Autocodificador generativo cuantizado de espectrogramas ficticios

M. C. Cebedio, L. A. Rabioglio, L. J. Arnone, J. Castiñeira Moreira, M. Medina y L. De Micco
ICYTE, Depto. de Electrónica y Computación, Facultad de Ingeniería - UNMDP
Mar del Plata, 7600, Argentina
{celestecebedio,lucas.rabioglio,leoarn,casti,mmedina,ldemicco}@fi.mdp.edu.ar

***Resumen*—Este trabajo presenta un autocodificador para la generación eficiente de espectrogramas de sonidos subacuáticos cuantizado y optimizado para ser implementado en dispositivos con recursos limitados. Se evalúa su rendimiento en diversos escenarios, demostrando su capacidad para generar espectrogramas con baja dimensionalidad y para adaptarse a diferentes entornos acústicos. La optimización para implementación en hardware se enfoca en encontrar un equilibrio entre el costo computacional, la precisión del modelo, manteniendo el desempeño del modelo no cuantizado.**

***Palabras Claves*—Autocodificador Generativo, Cuantización, Espectrogramas, Hardware, Optimización, Pruning, Punto Fijo, Tranfer Learning.**

I. Introducción

En los últimos años, los autocodificadores (AE) surgen como una herramienta poderosa en el campo del aprendizaje automático (Machine Learning, ML) y la inteligencia artificial (IA). Estos modelos son eficientes en diversas tareas, como la eliminación de ruido en archivos de audio, la generación de imágenes ficticias, el agregado de color en imágenes y la detección de objetos [1].

En el contexto específico de la generación de datos ficticios, se han desarrollado modelos generativos de aprendizaje profundo (Deep Learning, DL) basados en técnicas de aprendizaje no supervisado, como el autocodificador convolucional (CAE) y el autocodificador variacional (VAE) [2]. Estos modelos demandan grandes volúmenes de datos, poder computacional extenso y largos períodos de entrenamiento. Ante esta situación, ha surgido la técnica conocida como Transfer Learning (TL) [3], que permite la reutilización de modelos previamente entrenados como punto de partida de nuevas tareas [4]. La aplicación de metodologías de TL aumenta la capacidad de generalización de los modelos de AE, lo que posibilita la obtención de resultados satisfactorios con conjuntos de datos reducidos.

Sin embargo, la complejidad de los modelos pre-entrenados, como el VGG-net (Visual Geometry Group Net), dificulta su implementación en sistemas embebidos debido a la gran cantidad de parámetros y operaciones aritméticas asociadas [5]. Por ello surge la necesidad de contar con modelos simples, pre-entrenados que estén optimizados para ser implementados en dispositivos con recursos limitados, como FPGAs.

En este sentido en [6] se presenta una arquitectura de Autocodificador generativo de baja dimensionalidad que exhibe características que pueden ser aprovechadas para construir un modelo generativo de espectrogramas ficticios de uso general. Bajo esta hipótesis, se plantea la posibilidad de cuantizar y optimizar dicho modelo para su implementación en sistemas embebidos. Sin embargo, esta implementación requiere un análisis detallado de la cuantización y la posterior reducción de la arquitectura, con el fin de lograr una optimización que permita su ejecución efectiva. Existen trabajos que abordan esta temática, como en [7] donde se presentan metodologías de cuantización para redes convolucionales, y en [8] donde los autores presentan una implementación en FPGA de un VAE, con estructura pipeline de baja latencia. En [9] los autores abordan la cuantización del modelo y su implementación en FPGA. Allí consideran arquitecturas totalmente conectadas y convolucionales de AEs y VAEs, y analizan su optimización.

En este trabajo, se presenta una arquitectura de autocodificador generativo que ha sido cuantizada y optimizada específicamente para su despliegue en dispositivos con recursos limitados. El objetivo principal es desarrollar una estructura capaz de generar espectrogramas ficticios implementable en dispositivos reconfigurables. Esta arquitectura se diseña para facilitar tanto el Transfer Learning (TL) como el re-entrenamiento, adaptándose eficazmente a entornos con limitaciones de recursos. Se ilustra con un generador de espectrogramas que puede crear varios tipos de sonidos acuáticos ficticios, ya sea mediante re-entrenamiento o la aplicación de TL.

Con el objetivo de desarrollar una arquitectura de autocodificador generativo versátil y adecuada para implementaciones en hardware, se sigue una metodología que incluye la selección del autocodificador generativo, la evaluación de la flexibilidad de su arquitectura, la cuantización y optimización del modelo, y la generación de espectrogramas sintéticos. El propósito final de esta metodología es preparar el modelo para su implementación en sistemas embebidos, buscando ofrecer un tiempo de ejecución rápido, baja latencia y un alto grado de flexibilidad en la generación de espectrogramas ficticios, lo cual se logra mediante un enfoque concentrado en la cuantización y reducción del modelo.

II. Selección de Autocodificador Generativo

La elección de la arquitectura y la evaluación de un modelo sencillo, es el punto de partida para la posible implementación en un sistema embebido que posea baja potencia computacional. Usualmente, los modelos generativos emplean VAEs para evitar distribuciones potencialmente complejas y discontinuidades en el espacio latente [10], las cuales dificultan el proceso de muestreo en la generación de los datos ficticios. En un análisis previo [6]

se concluyó que el modelo VAE presenta peor desempeño que el autocodificador tradicional, posiblemente debido a la limitada cantidad de datos disponibles en el conjunto de re-entrenamiento y a la gran dimensionalidad de los datos de entrada. Por esta razón, se opta por utilizar un CAE tradicional y, al momento de muestrear el espacio latente, reproducir las distribuciones obtenidas en dicho espacio mediante el método propuesto en [11], el cual permite sintetizar mapas caóticos que generan datos con distribuciones arbitrarias y pre-establecidas. Estos mapas caóticos son fácilmente adaptados a lógica digital [12].

El modelo de CAE seleccionado (ver Fig. 1), inicialmente fue entrenado con espectrogramas logarítmicos, correspondiente a sonidos generados por "ballenas Barbadas". La principal ventaja de esta arquitectura radica en su simplicidad. Es decir, ofrece un rendimiento aceptable con una complejidad reducida, lo que es fundamental para la implementación sobre un sistema de bajos recursos.

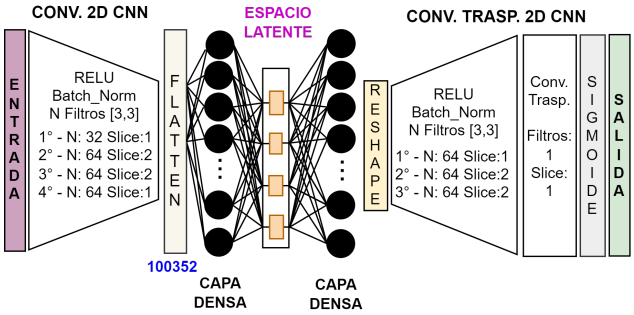


Fig. 1: Arquitectura propuesta de autocodificador convolucional.

El modelo presenta un total de 1.108.869 parámetros. Esta cantidad proporciona una idea de la dimensión del modelo, la complejidad de las operaciones involucradas y los tiempos de entrenamiento requeridos.

III. Evaluación del modelo Autocodificador Generativo

Para evaluar la flexibilidad de la arquitectura generativa propuesta, se llevan a cabo una serie de pruebas. El objetivo de estas pruebas es determinar la utilidad general de la arquitectura en diversos contextos.

1. Re-entrenamiento: Se re-entrena la red seleccionada con datos de background de la laguna Mar Chiquita - Bs. As., Argentina, para obtener un modelo generativo que se adecúe a estos nuevos datos.
2. Transfer Learning: Se emplea esta técnica sobre la arquitectura seleccionada para generar nuevos modelos a partir de la reducida cantidad de datos disponibles para el entrenamiento. Se evalúa en: diferentes tipo de ballenas Barbadas, ruido marino, peces y background de la laguna.

III-A. Resultados y análisis

Los resultados obtenidos (ver Tabla I) muestran que el modelo exhibe buenos resultados en una variedad de escenarios. Cuando se dispone de una cantidad considerable de datos, es conveniente re-entrenar el modelo inicial para adaptarse a las características específicas de los datos, mientras que en situaciones donde los datos son escasos, el modelo demuestra ser apto para el uso de TL.

		N	MSE		$< MSE >$	Ép.
			Train	Valid	Test	
1	Ballenas Barbadas	6708	0,005	0,0051	0,00546	80
	Background Laguna	8700	0,0023	0,0023	0,0026	60
2	Ballena Azul	146	0,0049	0,0076	0,0061	60
	Ballena Groenlandia	393	0,0043	0,0046	0,0049	60
	Peces	313	0,0053	0,0085	0,0070	60
	Ruido de Mar	1625	0,0081	0,0082	0,0106	60
	Background Laguna	268	0,0061	0,0052	0,0062	60
	Background Laguna	8700	0,0034	0,0037	0,0551	40

Tabla I: Resumen de métricas obtenidas en la evaluación de arquitectura: taza de aprendizaje=0,0005, Optimizador Adams, Dimensión de los datos [N,128,196,1]

IV. Cuantización y optimización del modelo

Los modelos de autocodificadores, suelen entrenarse en sistemas con representación en punto flotante. Sin embargo, al implementarlos en hardware de punto fijo, es necesario adaptar la precisión de los parámetros y operaciones del modelo a las limitaciones del hardware. En este punto del diseño, otros parámetros influyen en el *trade-off* entre precisión y costo.

Para evaluar esta cuestión, se emplea una arquitectura cuantizada que replica las mismas capas que el modelo original. Además, se lleva a cabo una optimización colaborativa que incorpora varias técnicas con el objetivo de obtener un modelo que, al implementarse, logre el mejor equilibrio entre velocidad de inferencia, tamaño del modelo y precisión

El primer análisis se centra en la técnica de entrenamiento a utilizar una vez cuantizado el modelo. Dado que esta red es relativamente pequeña se utiliza un enfoque de entrenamiento consciente de la cuantización, ya que ofrece mejor precisión. Además se usa una taza de aprendizaje muy baja para garantizar que el modelo pueda ajustarse de manera precisa a los datos disponibles.

El segundo análisis consiste en decidir si se realiza un "Fine-Tuning" de una arquitectura pre-entrenada en punto flotante o se construye una arquitectura cuantizada y se entrena desde cero. Para realizar un análisis comparativo, se evalúan dos arquitecturas de iguales características bajo estos dos enfoques. Se observa que el modelo entrenado desde cero requiere 320 épocas para alcanzar un MSE de entrenamiento de 0,0135, mientras que el modelo con ajuste fino logra un error de 0,008 con solo 100 épocas, indicando una convergencia más rápida y una mayor precisión. Por lo tanto, se adopta la estrategia de ajuste fino para la siguiente etapa.

Definida la estrategia de entrenamiento, se continúa con la cuantización del modelo con el fin de evaluar sus efectos en los recursos de hardware y el error en la estimación. Para ello, se analiza la utilización de diferentes estrategias y se comparan resultados.

En la última etapa del diseño, se evalúan otras estrategias que colaboren en la compresión del modelo. En esta instancia se busca, mantener la precisión del modelo, con la implementación que menos hardware conlleve, mediante la técnica de poda (*Pruning*). Esta técnica implica la eliminación de conexiones redundantes o poco importantes en el modelo, lo que reduce la complejidad y el tamaño. Aplicar la técnica de *Pruning* sin comprometer significativamente su rendimiento, requiere un nuevo ajuste fino del modelo. El aumento de la dispersión (*Sparcity*) asociado al *Pruning* colabora con la compresión del modelo en el dispositivo final.

Modelo	Conv2D/Conv2DTrans		Activación	Densa		Sin Pruning		Con Pruning		Energía
	Kernel	Bias		Kernel	Bias	MSE*	Sparsity	MSE*	Sparsity	
1	Qbit(8,0,1)	QPO2(4)	Relu_binary	Qbit(4,0,1)	Qbit(4)	0,0084	0,2263	0,0078	0,5674	1
2	Sth_Ternary	PO2(4)	Relu_binary	Qbit(4,0,1)	Qbit(4)	0,0095	0,3056	0,0103	0,5693	0,278
3	Qbit(8,0,1)	QPO2(4)	Adapt_Relu	Qbit(4,0,1)	Qbit(4)	0,0062	0,2263	0,0063	0,5619	1,04
4	Qbit(8,0,1)	QPO2(4)	Relu_binary	Sth_Ternary	Qbit(4)	0,0080	0,4159	0,0079	0,5796	0,995
5	Sth_Ternary	QPO2(4)	Adapt_Relu	Qbit(4,0,1)	Qbit(4)	0,0066	0,4536	0,0068	0,6547	0,3
6	Qbit(8,0,1)	QPO2(4)	Relu_binary	Qbit(4,0,1)	QPO2(4)	0,0078	0,2225	0,0078	0,5678	1
7	Sth_Ternary	QPO2(4)	Adapt_Relu	Sth_Ternary	QPO2(4)	0,0066	0,4736	0,007	0,5722	0,3

Tabla II: Tabla resumen de resultados. MSE*: error cuadrático medio, correspondiente al promedio de todos los MSE del conjunto de Test.

IV-A. Resultados y análisis

Los resultados se presentan en la Tabla II, donde se destacan las métricas de precisión (MSE), dispersión de ceros (Sparsity) y energía total para cada modelo. La energía refleja los elementos y operaciones en una implementación posterior, mientras que el Sparsity indica la posible reducción de la arquitectura. Es importante señalar que la herramienta utilizada proporciona una estimación de la cantidad de elementos, pero no representa una medida precisa de la implementación final. Esta estimación se utiliza para la comparación entre arquitecturas bajo condiciones similares, pero los resultados finales variarán según el dispositivo específico en el cual se implemente el modelo. La Tabla III muestra el reporte de elementos de la etapa codificadora para el Modelo 1, que se utiliza de punto de comparación.

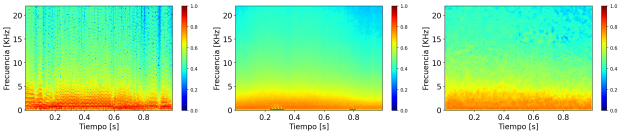
Codificador	Tipo de operaciones	Pesos	Bias	Sparcity
Densa	401408 (sfmult_4_32)	401408 (4-bit unit)	100352 (4-bit unit)	0,4972
Convolucional_0	7225344 (smult_8_8)	288(8-bit unit)	32 (4-bit unit)	0,0844
Convolucional_1	115605504 (smux_8_1)	18432 (8-bit unit)	64 (4-bit unit)	0,1655
Convolucional_2	57802752 (smux_8_1)	36864 (8-bit unit)	64 (4-bit unit)	0,1111
Convolucional_3	57802752 (smux_8_1)	36864 (8-bit unit)	64 (4-bit unit)	0,1060

Tabla III: Reporte de elementos, operaciones y Sparcity para la etapa Codificadora del modelo 1

En la Tabla II, el Modelo 3 destaca por su alta precisión, incluso superando al entrenamiento en punto flotante, a costa de un mayor valor de energía. La cuantización PO2 no mejora significativamente la precisión ni reduce la energía de forma notable, pero simplifica las operaciones aritméticas al limitar los valores a potencias de 2, lo que se debería reflejar en la rapidez de la implementación. En contraste, el método Stochastic Ternary reduce considerablemente la energía, aunque con una pérdida importante en la precisión.

El Modelo 5 permite evaluar si la reducción energética del método Stochastic Ternary compensa la pérdida de precisión por la Activación Adaptativa. Finalmente, el Modelo 7 logra un equilibrio óptimo integrando estas observaciones según requisitos específicos. La selección del modelo óptimo depende del objetivo de optimización deseado.

A modo de obtener un panorama rápido de los resultados obtenidos al aplicar datos de Test a la red completa (codificador + decodificador), se presenta en la Fig.2 una comparación entre espectrogramas generados a partir de audios reales, espectrogramas reconstruídos utilizando la arquitectura CAE de punto flotante y espectrogramas reconstruídos con la arquitectura CAE cuantizada y optimizada. Se observa que las imágenes generadas por los CAEs no son réplicas exactas de las originales. Esta discrepancia se debe a una característica fundamental de estos modelos: no están diseñados para replicar fielmente las imágenes de entrada, sino más bien para generar nuevas imágenes que comparten similitudes con el conjunto de datos de entrenamiento. En lugar de copiar cada detalle de una imagen dada, los CAEs aprenden a capturar las características distintivas y el estilo general de las imágenes. Esto les permite crear nuevas muestras que son reconocibles dentro del mismo dominio, pero no idénticas a ninguna imagen específica de entrenamiento.



(a) (b) (c)

Fig. 2: Espectrogramas de registros de audio correspondientes a ballenas Barbadas: a) Original de Test. b) Reconstruido con CAE entrenado en punto flotante, con MSE= 0,00365. c) Reconstruido con Modelo 1 de CAE, con MSE= 0,003562.

V. Generación de espectrogramas ficticios con arquitectura cuantizada y optimizada

Para generar datos ficticios, se emplea exclusivamente el bloque decodificador del CAE cuantizado, que recibe como entrada un vector de dimensión 4. Este vector representa las imágenes que se desean generar de manera ficticia, y por lo tanto, la distribución de las posibles muestras desempeña un papel crucial en la reconstrucción del espectrograma. La distribución de cada valor en el espacio latente se obtiene a partir de todos los vectores de espacio latente generados con los datos de entrenamiento. En este caso particular, las distribuciones han demostrado ser Gaussianas y no correlacionadas. Por lo tanto, basta con calcular el valor medio y la desviación estándar. Estos valores actúan como punto de partida para generar vectores de códigos aleatorios de dimensión 4 que se introducen en el bloque decodificador entrenado.

En situaciones donde el espacio latente no presenta distribuciones Gaussianas, se procede a aproximar la distribución utilizando el método propuesto por Rogers et al.

La Fig. 3 muestra espectrogramas reales y sintéticos, con sus correspondientes PDFs. Dado que la generación es aleatoria, no hay correspondencia directa entre a) y c). Las imágenes mostradas en la Fig. 3c conservan el estilo característico de un conjunto de espectrogramas de ballenas barbadas. La comparación uno a uno entre imágenes permite observar el estilo, pero nada dice al respecto de la capacidad de emular una distribución. En la Fig. 3c, se evidencia que la imagen muestra variaciones en la distribución de la amplitud a lo largo de la frecuencia, en contraste con la muestra específica presentada en a). Estas variaciones pueden ser similares a otras muestras del conjunto de datos, pero no son réplicas exactas de ninguna de ellas. Esta observación subraya que los autoencoders generativos no se centran en replicar las entradas

de manera exacta, sino en generar nuevas imágenes que pertenezcan al mismo dominio visual que los datos de entrenamiento. La comparación más relevante se realiza al observar los histogramas correspondientes a los audios: el audio real (Fig. 3b) y el audio reconstruido a partir del espectrograma generado (Fig. 3d). Estos histogramas muestran la similitud en la distribución de los audios, lo que ilustra cómo los autoencoders generativos pueden capturar el estilo general de los datos de entrenamiento mientras introducen variaciones que los distinguen como muestras únicas dentro del mismo contexto acústico y visual.

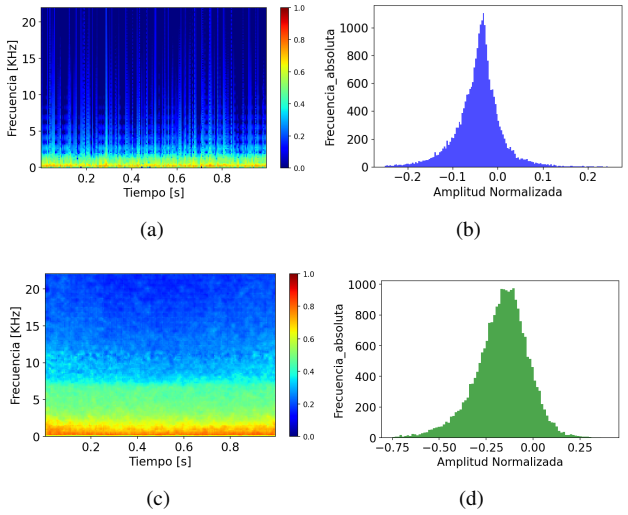


Fig. 3: Ballenas barbadas. a) Espectrograma Real b) PDF del audio Real. c) Espectrograma Ficticio y d) PDF de audio reconstruído a partir de espectrograma Ficticios obtenido con Modelo 1 luego del Pruning.

VI. Conclusión y trabajo a futuro

Se logró una arquitectura adaptable y eficiente, optimizada para latencia y retardo, con aplicabilidad en diversos dominios. Estas optimizaciones demuestran la viabilidad de la implementación de la red generadora. Se observa la posibilidad de alcanzar una precisión superior al entrenamiento en punto flotante a partir de una arquitectura previamente entrenada.

El análisis se basa en un enfoque de grilla, que, aunque menos eficiente que herramientas modernas como la optimización Bayesiana, permite una exploración de la influencia de cada parámetro en el modelo.

Como trabajo futuro, se propone la implementación del modelo mediante una arquitectura reconfigurable que permita ajustar todos los parámetros.

VII. Agradecimientos

Esta investigación fue financiada por la Agencia Nacional de Promoción Científica y Tecnológica (PICT2019-3024) y la Facultad de Ingeniería de la Universidad Nacional de Mar del Plata (FI-UNMDP).

Referencias

[1] D. Bank, N. Koenigstein, and R. Giryes, "Autoencoders," *Machine learning for data science handbook: data mining and knowledge discovery handbook*, pp. 353–374, 2023.

[2] 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.

[3] E. Tsalera, A. Papadakis, and M. Samarakou, "Comparison of Pre-Trained CNNs for Audio Classification Using Transfer Learning," *Journal of Sensor and Actuator Networks*, vol. 10, p. 72, 12 2021.

[4] D. Sarkar, R. Bali, and T. Ghosh, *Hands-On Transfer Learning with Python*, 1st ed. Packt, 1995, ISBN 13: 9781788831307.

[5] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint*, 2017.

[6] xxxxxxxxxxxxxxxx, "Espectrogramas de registros de Ballenas Barbadas sintetizados a partir de arquitecturas de Autoenconders: CAE, VAE y CAE-LSTM," *Revista elektron*, vol. 6, 2022.

[7] F. G. Zacchigna, "Methodology for cnn implementation in fpga-based embedded systems," *IEEE Embedded Systems Letters*, 2022.

[8] Que and et. al, "Low latency variational autoencoder on fpgas," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 2024.

[9] Govorkova and et. al, "Autoencoders on field-programmable gate arrays for real-time, unsupervised new physics detection at 40 mhz at the large hadron collider," *Nature Machine Intelligence*, vol. 4, no. 2, p. 154–161, Feb. 2022.

[10] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[11] A. Rogers, R. Shorten, and D. M. Heffernan, "Synthesizing chaotic maps with prescribed invariant densities," *Physics Letters A*, vol. 330, no. 6, pp. 435–441, 2004.

[12] R. E. Lopresti, M. Antonelli, J. D. Dondo Gazzano, and L. De Micco, "Diseño e implementación de prng caótico con distribución variable en el tiempo," *Elektron*, vol. 6, no. 1, pp. 46–51, 2022.