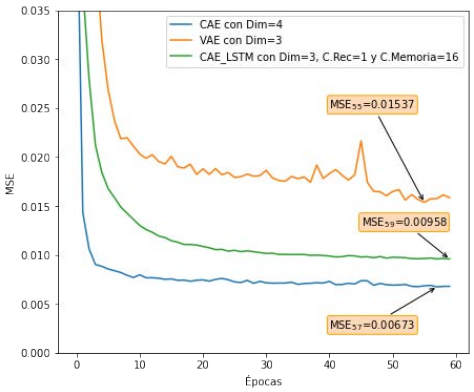
En la Fig. 5(a) se presenta el MSE correspondiente a cada arquitectura y, además, para el caso de CAE-LSTM los errores con datos de validación y entrenamiento en la Fig. 5(b). Los demás casos fueron previamente presentados en [12].

Se observa que la arquitectura que posee menor error cuadrático medio, corresponde a la CAE con dimensión del espacio latente igual a 4. Es notable destacar que la arquitectura CAE-LSTM no presenta mejora con respecto a la CAE, pero sí frente a la VAE.

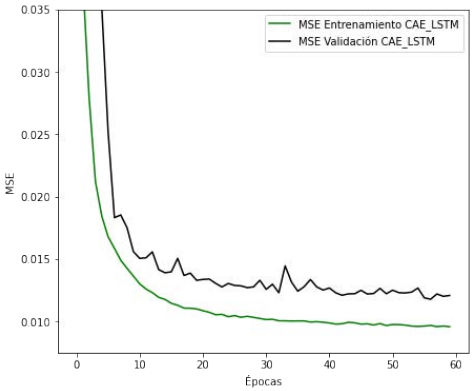
En la Fig. 6 se evalúan las diferentes reconstrucciones logradas con una imagen perteneciente al conjunto de evaluación (seleccionada al azar), con el fin de corroborar el significado del MSE en forma gráfica. Se observa que las reconstrucciones realizadas, se corresponden con los resultados obtenidos durante el entrenamiento. En este caso, la arquitectura CAE-LSTM seleccionada es aquella que obtuvo menor MSE durante la etapa de entrenamiento.

Complementariamente, en la Tabla II, se presentan los valores de MSE y error de similitud estructural (SSIM) (el cual ofrece buena precisión de evaluación y simple formulación [21]) calculados sobre la misma imagen de evaluación. Adicionalmente, en dicha Tabla se listan las cantidades de parámetros asociados a cada arquitectura, a modo de figura representativa de la complejidad asociada a cada una.

Es notable que la arquitectura CAE-LSTM empleada presenta un error MSE levemente superior al obtenido mediante la arquitectura CAE, pero, a su vez, requiere menos de la mitad de los parámetros. Esto representa una ganancia en relación a la complejidad del modelo al momento de su implementación.

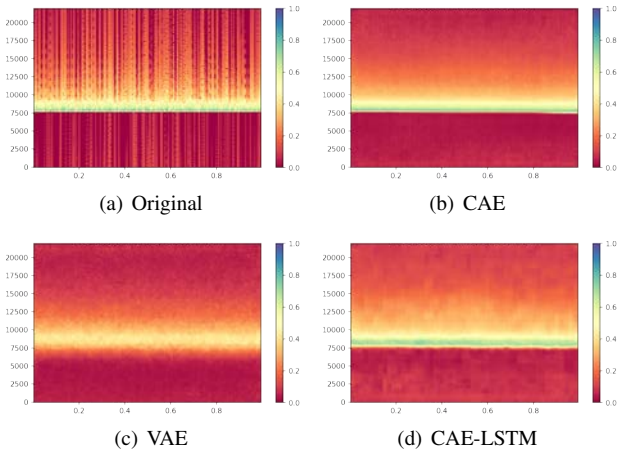


(a)



(b)

Fig. 5: a) MSE de entrenamiento para diferentes arquitecturas de autoencoders. b) MSE con datos de validación y de entrenamiento para CAE-LSTM.



(a) Original (b) CAE

(c) VAE (d) CAE-LSTM

Fig. 6: Espectrograma original y diferentes reconstrucciones según la arquitectura de AE empleada.a) Espectrograma original b) Reconstrucción con CAE c) Reconstrucción con VAE d) Reconstrucción con CAE-LSTM

Tabla II: Características de las diferentes arquitecturas. Los errores MSE y SSIM se calculan sobre una imagen del conjunto de datos de evaluación.

	CAE	VAE	CAE-LSTM
Cant. Parámetros	707459	1209223	318551
MSE (Evaluación)	0.00468	0.00977	0.00703
SSIM (Evaluación)	0.53452	0.44437	0.48504

En la Tabla III se comparan las arquitecturas CAE-LSTM estudiadas en este trabajo, con respecto a la variación del MSE en función de la cantidad de capas recursivas y la cantidad de celdas de memoria. A partir de un análisis

de la misma, se concluye que la incorporación de más etapas de recursividad no mejora significativamente el comportamiento de la red, por lo que un modelo con una etapa de recursividad de 16 o 32 celdas resulta ser la mejor alternativa dentro de esta arquitectura.

Tabla III: Valores de MSE para distintas arquitecturas CAE-LSTM

C. Recursivas	C.Memoria	Dim	MSE Train	MSE Val
1	4	2	0.0150	0.0206
		3	0.0445	0.466
		4	0.0108	0.0134
	8	2	0.0125	0.0134
		3	0.01004	0.0145
		4	0.0101	0.0125
	16	2	0.0110	0.0155
		3	0.0096	0.0121
		4	0.0097	0.0124
	32	2	0.0105	0.0152
		3	0.0094	0.0124
		4	0.0094	0.0118
	64	2	0.0109	0.0165
		3	0.0092	0.0119
		4	0.0092	0.0112
2	4	2	0.0148	0.0203
		3	0.0445	0.0466
		4	0.0445	0.0491
	8	2	0.0111	0.0158
		3	0.0445	0.0501
		4	0.0444	0.0482
	16	2	0.0108	0.0149
		3	0.0444	0.0484
		4	0.0445	0.0487
	32	2	0.0100	0.0140
		3	0.0445	0.0469
		4	0.0444	0.0487
	64	2	0.0095	0.0131
		3	0.0444	0.0472
		4	0.0092	0.0115

III-E. Generación del vector de código

Durante la instancia de síntesis se utiliza únicamente el bloque decodificador de un modelo ya entrenado. Como se observa en la Fig. 7, a este bloque ingresa un vector de valores aleatorios que debe ser capaz de generar una imagen sintética semejante a las que entrenaron el modelo y que denominamos vector de código o vector generador.

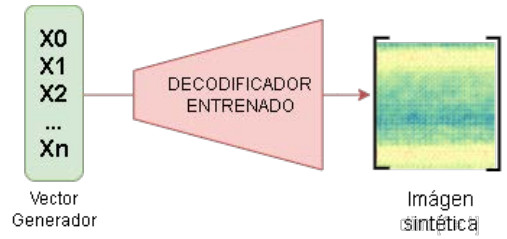


Fig. 7: Esquema de síntesis.

El vector de código generado para este fin deben ser coherente con los casos presentados en instancias previas. Por lo tanto, para obtener un valor significativo, se sigue el siguiente criterio que fue presentado inicialmente en [12]:

1. Con los vectores de espacio latente generados en el entrenamiento, se obtiene una matriz de dimensiones [N_{Train},3].
2. Se analiza la interdependencia entre los datos de cada vector de espacio latente (fila de la matriz) mediante la matriz de correlación. En (1), la matriz de correlación muestra que no existe una marcada relación intra-vector.

$$\begin{bmatrix} 1 & -0{,}0183 & -0{,}1159 \\ -0{,}0183 & 1 & -0{,}0197 \\ -0{,}1159 & -0{,}0197 & 1 \end{bmatrix} \quad (1)$$

3. Una vez analizada la interdependencia entre las dimensiones del espacio latente, se analiza la distribución de los valores con el objetivo de asemejarlo a una función densidad de probabilidad parametrizable.
4. Se genera un vector de código aleatorio [X_0,X_1, X_2,X_3] siguiendo las funciones encontradas en el paso 3.

Estos valores actúan como punto de partida para generar vectores de códigos aleatorios que ingresan en el bloque decodificador entrenado.

IV. Resultados

La adecuación de los datos de entrada, el entrenamiento de los modelos, los vectores de código generados y la síntesis de los espectrogramas sintéticos fueron realizados sobre una Notebook con las siguientes características: Procesador Ryzen 7 serie 5800, con 16GB de memoria RAM, placa de video NVIDIA GeForce RTX 3050 TI, disco SSD de 100GB, S.O Windows 11.

IV-A. Síntesis de espectrogramas sintéticos

Para la síntesis de espectrogramas sintéticos, el primer paso consiste en la generación de vectores de espacio latentes que actúan como entrada para la etapa de decodificación (vector de código). El procedimiento seguido con tal fin es el descrito en III-E.

En la Fig. 8 se observa la distribución de cada valor del espacio latente, generado con la arquitectura CAE-LSTM durante el entrenamiento. Se deduce que la distribución de valores de código obtenidos, pueden aproximarse mediante una distribución Gaussiana. De esta manera, es posible obtener un valor medio y una desviación estándar representativa de los datos reales con los que fue entrenada la red.

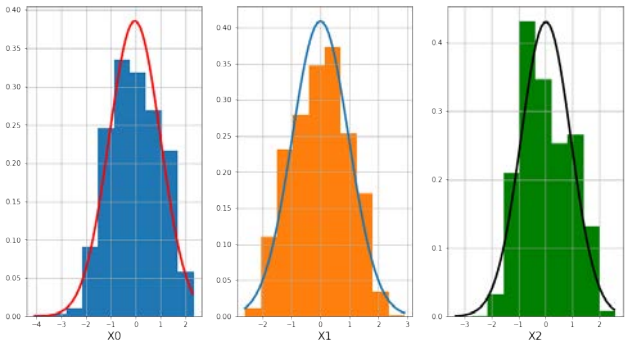


Fig. 8: Distribución normalizada de los valores del vector de espacio latente obtenidos con todos los datos de entrenamiento para la arquitectura CNN-LSTM.

Finalmente, una vez obtenida la estadística para cada dimensión del espacio latente, las mismas se emplean para generar vectores aleatorios que actúen como semilla del proceso de síntesis.

Luego, en las Figs. 9, 10 y 11 se muestran ejemplos de imágenes reales y sintetizadas con los modelos CAE y

CAE-LSTM entrenados. Los códigos generadores de estos modelos pueden consultarse en [22].

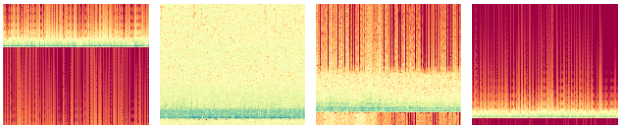


Fig. 9: Espectrogramas Reales, obtenidos a partir de registros aleatorios de ballenas Barbadas.

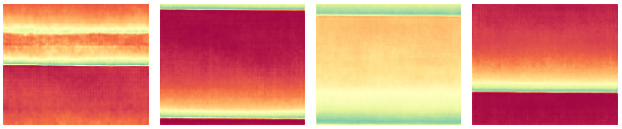


Fig. 10: Espectrogramas sintetizados aleatoriamente, obtenidos con un decodificador CAE entrenado, a partir de registros de ballenas Barbadas.

En el primer caso, se presentan espectrogramas de ballenas reales, seleccionados aleatoriamente. En los restantes casos, se sintetizan espectrogramas partiendo de vectores de espacio latente obtenidos con el proceso descripto previamente, por lo tanto, no existe correspondencia entre las imágenes presentadas.

V. Conclusión

En base a los resultados obtenidos se puede afirmar que la arquitectura más prometedora es la CAE. El modelo CAE propuesto posee una cantidad reducida de capas convolucionales y de parámetros asociados a operaciones matemáticas. Esto reduce notablemente los errores asociados en una implementación física, donde existe una cantidad de bits limitados para el procesamiento.

Por otra parte, se observó que las arquitecturas con espacios recurrentes permitieron una reducción más notoria en el número de parámetros en comparación a la arquitectura anteriormente mencionada. Sin embargo, tal reducción del modelo conlleva un leve detrimento del desempeño logrado por el modelo. Además, la naturaleza recursiva de la red, implica mayor cantidad de operaciones por parámetro. En consecuencia, se puede concluir que estas arquitecturas son recomendables en aquellos casos donde el número de parámetros a implementar es el principal factor limitante.

Como trabajo a futuro se plantea la generación de registros de audio a partir de espectrogramas y la implementación de estos modelos sobre algún sistema embebido de bajo costo, como puede ser un microcontrolador o una FPGA de bajos recursos. Se plantea la evaluación sobre diferentes plataformas y la elección de la mejor opción ponderando el costo y el desempeño.

VI. Agradecimientos

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

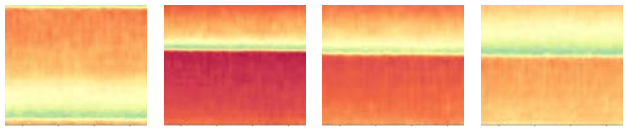


Fig. 11: Espectrogramas sintetizados aleatoriamente, obtenidos con un decodificador CAE-LSTM entrenado, a partir de registros de ballenas Barbadas.

Referencias

[1] T. Markus and S. P. P. Silva, *Managing and Regulating Underwater Noise Pollution*. Springer International Publishing, 2018, pp. 971–995. [Online]. Available: https://doi.org/10.1007/978-3-319-60156-4_52

[2] N. Jones, "Ocean uproar: saving marine life from a barrage of noise," *Nature*, vol. 568, pp. 158–161, 04 2019.

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

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

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

[6] 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.

[7] 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.

[8] N. Mansouri and Z. Lachiri, "Human Laughter Generation using Hybrid Generative Models," *KSII Transactions on Internet and Information Systems (TIIS)*, pp. 1590–1609, 2021.

[9] A. Sarroff and M. Casey, "Musical audio synthesis using autoencoding neural nets," in *In Joint International Computer Music Conference (ICMC) and Sound and Music Computing conference (SMC)*, 2014.

[10] 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.

[11] J. Wei, "AlexNet: The Architecture that Challenged CNNs," Towards Data Science, 2019. [Online]. Available: https://acortar.link/IrMULc(acceso:25dejuniode2022).

[12] M. Carnaghi and M. C. Cebedio, "Espectrogramas de registros de Ballenas Barbadas, sintetizados a partir de Autoencoders," *Congreso Argentino de Sistemas Embebidos CASE*, 08 2022.

[13] "Ocean Sound Library: Natural and Man-Made," Ocean Conservation Research, 2022. [Online]. Available: https://ocr.org/sound-library/

[14] "Song and Sound," Whale Trust, 2022. [Online]. Available: https://whaletrust.org/song-sound//

[15] "Marine Mammals," Discovery of Sound in the Sea, 2022. [Online]. Available: https://dosits.org/galleries/audio-gallery/marine-mammals/

[16] "Watkins Marine Mammal Sound Database," Woods Hole Oceanographic Institution, 2022. [Online]. Available: https://whoicf2.whoi.edu/science/B/whalesounds/index.cfm

[17] M. C. Cebedio and M. Carnaghi, "Datos," Google Drive, 2022. [Online]. Available: https://drive.google.com/drive/folders/1HxalJvSf3L4MXW8VsFXsvYelkTb5xYDj?usp=sharing

[18] J. Brownlee, *Long Short-Term Memory Networks With Python*, 1st ed. Machine Learning Mastery, 2017.

[19] 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

[20] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, 12 2014.

[21] 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.

[22] M. C. Cebedio and M. Carnaghi, "Repositorio-CASE2022," GitHub, 2022. [Online]. Available: https://github.com/Reposinnombre/CASE2022