Generación de Información Acústica Sintética Usando Redes Neuronales Variational Autoencoder Y Conditional Variational Autoencoder

Sebastián Carmona Aguiar

Ingeniería de Sistemas

Andrés Eduardo Castro Ospina

Laura Stella Vega Escobar

INSTITUTO TECNOLÓGICO METROPOLITANO

2023

RESUMEN

Los datos acústicos se encuentran en muchas áreas de la vida y contienen un sinnúmero de información de alto valor para diferentes aplicaciones en la ciencia. Una vez los datos acústicos se digitalizan con dispositivos capaces de registrar la gama audible humana y ultrasónica, este potencial puede ser explorado y aprovechado, más aún con el poder que nos ofrece la inteligencia artificial para el análisis de datos e identificación de patrones. A pesar de todo ello, la grabación o recopilación de estos datos acústicos esconde muchos desafíos, ya sean por el esfuerzo logístico, limitaciones técnicas o de medios, que en su mayoría dejan brechas temporales sin información, incluso se dan fallos del dispositivo o por la misma técnica de grabación, que muchas veces se da por fracciones tiempo. Considerando estas situaciones, exploramos en este trabajo la posibilidad de generar información acústica sintética que se encuentre dentro del dominio de estudio para cubrir estos espacios de información, a partir de la implementación de redes neuronales autoencoder variacional (VAE) y autoencoder variacional condicional (CVAE), junto con las técnicas de generación por interpolación o basadas en etiquetas de clase. Para alcanzar nuestro objetivo se realizan pruebas sobre un conjunto de datos experimental de imágenes (MNIST) para posteriormente trabajar, con un conjunto de datos reales basado en información acústica de sonidos ambientales (UrbanSound8K). Durante este proceso se ejecutan tareas de caracterización embebida con el modelo pre-entrenado VGGish, optimización de hiperparámetros con el apoyo de la API Optuna, visualización y comprobación de resultados sobre espacios latentes mediante técnicas de visualización en baja dimensionalidad t-SNE. Medios que permitieron alcanzar el propósito de este trabajo e identificando cómo las arquitecturas VAE y CVAE, basadas en redes neuronales autoencoder, son modelos efectivos para esta aplicación en la generación sintética de información acústica dentro del dominio de estudio.

Palabras clave: autoencoders, generación de información sintética, incrustación de vecinos estocásticos distribuidos en t (t-SNE), optimización de hiperparámetros (Optuna), red pre-entrenada VGGish, sonidos ambientales.

RECONOCIMIENTOS

Este trabajo de grado es el resultado de un proceso de aprendizaje, investigación y desarrollo que no hubiera sido posible sin la colaboración y el apoyo de muchas personas e instituciones.

En primer lugar, quiero expresar mi más sincero agradecimiento a mis asesores de trabajo de grado, el profesor Andrés Eduardo Castro Ospina y la profesora Laura Stella Vega Escobar, por su valiosa orientación, sus conocimientos y sus observaciones, que me permitieron mejorar la calidad y el rigor de este trabajo. Su paciencia, dedicación y entusiasmo fueron fundamentales para llevar a cabo este proyecto.

Asimismo, quiero agradecer a todos los profesores universitarios que contribuyeron a mi formación profesional y personal, especialmente al profesor de inteligencia y visión artificial, cuyas enseñanzas fueron la base para el desarrollo de este trabajo. Gracias por compartir sus conocimientos, experiencias y consejos conmigo.

También quiero reconocer al programa de Jóvenes Investigadores, donde tuve la oportunidad de profundizar y explorar más a fondo los conceptos sobre máquinas inteligentes y reconocimiento de patrones. Gracias por brindarme las herramientas, los recursos y el espacio para desarrollar mis habilidades y capacidades como investigador.

Además, agradezco al proyecto *"Estimación de patrones de heterogeneidad espacial a partir del análisis acústico utilizando algoritmos de aprendizaje de máquina"* enmarcado en el programa No. 111585269779, *"Conservación Biológica usando Inteligencia Artificial"* financiado por la convocatoria 852 de Minciencias.

A nivel personal, quiero expresar mi profunda gratitud a mi padre, mi hermana y mis familiares por su apoyo incondicional, su paciencia y su motivación. Gracias por darme su tiempo, su amor y su confianza. Gracias por estar siempre presentes y por creer en mí.

Finalmente, quiero dedicar este trabajo a mi esposa, por su paciencia, su compañía, su visión y su amor. Gracias por acompañarme en este camino, por entender mis momentos de estrés, las largas noches, fines de semana en casa y cansancio, por celebrar mis logros y por animarme en los momentos difíciles. Gracias por ser mi inspiración y mi mejor amiga. Te amo.

ACRÓNIMOS

API	Application Programming Interface o Interfaz De Programación De Aplicaciones
CNN	Convolutional Neuronal Network o Red Neuronal Convolucional
CVAE	Conditional Variational Autoencoder
FC	Red Fully connected o Completamente conectada
GAN	Generative Adversarial Network o Red Generativa De Confrontación
GPU	Graphics Processing Unit o Procesador Gráfico
HDF	Hierarchical Data Format o Formato de Datos Jerárquicos
KL	Kullback-Leibler
MMG	Gaussian Mixture Model o Modelo de Mezcla Gaussiana
MNIST	Modified NIST o NIST Modificado
MSE	Mean Squared Error o Error Medio Cuadrático
NIST	National Institute of Standards and Technology o Instituto Nacional de Estándares y Tecnología
ReLU	Rectified Linear Unit o Unidad Lineal Rectificada
TPE	Tree-structured Parzen Estimator algorithm o Algoritmo Estimador de Parzen Estructurado en Árbol
t-SNE	t-Distributed Stochastic Neighbor Embedding o Incrustación de Vecinos Estocásticos Distribuidos en t
VAE	Variational Autoencoder
WAV	Waveform Audio File Format o Formato de Archivo de Audio de Forma de Onda

TABLA DE CONTENIDO

1. INTRODUCCIÓN

JUSTIFICACIÓN

Hoy en día se presentan grandes desafíos a la hora de recopilar u obtener datos acústicos, más aún cuando estos deben ser útiles para su procesamiento e insumo para las redes neuronales. Por una parte se encuentra la escasez, pues recopilar grandes volúmenes de datos acústicos implica un gran esfuerzo logístico, sumado a largas horas de grabación, situación que se agrava por el hecho de que el sonido es una señal altamente variable que se encuentra sujeta a diversos factores que pueden desfavorecer la calidad del audio, tales como: ruido, interferencia o distorsión e incluso en aspectos tan fundamentales como la posición del dispositivo de grabación o factores externos que no siempre pueden ser controlados, más aún en espacios abiertos, provocando retrocesos y pérdida de tiempo. Adicionalmente, los datos acústicos presentan la particularidad de que requieren ser etiquetados, en su mayoría de forma manual, con el fin de identificar los sonidos presentes en cada fragmento de audio, un proceso bastante tedioso, propenso a errores humanos e incluso costoso, ya que puede requerir de la participación de expertos en el dominio para obtener etiquetas lo más precisas posibles. Estas situaciones llevan a plantear la alternativa de producir datos acústicos de manera sintética que se encuentren dentro del dominio de estudio y que permitan ampliar el volumen del conjunto de datos disponible para trabajar, impactando positivamente en los entrenamientos de redes neuronales, los cuales parten de una base muy amplia de datos para generar información con mayor precisión.

PROBLEMA ABORDADO

Las redes autoencoder están desempeñado un papel importante en el desarrollo actual de la inteligencia artificial como modelos de aprendizaje profundo, en campos como la generación sintética de información de texto e imágenes, alcanzando grandes avances; así como también para la estimación de características, pues se aprovecha del gran poder que poseen estas redes en la codificación y generación de espacios latentes, que concentran los patrones esenciales de la fuente de datos. Este espacio latente es clave, por un lado, para la generación de nueva información, y por otro, como un espacio de trabajo bastante depurado desde el cual se espera obtener la detección o clasificación de datos con mayor precisión. Recientemente, se han presentado trabajos como el de Punam Bedi y Pushkar Gole que exponen un modelo híbrido de Conditional Variational Autoencoder (CVAE) y una Convolutional Neuronal Network (CNN) para identificar enfermedades en plantas, aprovechándose del espacio latente que produce CVAE y a partir de este, usan una CNN para clasificar (Bedi & Gole, 2021). En otro trabajo, se presenta un sistema de síntesis de voz expresiva de texto a audiovisual que aprende de un espacio latente de emociones utilizando un modelo generativo condicional CVAE, el cual fue capaz de generar matices de una emoción dada o de generar nuevas emociones que no existían en la base de datos (Dahmani et al., 2019). Versiones

especializadas del modelo Variational Autoencoder (VAE), que combina el modelo oculto de Markov para mejorar la generación de textos largos, alcanza mejoras al obtener un modelo más eficaz en el conjunto de datos con lógica fuerte, alivia problemas de colapso posterior y genera textos más largos, más continuos y conectados lógicamente (Zhao et al., 2021).

Así mismo, en la literatura se encuentran trabajos que aprovechan la posibilidad de estos modelos para el aumento de datos alcanzando mejoras en la robustez y la capacidad de generalización del sistema. A modo de ejemplo, el trabajo de Wu et al., busca mejorar la verificación por voz de la identidad de un cliente a través de segmentos hablados y que a partir del diseño de modelos CVAE que aprenden el patrón de voz, logran aumentar los datos con mayor diversidad, algo que de acuerdo a los autores, supera al enfoque de aumento de datos manual y a su anterior método de aumento de datos basado en el modelo de Generative Adversarial Network (GAN) sobre el conjunto de datos de evaluación estándar NIST SRE16 (Wu et al., 2019). O como el trabajo de Jane Saldanha et al., que presenta un elaborado artículo sobre el aumento de datos como herramienta para mejorar la identificación de enfermedades pulmonares, pues la fuente de información presenta alto desbalance de clases, lo cual impide que una red neuronal entregue resultados confiables. Al buscar alternativas que aumenten las clases minoritarias y al evaluar esto con el uso de diferentes modelos entre ellos VAE y CVAE para la generación sintética de esta información, se observan mejoras significativas en las métricas de desempeño de clasificación, recomendando este tipo de aplicaciones basados en aprendizaje profundo como una solución prometedora (Saldanha et al., 2022).

Logros como los hallados, más lo prometedor y positivo de los resultados que han demostrado estos modelos, permiten plantear la posibilidad de trabajar la implementación de modelos generativos en otros dominios. Un ejemplo, es la acústica ambiental con el dataset UrbanSound8K, ampliando la aplicación de las técnicas generativas y aprovechando su capacidad para suplir la escasez de datos que frecuentemente se presenta en diferentes áreas de investigación asociadas a la acústica. Otro ejemplo, la ecología y el estudio de paisajes sonoros, cuya recolección de datos presenta grandes desafíos por lo inhóspitos que pueden ser los ambientes y lo poco invasivos que deben ser los dispositivos, además de la intemperie y sucesos que no siempre se pueden controlar y que generan espacios de tiempo sin información, lo que afecta los modelos que usan esta fuente de información. Es allí donde técnicas generativas como VAE y CVAE pueden ayudar a recuperar datos de manera sintética y dentro del dominio de estudio.

OBJETIVO GENERAL

Generar información acústica sintética a partir del uso de técnicas basadas en las redes neuronales Variational Autoencoder y Conditional Variational Autoencoder.

OBJETIVOS ESPECÍFICOS

- Caracterizar datos acústicos usando modelo pre-entrenado VGGish para el análisis de técnicas profundas.

- Entrenar redes neuronales basadas en las técnicas Variational Autoencoder y Conditional Variational Autoencoder usando el conjunto de datos generado.
- Validar la generación de datos sintéticos con los modelos previamente entrenados usando datos experimentales y reales.

ORGANIZACIÓN DE LA TESIS

El desarrollo del presente trabajo se ha divido en cuatro secciones. La primera MARCO TEÓRICO, que se centra en la explicación de los diferentes conceptos usados para la realización de este trabajo, allí se definen aspectos como el sonido, redes convolucionales y las redes neuronales autoencoders VAE y CVAE, las bases de datos usadas como conjunto de datos experimentales y reales, más la API para optimización de hiperparámetros, las técnicas de visualización en baja dimensión, y de generación de información sintética acústica. En la sección dos, METODOLOGÍA, se detallan todos los aspectos acerca de la implementación realizada, desde la forma de extracción de características para cada conjunto de datos como la implementación de los modelos autoencoders, la configuración para la búsqueda y optimización de hiperparámetros; finalizando esta sección, se encontrarán detalles sobre la implementación para la generación de datos sintéticos con cada una de las técnicas según la red autoencoder, más las definiciones sobre la técnica de validación de los resultados. En la sección tres, RESULTADOS Y DISCUSIÓN, se encuentran los diferentes productos de las implementaciones mencionadas en la sección anterior, aquí se podrá observar la representación en baja dimensión t-SNE tanto del aprendizaje de las redes autoencoders en el conjunto de datos experimentales como también en los reales, así mismo los resultados sobre la generación de datos, y entre ellos, se encuentran los resultados de la optimización de hiperparámetros para cada una de las redes, más algunos puntos de cuestión sobre las representación en baja dimensión de los espacios latentes para la red CVAE. Finalmente, en la sección cuarta se exponen las CONCLUSIONES, RECOMENDACIONES Y TRABAJO FUTURO.

2. MARCO TEÓRICO

SONIDO

Forma de energía producida por vibraciones en el aire u otros medios de propagación, como el agua o los sólidos. Estas vibraciones generan ondas que se propagan a través del medio hasta ser percibidas por nuestros oídos. Un fragmento de audio posee las siguientes propiedades:

- La intensidad de las vibraciones o lo que se conoce como la Amplitud: se refiere a la distancia máxima desde su punto de equilibrio, que alcanza una partícula del medio de propagación cuando se produce una vibración. Una mayor amplitud significa que las partículas se mueven con mayor desplazamiento y, por lo tanto, el sonido es más intenso o fuerte, así como también una menor amplitud se refiere a un menor desplazamiento o un sonido más suave. Esta amplitud muchas veces es calculada en decibelios (dB), medida que se relaciona con el nivel de volumen percibido por el oído humano.
- El número de ondas hechas por la señal en un segundo se llama Frecuencia: se mide en hercios (Hz), donde un hercio equivale a un ciclo por segundo. La frecuencia está relacionada con la percepción de los diferentes tonos. Una frecuencia baja produce un sonido grave, mientras que una frecuencia alta genera un sonido agudo. Generalmente el rango audible de frecuencia para los humanos abarca desde 20 Hz a 20 kHz (Moebs et al., 2021).

Es importante considerar que los sonidos pueden ser composiciones de diferentes frecuencias a diferentes amplitudes, que se suman para crear señales compuestas con patrones más complejos y diversos.

Para procesar el sonido en un computador, primero debe ser convertido a un formato digital, es decir, a una secuencia de números binarios que representen las variaciones de la señal sonora. Para ello se usan los micrófonos, los cuales poseen una membrana que vibra o se mueve al recibir las ondas sonoras. Esta vibración genera una señal eléctrica que cambia según la frecuencia y la amplitud del sonido percibido. Esta señal es enviada a un conversor digital que transforma esta señal análoga a secuencias de números binarios representando el voltaje y la presión sonora en cada instante. Este proceso recibe el nombre de muestreo del sonido, donde se toman pequeñas muestras de la señal análoga a intervalos regulares de tiempo, por lo que cada muestra se codifica a bits determinados; por lo tanto, a mayor frecuencia de muestreo, mayor fidelidad del audio original.

En la Figura 1 a continuación, se puede observar esta técnica de muestreo y su relación con la calidad del audio. La línea azul representa la señal de audio análoga u original para un audio de 10 segundos de duración, sobre esta línea azul se posicionan unos marcadores en forma de triangulo invertido

que representa las muestras tomadas a razón del eje horizontal, donde la primera imagen muestrea la señal de audio cada 0.5 segundos en cambio, la segunda cada 0.25 segundos. De esta forma se puede observar que a mayor frecuencia de muestreo se puede obtener mayor información de la señal de audio y por ende mayor fidelidad de digitalización del audio original.

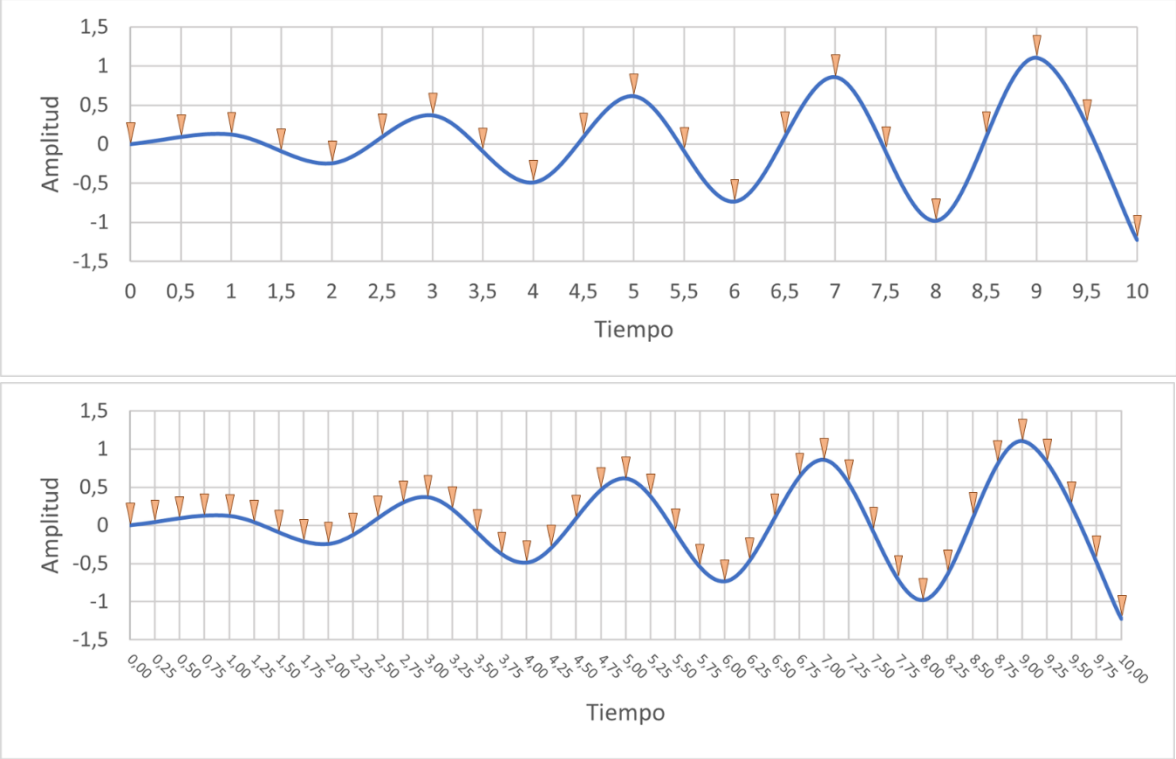


Figura 1. Comparación frecuencia de muestreo

ESPECTROGRAMA

Un espectrograma de audio es la representación gráfica de las frecuencias y las amplitudes de un sonido, es como una "fotografía" de la señal (Doshi, 2021). Se puede obtener a partir del análisis espectral que consiste en aplicar una transformada de Fourier a la señal de audio y así descomponerla en sus frecuencias fundamentales y armónicas.

El espectrograma se presenta como una imagen bidimensional, donde el eje horizontal representa al tiempo y el eje vertical a la frecuencia o espectro de la señal, los colores y su intensidad es la amplitud o energía de cada frecuencia. Gracias a esto, se puede observar cómo evoluciona en el tiempo el espectro de audio y obtener de allí patrones, variaciones e incluso características del sonido.

En la Figura 2, se presenta una señal de audio en dos espacios, la primera en el dominio del tiempo (Forma de la Onda), donde se puede observar qué tanta energía o silencio posee un audio en cualquier momento. Y la segunda, pertenece al espectrograma en el dominio tiempo-frecuencia (Espectrograma), presentando los diferentes patrones que componen la misma señal de audio.

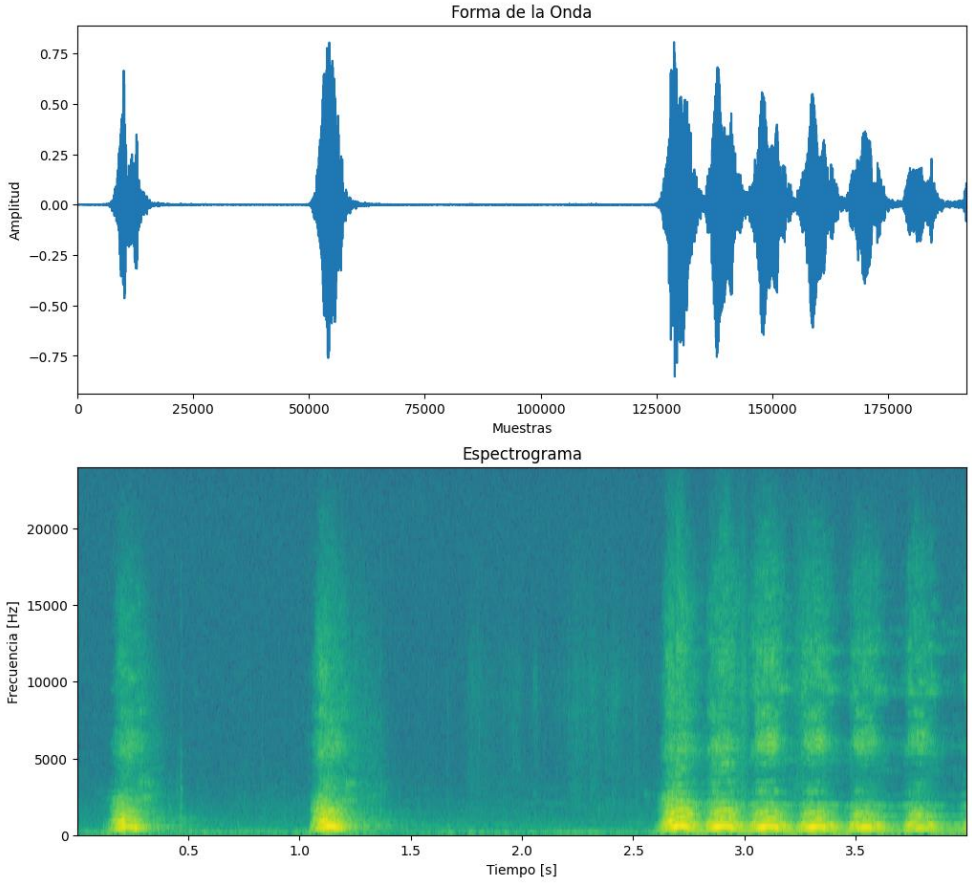


Figura 2. Espectrograma de una señal de audio

REDES CONVOLUCIONALES

Las redes neuronales convolucionales o CNN corresponden a un tipo de red neuronal compuesta por múltiples capas encargadas de realizar operaciones de convolución, es decir, operaciones matemáticas para extraer características importantes de los datos de entrada. “Las CNN proporcionan un enfoque más escalable para las tareas de clasificación de imágenes y reconocimiento de objetos, aprovechando principios del álgebra lineal, específicamente la multiplicación de matrices, para identificar patrones dentro de una imagen” (IBM, s/f).

Estas redes frecuentemente se componen de tres partes: capa de entrada, capas ocultas y capa de salida. La capa de entrada es la imagen original sin ser procesada, la capa de salida es el resultado de clasificar las características, y las capas ocultas se componen de neuronas con una compleja estructura, que puede incluir capas de convolución y de submuestreo (Tian, 2020).

En una arquitectura CNN, se encuentran varias capas convolucionales y capas de agrupación que en principio se usan para simplificar la complejidad de los datos de entrada y reducir su tamaño. Lo que hace de este conjunto muy útil para extraer patrones complejos (Yalçın, 2021b).

"Las primeras capas se enfocan en características simples, como colores y bordes. A medida que los datos de la imagen avanzan a través de las capas de CNN, se comienzan a reconocer elementos o formas más grandes del objeto, hasta que finalmente se identifica el objeto previsto" (IBM, s/f).

ARQUITECTURA DE UNA CNN

Capa de convolución

También llamadas capas ocultas, en ella se realiza la tarea de convolución de la imagen de entrada con un filtro o kernel. "Es el pilar central de una CNN, y es donde ocurre la mayor parte del cálculo" (IBM, s/f).

En esta capa se toman grupos de valores cercanos de la matriz de entrada, luego se calcula el producto punto entre este grupo y el filtro (convoluciones). Esta operación se repetirá, desplazando la selección del grupo de valores cercanos hasta cubrir la matriz de entrada. Como resultado se obtendrá una nueva matriz llamada mapa de características. La Figura 3, presenta un ejemplo de esta tarea con un filtro de convolución de 3 × 3 sobre una matriz de intensidad de pixeles.

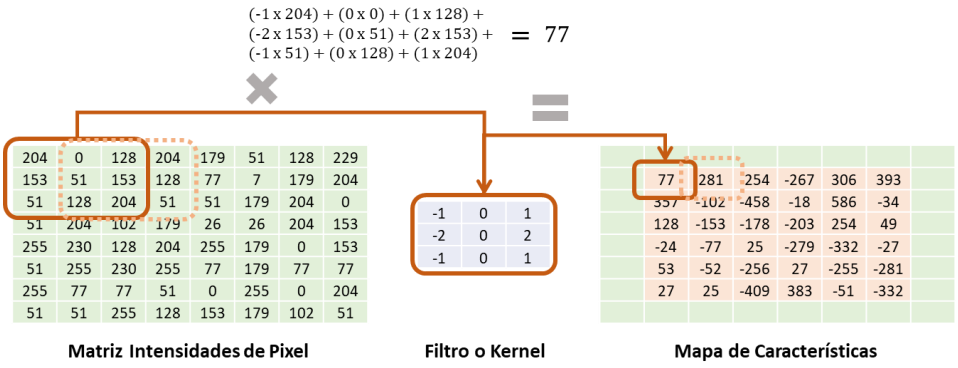


Figura 3. Ejemplo de convolución a una imagen con un filtro 3x3

Los filtros o kernel son matrices de pesos sintonizables que se aplican sobre una región de la entrada para extraer características relevantes como bordes, colores, formas hasta características más complejas como texturas, patrones, objetos o rostros.

El kernel o "detector de características es una matriz bidimensional (2-D) de pesos, que representa parte de la imagen. Aunque su tamaño puede variar, suele ser una matriz 3 × 3; esto también determina el tamaño del campo receptivo." (IBM, s/f).

Función de activación

Después de la convolución se aplica al mapa de características una función de activación, la cual añade factores no lineales a la red neuronal para que ésta pueda adaptarse a problemas más complejos (Tian, 2020). También, aumenta la capacidad de la red neuronal para capturar información relevante y suprimir la que no lo es. Una red neuronal sin una función de activación es esencialmente un modelo de regresión lineal (Yalçın, 2021a).

Algunas funciones de activación son las siguientes:

- **Sigmoide:** se utiliza para predecir una probabilidad como salida y su rango de valores varía entre [0,1]. Funciona mejor en tareas de clasificación binaria (Yalçın, 2021a). La Ecuación (1) representa esta función.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{1}$$

- **Tangente hiperbólica (Tanh):** tiene forma similar a la función sigmoide, aunque el rango es [-1,1]. La ventaja es que los valores cero se mapearán cerca de cero, y los valores negativos se mapearán fuertemente negativos (Naranjo Torres et al., 2020). A continuación, la Ecuación (2) que representa la función.

$$tanh(x) = \frac{2}{1 + e^{-2x}} - 1 \tag{2}$$

- **ReLU (Rectified Linear Unit):** de propósito general y ampliamente utilizada (Yalçın, 2021a). Rango de valores varía entre [0,1]. Es una función lineal por partes cuya salida es la entrada si es positiva, o cero si es negativa. Ver Ecuación (3).

$$f(x) = \max(0,1) = \begin{cases} 0 \; si \; x < 0 \\ x \; si \; x \geq 0 \end{cases} \tag{3}$$

- **Leaky ReLU:** la función Leaky ReLU modifica el comportamiento de la función ReLU para valores negativos, multiplicándolos por un pequeño coeficiente rectificador α, que suele ser 0.01; evitando el problema de los gradientes nulos que puede ocurrir con la función ReLU cuando la entrada es negativa. Ver Ecuación (4).

$$f(x) = \max(\alpha x, x) = \begin{cases} \alpha x \; si \; x < 0 \\ x \; si \; x \geq 0 \end{cases} \tag{4}$$

Tal y como se observa en la Figura 4 a continuación, la función de activación no lineal produce los mapas de activación, que contienen sólo las características activadas (celdas color naranja) que resultaron al usar la función de activación ReLU sobre el mapa de características de la Figura 3, cuyos valores por debajo de cero se reemplazan por cero y se obtiene la matriz de la derecha, que pasarán a la siguiente capa de la red.

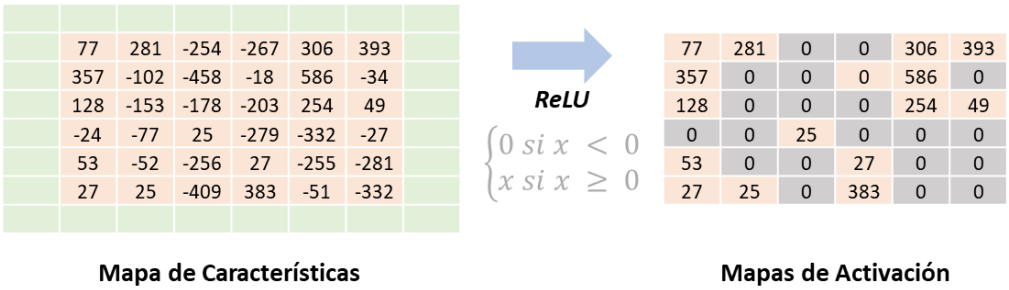


Figura 4. Ejemplo función de activación ReLU

Capa de agrupación

Una vez se cuenta con el mapa de activación, frecuentemente se pasa a la capa de agrupación. La capa de agrupación submuestrea el mapa y reduce el tamaño de los datos de entrada para reducir la carga computacional, el uso de memoria y el número de parámetros. Al reducir el número de parámetros que debe procesar la red, también se limita el riesgo de sobreajuste. El resultado de la capa de agrupamiento es un mapa de características agrupado (Paper, 2021).

La capa de agrupamiento es también un tipo muy común de capa oculta utilizada en las CNN. Dado que las características locales están relacionadas, la puesta en común puede reducir en gran medida la cantidad de cálculos, pero no perderá las características principales de la imagen (Tian, 2020).

Las principales razones para utilizar la capa de agrupamiento son: realizar un procesamiento de reducción de muestreo y dimensionalidad, reduciendo así la carga de computación de la red, realizar la invariancia de escala, invariancia de traslación e invariancia de rotación de la imagen de entrada, y hacer que el mapa de características de salida sea más robusto a la distorsión y el error de una sola neurona (Chen et al., 2021).

Algunas funciones de agrupación son las siguientes:

- **Max pooling:** La capa de agrupación máxima conserva el valor máximo al deslizar el filtro sobre la entrada. Por lo general, se aplican filtros de 2 × 2 con un paso de 2 (Naranjo Torres et al., 2020).
- **Average pooling:** La capa de agrupación promedio reduce la activación convolucional dividiendo la entrada en regiones de agrupación y calculando sus valores promedio (Naranjo Torres et al., 2020).

A continuación, la Figura 5 presenta un ejemplo aplicando las funciones de agrupación antes mencionadas sobre los mapas de activación obtenidos en la Figura 4. A la derecha se encuentran los resultados de aplicar un filtro 2 × 2 con la función de agrupación max y average pooling

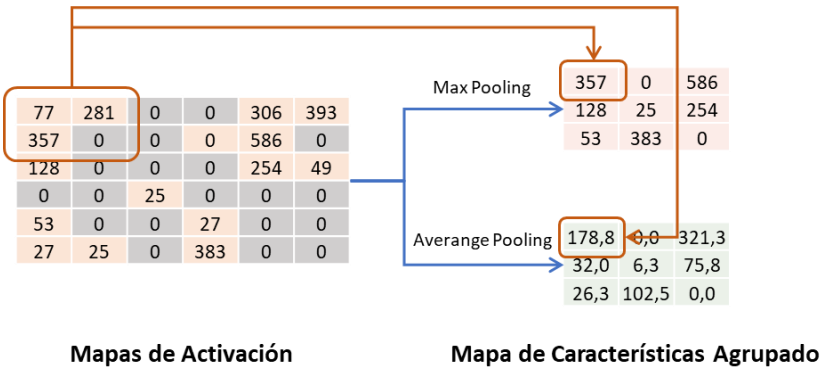


Figura 5. Ejemplo funciones de agrupación

Capa totalmente conectada

Se encarga de clasificar los resultados de las capas anteriores para definir su etiqueta de clase. La función de activación de la última capa calcula las probabilidades finales para cada clase y generalmente se selecciona de acuerdo a la probabilidad más alta. Normalmente, una tarea de clasificación de múltiples clases utiliza la función Softmax, donde el valor de probabilidad de cada clase oscila entre [0,1] y su suma total es igual a 1. Finalmente, cada neurona de salida decide sobre cada una de las etiquetas, y la de mayor valor de salida corresponde a la decisión de clasificación (Naranjo Torres et al., 2020).

La capa totalmente conectada contiene vectores de pesos y sesgos que se multiplican y suman por el vector de entrada, generando un puntaje. El vector de entrada corresponde al mapa de características agrupado que se redimensiona como un vector. Cada peso representa la fuerza de la conexión entre las neuronas y cada sesgo representa el umbral de activación de una neurona, ambos son ajustados durante el proceso de entrenamiento de la red buscando minimizar el error de salida. Con esto se implementa la función de activación Softmax (ver Ecuación (*5*)) y de acuerdo a ello, determinar la clase que mejor representa ese vector de entrada.

$$\hat{y} = \frac{e^{S_i}}{\sum_{j=1}^{k} e^{S_j}}, i = 1, \dots, k \tag{5}$$

El numerador de la ecuación anterior toma cada valor calculado del puntaje (el resultado del vector de entrada operado con cada uno de los pesos y sesgos) y lo usa como potencia para el exponencial; k es la cantidad de puntajes calculados y el denominador representa la sumatoria de los resultados de la expresión anterior.

La Figura 6, presenta un ejemplo del proceso realizado en la capa totalmente conectada para determinar la clase que mejor representa el vector de entrada.

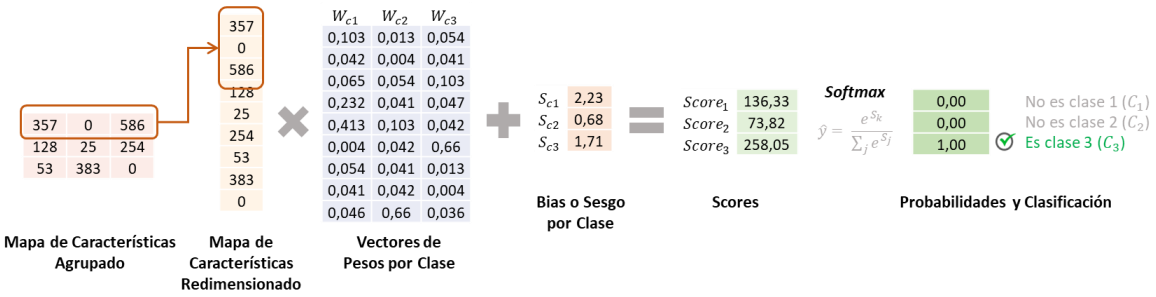


Figura 6. Ejemplo definición de clase

AUTOENCODERS

Las redes autoencoder son modelos de aprendizaje profundo no supervisado desarrollados como método para la generación sintética de información, como también para la detección de características. Tienen la capacidad de aprender, reducir y reconstruir datos de entrada, pues se aprovecha el gran poder que poseen estas redes en la codificación y generación de espacios latentes. Los espacios latentes se pueden considerar como una representación que condensa los patrones esenciales de la fuente de datos. Este espacio latente es clave para generar nueva información, pues la red parte de los patrones aprendidos y se desplaza entre ellos generando información sintética dentro del dominio de información. Por otro lado, este espacio latente es un conjunto de datos bastante depurado desde el cual, se puede esperar mayor eficiencia a la hora de usarse en tareas de detección o clasificación de datos.

Los autoencoders están compuestos por dos redes, un codificador o encoder e y un decodificador o decoder d:

El encoder aprende una transformación no lineal e: $X \rightarrow z$ que proyecta los datos desde el espacio de entrada original de alta dimensión X a un espacio latente de menor dimensión z. Es decir, $z = e(X)$ es el vector latente. Un vector latente es una representación de baja dimensión de un punto de datos que contiene información sobre X. La transformación e debe tener ciertas propiedades, como valores similares de X deben tener vectores latentes similares (y valores disímiles de X deben tener vectores latentes distintos) (Van de Kleut, 2020).

Un decoder aprende una transformación no lineal d: $z \rightarrow X$ que proyecta los vectores latentes de vuelta al espacio de entrada original de alta dimensión X. Esta transformación debe tomar el vector latente $z = e(X)$ y reconstruir los datos de entrada originales $\hat{x} = d(z) = d\big(e(X)\big)$ (Van de Kleut, 2020).

La Figura 7, presenta el diseño general de una red autoencoder y la comunicación entre sus capas (Encoder y Decoder).

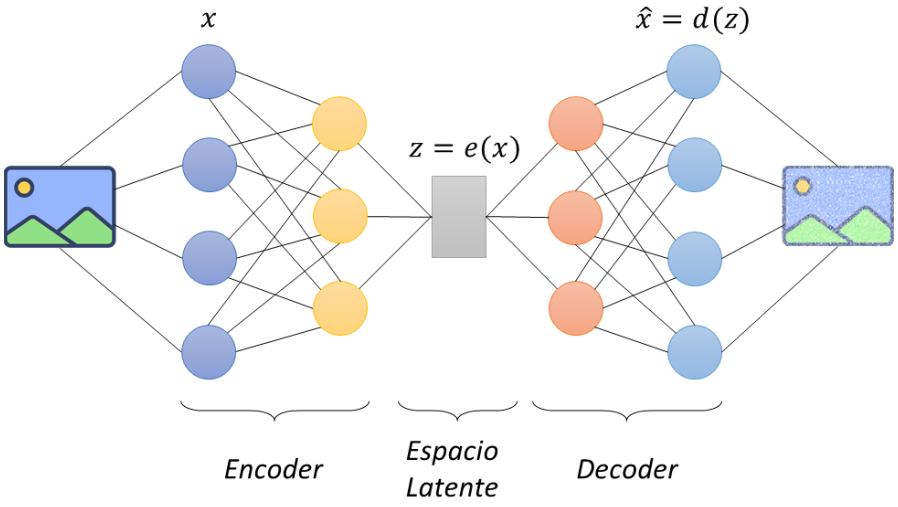


Figura 7. Diseño de una red Autoencoder

Nota: la imagen anterior presenta el diseño de la red para la codificación de una imagen o entrada x a un espacio latente $z = e(x)$ y su decodificación $\hat{x} = d(z) = d\big(e(x)\big)$. Adaptado de: (Van de Kleut, 2020).

Un autoencoder es la composición del codificador y el decodificador $f(x) = d\big(e(X)\big)$ que se entrena para minimizar la diferencia entre la entrada *X* y la reconstrucción $\hat{x}$ utilizando una función de pérdida de reconstrucción. El autoencoder se entrena como un todo (se dice que que se entrena "de extremo a extremo"), por lo que se optimizan simultáneamente el codificador y el decodificador (Van de Kleut, 2020).

Variational Autoencoder – VAE

Es un modelo de aprendizaje no supervisado que mezcla las redes neuronales con distribuciones de probabilidad. Esta aplicación busca solucionar problemas de vacíos en el espacio latente generado por los autoencoder y que suele ser un inconveniente a la hora de producir nueva información, pues el resultado puede no tener ningún sentido (Van de Kleut, 2020).

En los autoencoders tradicionales, las entradas se asignan de forma determinista a un vector latente $z = e(X)$. Mientras que, con los VAE, las entradas se asignan a una distribución de probabilidad sobre vectores latentes, y que luego se muestrea un vector latente a partir de esa distribución. Como resultado, el decodificador es más robusto a la hora de decodificar vectores latentes (Van de Kleut, 2020).

Las VAE consta de tres componentes principales: un codificador, un decodificador y una función de pérdida. El codificador y el decodificador cumplen las mismas funciones ya definidas previamente para los autoencoders. Adicionándose el cálculo de una distribución de probabilidad, útil para medir la similitud entre los datos de entrada X y la salida $\hat{x}$ y penalizar la red cuando se entreguen salidas muy distintas a X, buscando minimizar la diferencia entre sí.

El espacio latente de VAE es continuo, ya que en este diseño el decodificador genera dos vectores, un vector de medias μ y un vector de deviación estándar σ, donde μ controla el centro aproximado en el que una entrada debe ser codificada, mientras que σ controla cuánto se puede desviar de ese centro (Saldanha et al., 2022). Estos vectores en conjunto forman el espacio latente z, a partir de calcular entre ellos una distribución normal estándar $z \sim N(\mu, \sigma)$. Esta aplicación además de generar un espacio latente continuo, contribuye a mejorar el rendimiento y facilitar el proceso de decodificación. La Figura 8, presenta la estructura antes expuesta sobre la red VAE.

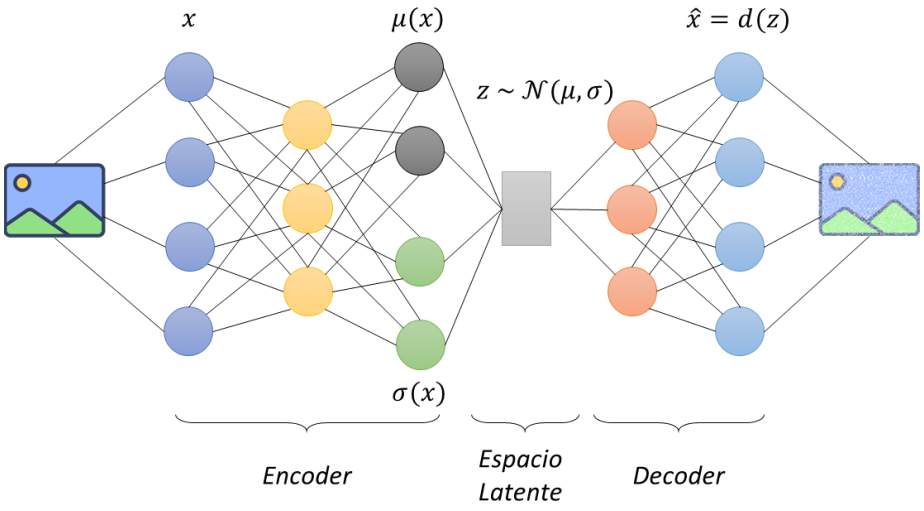


Figura 8. Diseño de una red Variational Autoencoder

Nota: la imagen anterior presenta el diseño de la red para la codificación de una imagen o entrada x, donde se generan dos vectores: medias μ y desviación estándar σ, a partir de estos se crea el espacio latente continuo $z \sim N(\mu, \sigma)$, y como salida el espacio latente es decodificado $\hat{x} = d(z)$. Adaptado de: (Van de Kleut, 2020).

La función de pérdida de VAE es una medida que combina dos términos:

- **Divergencia Kullback-Leibler (KL):** es una medida asimétrica de distancia entre dos densidades de probabilidad y regulariza la posterior, fomentando la proximidad a la anterior (Gibb et al., 2023). Se encarga de medir la diferencia entre el vector de medias y deviación estándar.
- **Pérdida de reconstrucción:** busca contener los datos generados dentro del dominio de los datos de entrada, previniendo que los datos aprendidos tomen valores demasiado probabilísticos y por ende tiendan a ser iguales. Algunas funciones aplicadas para el cálculo son error medio cuadrático (MSE) o entropía cruzada binaria.

Estos dos términos actúan como dos fuerzas que buscan incrementar el aprendizaje de la red cuyo objetivo es conservar los datos dentro de un espacio latente continuo, pero que al mismo tiempo sea congruente con el dominio de los datos de entrada. Gracias a ello el codificador y el decodificador aprenden a comprimir y reconstruir los datos de forma eficiente y variada.

La función de pérdida se minimiza mediante un algoritmo de optimización basado en gradiente descendente, como el Adam. Adam es una de las herramientas más usadas actualmente por lo poderosa y sencilla que es, permitiendo hacer búsquedas en menor tiempo y pasos, y trayendo ventajas de otros algoritmos basados en gradiente descendente y momentos (Rojano Aguilar et al., 2021). Es un algoritmo para la optimización basada en gradientes de primer orden de funciones objetivo estocásticas, basado en estimaciones adaptativas de momentos de orden inferior. Es computacionalmente eficiente, tiene pocos requisitos de memoria, es invariante al re-escalado diagonal de los gradientes, y es muy adecuado para problemas que son grandes en términos de datos o parámetros (Kingma & Lei Ba, 2014).

Conditional Variational Autoencoder – CVAE

CVAE corresponde a una versión adaptada de VAE que busca tener mayor control sobre el tipo de datos que genera, CVAE propone la inclusión de la etiqueta y a los datos de entrada. Lo que enriquece el modelo al promover en el encoder al aprendizaje de este valor junto con sus características, relacionándolos entre sí. A la hora de solicitar al decoder la proyección del vector latente, se podrá especificar una y, cuyo resultado será una matriz de datos que se encuentran en el dominio de dicha etiqueta. Logrando así controlar los datos generados. A la hora del entrenamiento de CVAE, esta conserva las características ya mencionadas para VAE, a razón del cálculo de la función de perdida, compuesta por el cálculo de KL y la pérdida de reconstrucción. En la Figura 9, se pueden observar los cambios que sufre el diseño original de una red VAE al adicionar la etiqueta de clase y.

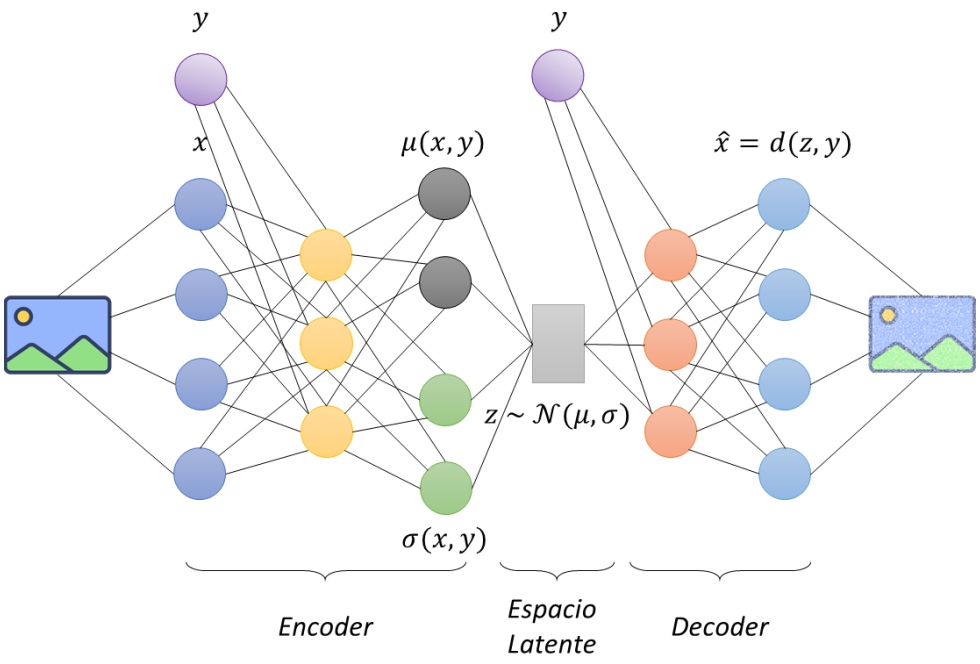


Figura 9. Diseño de una red Conditional Variational Autoencoder

Nota: la imagen anterior presenta el diseño de la red, que incluye la entrada de los datos x y de sus etiquetas y, la codificación de estas entradas (x, y), la producción del espacio latente $z \sim N(\mu, \sigma)$, y la decodificación, que nuevamente incluye las etiquetas y para generar datos $\hat{x} = d(z, y)$. Adaptado de: (Van de Kleut, 2020).

VGGish

Es una red neuronal convolucional pre-entrenada, creada por Google y usada para extraer características de audio. El modelo VGGish es una red derivada de la red CNN Visual Geometry Group (VGG16) entrenada con un gran conjunto de datos de YouTube. Su estructura incluye ocho capas convolucionales, cinco capas de agrupación y tres capas completamente conectadas. Cada capa convolucional utiliza un núcleo de convolución de 3x3. VGGish convierte la función de entrada de audio en un vector de 128 dimensiones de alto nivel y semánticamente significativo, que puede alimentarse como entrada a un modelo de clasificación posterior (Di et al., 2023).

VGGish se entrenó con el AudioSet de Google, una biblioteca de sonidos compuesta por audio extraído de millones de vídeos de YouTube. El audio de YouTube se codifica mediante Advanced Auto Coding (AAC), un algoritmo de compresión con pérdida que codifica hasta un máximo de 192 kbps en una amplia gama de frecuencias de muestreo de 8-96 kHz (Gibb et al., 2023).

De manera general, VGGish divide la señal de audio en ventanas consecutivas de 0.96 segundos, lo cual es una técnica utilizada para analizar señales de audio en pequeñas secciones de tiempo. Por cada uno de estos fragmentos se genera un espectrograma sobre el cual, la red extrae sus características. El formato de salida de la red es [H, 128]. H se expresa como $H = \frac{t}{0.96}$, donde, t es la duración de la señal y 0.96 es la duración de cada espectrograma Mel (Qiu et al., 2023).

BASES DE DATOS

MNIST

Es un recurso del Instituto Nacional de Estándares y Tecnología (NIST), que consiste en un conjunto de imágenes de números en sistema decimal escritos a mano (ver Figura 10), que se utilizan ampliamente en el reconocimiento óptico de caracteres y la investigación del aprendizaje automático (Deng, 2012).

La base de datos MNIST, se ha convertido en un estándar para probar rápidamente algoritmos de aprendizaje automático. Se construyó a partir de la base de datos NIST original; de ahí lo de NIST modificado o MNIST. Hay 60.000 imágenes de entrenamiento (algunas de estas imágenes de entrenamiento también se pueden utilizar con fines de validación cruzada) y 10.000 imágenes de prueba, ambas extraídas de la misma distribución. Todos estos dígitos en blanco y negro, están normalizados en tamaño y centrados en una imagen de tamaño fijo en la que el centro de gravedad de la intensidad se encuentra en el centro de la imagen con 28 x 28 píxeles (Deng, 2012).

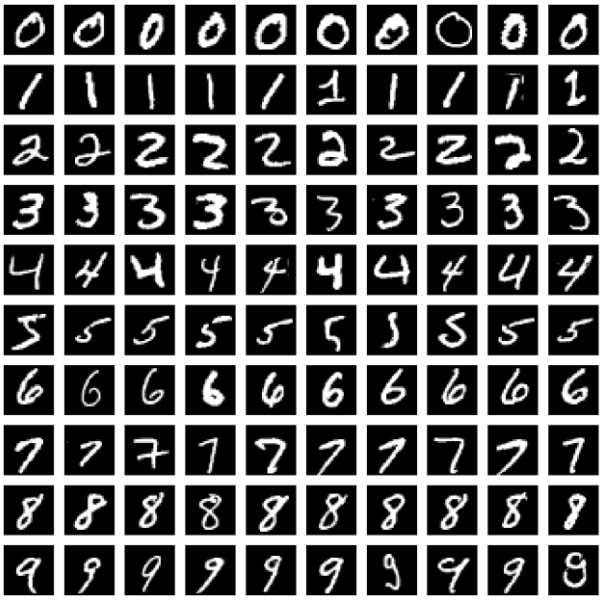


Figura 10. Muestra del conjunto de datos MNIST

UrbanSound8K

Es un conjunto de datos que contiene 8.732 fragmentos sonoros etiquetados, de duración menor a 4 segundos en formato WAV, que incluye sonidos ambientales urbanos de 10 clases. Las clases están definidas como dígitos del 0 al 9, representando los siguientes valores:

- 0 = air_conditioner
- 1 = car_horn
- 2 = children_playing
- 3 = dog_bark
- 4 = drilling
- 5 = engine_idling
- 6 = gun_shot
- 7 = jackhammer
- 8 = siren
- 9 = street_music

Las clases se extraen de la taxonomía del sonido urbano. Las taxonomías se comienzan a definir a partir de 4 grupos de nivel superior: humanos, naturaleza, mecánicos y musicales (Salamon et al., 2014). En la Figura 11, podrá observarse la distribución de las clases sobre el conjunto total de datos.

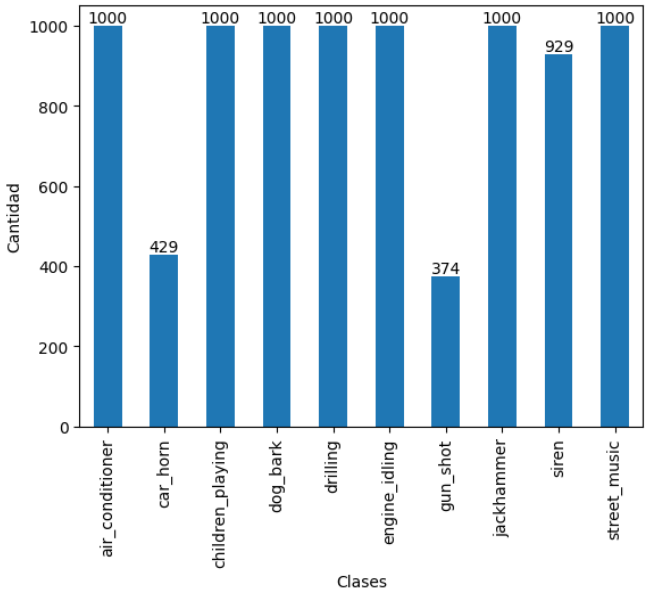


Figura 11. Audios por clase en el dataset UrbanSound8K

Los archivos de audio están preclasificados en diez carpetas (denominadas fold1 hasta fold10) para facilitar la reproducción y la comparación con los resultados de la clasificación automática (Salamon et al., s/f).

En la Figura 12, se puede observar cómo se distribuyen los audios de este conjunto de datos, según su duración por rango de tiempo en segundos.

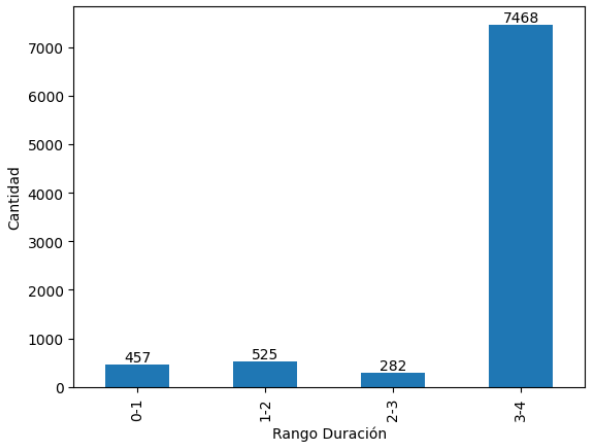


Figura 12. Rango duración de los audios en el dataset UrbanSound8K

OPTIMIZACIÓN DE HIPERPARÁMETROS Y VISUALIZACIÓN DE RESULTADOS

Optuna

Es una API que permite a los usuarios construir el espacio de búsqueda de hiperparámetros de forma dinámica, que incluye una implementación eficiente y de estrategias de búsqueda con poda para proyectos de aprendizaje automático. La poda es la monitorización del resultado intermedio de cada ensayo y su eliminación prematura donde los ensayos sean poco prometedores, para acelerar la exploración y optimización de rendimiento en caso de una baja disponibilidad de recursos. Otra forma de acelerar el proceso de optimización es la distribuida, permitiendo el procesamiento en paralelo de múltiples ensayos (Akiba et al., 2019).

Optuna formula la optimización de hiperparámetros como un proceso de minimizar/maximizar una función objetivo, que toma un conjunto de hiperparámetros como entrada y devuelve su puntuación. Esta función construye dinámicamente el espacio de búsqueda de la arquitectura de la red neuronal (el número de capas, el número de unidades ocultas, learning rate u otros hiperparámetros) sin depender de variables estáticas definidas externamente. Optuna se refiere a cada proceso de optimización como un estudio, y a cada evaluación de la función objetivo como un ensayo (Akiba et al., 2019).

La publicación de esta API como documentación para su instalación y uso se encuentra bajo la siguiente dirección web Optuna.

t-SNE

t-Distributed Stochastic Neighbor Embedding o incrustación de vecinos estocásticos distribuidos en t, es una técnica de reducción de dimensionalidad para visualizar datos de alta dimensión en un espacio de baja dimensión, desarrollada por Laurens van der Maaten y Geoffrey Hinton en 2008. Esta técnica toma los puntos que están cerca en el espacio de alta dimensión manteniéndolos en el espacio de baja dimensión, para ello usa una distribución de probabilidad que mide la similitud entre dos puntos, para luego minimizar la diferencia en esta distribución y una distribución similar en el espacio de baja dimensión.

Es una técnica de visualización datos de alta dimensión basado en los métodos de reducción de dimensionalidad que convierten el conjunto de datos de alta dimensión $X = \{x_1, x_2, \dots, x_n\}$ en datos bidimensionales o tridimensionales $Y = \{y_1, y_2, \dots, y_n\}$ que se puede mostrar en un diagrama de dispersión. El objetivo de la reducción de dimensionalidad es preservar la mayor cantidad posible de la estructura significativa de los datos de alta dimensión en el mapa de baja dimensión. t-SNE es capaz de capturar muy bien gran parte de la estructura local de los datos de alta dimensión, al mismo tiempo que revela una estructura global, como la presencia de grupos en varias escalas (Van Der Maaten & Hinton, 2008).

GENERACIÓN DE DATOS

Generación con VAE: Interpolación

La técnica de interpolación se basó en la publicación de (Van de Kleut, 2020), donde se presenta una alternativa para generar información sintética con la red autoencoder VAE dada dos entradas x_1 y x_2, a las cuales se les genera su correspondiente vector en el espacio latente z_1 y z_2 con el encoder, y sobre los que se puede interpolar u obtener nuevos puntos entre ellos, decodificando vectores latentes intermedios entre z_1 y z_2, estos nuevos puntos corresponden a nuevos datos sintéticos generados en el dominio de los puntos dados, ver Ecuación (6) a continuación.

$$\hat{x} = z_1 + (z_2 - z_1) \cdot t \quad t \in [0, 1] \tag{6}$$

Generación con CVAE: Etiquetas de Clase

La generación sintética de información con CVAE fue basa en la publicación de (Raschka, s/f) donde se encuentra una implementación mediada por la generación de dos tensores, uno de ellos representa las etiquetas de clase cuyo tamaño es la cantidad datos a generar; por otra parte, se genera un espacio latente aleatorio basado en la creación de un tensor con números aleatorios dentro de una distribución normal estándar, donde su tamaño será la cantidad de datos a generar y el tamaño del espacio latente al que codifica la red los datos de entrada. Estos dos tensores se le entregan al decoder de CVAE, regresando una matriz de características que se encuentran en el dominio de valores de las etiquetas entregadas, completando así la tarea de generación de datos sintéticos.

3. METODOLOGÍA

El desarrollo del presente trabajo se ha divido en cuatro partes como se muestra en la Figura 13. La primera parte, corresponde a la carga y procesamiento para extracción de características del conjunto de datos reales y controlados. La segunda, es la implementación de las redes neuronales autoencoders VAE y CVAE. En la tercera, se inician actividades de optimización de las redes con la ayuda de Optuna y de esta forma hallar los mejores hiperparámetros para cada una de ellas. Finalmente, en la cuarta parte se realiza la validación de las técnicas implementadas generando nueva información y contrastándola con los datos base, mediante la técnica de visualización de datos t-SNE.

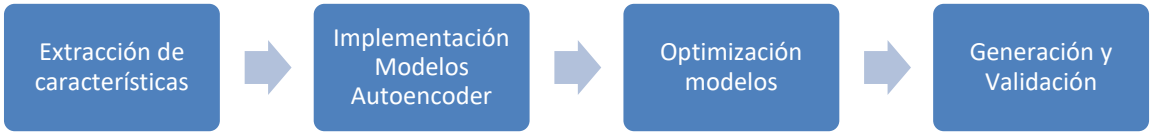


Figura 13. Metodología de desarrollo

EXTRACCIÓN DE CARACTERÍSTICAS

MNIST

Es el conjunto de datos controlado que contiene 60.000 imágenes de 28x28 píxeles de números escritos a mano y es un conjunto ampliamente usado en investigaciones de aprendizaje automático. Se cuenta con la ventaja de que este conjunto ya se está procesado y depurado, el tamaño de las imágenes se encuentra normalizado y los dígitos están centrados en el espacio de la imagen. Además de esto, es una base de datos de fácil acceso, que se accede mediante la librería `torchvision`[1], que facilita la información ya normalizada en el rango de 0 y 1, representando las intensidades de píxel de cada imagen.

Como ya se ha mencionado, este conjunto de datos está formado por imágenes en blanco y negro, lo que significa que por cada imagen obtenemos una matriz bidimensional de intensidades de píxel del tamaño 28 x 28 pixeles, esta matriz bidimensional fue aplanada logrando un vector de 784 valores por cada imagen. Estos vectores aplanados por imagen fueron apilados formando la matriz de características.

El conjunto de datos MNIST fue usado para tareas de implementación y validación rápida visual de resultados. En este sentido, para comparar el aprendizaje alcanzado por las redes VAE y CVAE sobre

[1] https://pytorch.org/vision/stable/generated/torchvision.datasets.MNIST.html

este conjunto de datos, se generó una visualización de baja dimensión usando la técnica t-SNE sobre el 30 % de las imágenes del conjunto de datos MNIST, tomadas de forma aleatoria sobre el conjunto de datos completo. Es decir, se representaron en un espacio de baja dimensión las 784 características de 18.000 imágenes. Este porcentaje fue usado con el objetivo de buscar la optimización de los recursos, pues t-SNE es una técnica de uso intensivo, que crece aún más al contar con una matriz de características tan grande. Más adelante, la Figura 16 representa el espacio de baja dimensión generado.

UrbanSound8K

Este conjunto de datos corresponde al conjunto de datos reales. De este se seleccionaron los audios cuya duración fuera de 4 segundos, obteniendo un subconjunto de 7.077 audios, que representa el 81% de los audios totales. Las clases de este subconjunto se distribuyen como se muestra a continuación, en la Figura 14

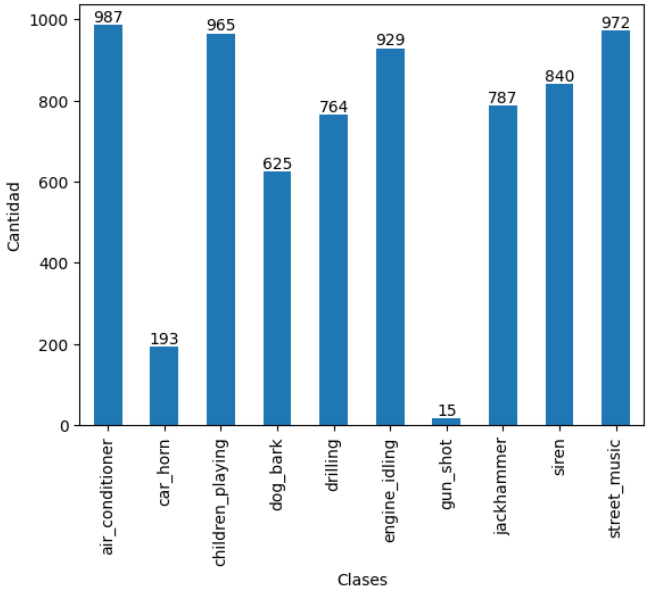


Figura 14. Audios de 4 segundos por clase en el dataset UrbandSound8K

Esta selección es conveniente para la extracción de características con VGGish, pues al crearse las pequeñas secciones de audios de la forma $H = \frac{t}{0.96}$, el resultado para audios de 4 segundos será de aproximadamente 4.16 segmentos, cubriendo de esta forma la mayor parte del tiempo de los audios.

El modelo pre-entrenado VGGish se obtuvo desde GitHub[2], y se importó mediante `PyTorch`. Para la extracción de características se le provee al modelo pre-entrenado, cada uno de los 7.077 audios seleccionados, obteniendo una matriz de cada audio con la forma [1, 4, 128], es decir, por cada

[2] https://github.com/harritaylor/torchvggish

audio procesado, se extraen 4 segmentos con 128 características. Las matrices de características se apilaron en un tensor, el cual posteriormente fue aplanado y en la última posición, fue agregada la etiqueta de clase correspondiente, el resultado es una matriz con dimensiones [7.077, 513]. Finalmente, esta matriz con todas las características de los audios se almacenó en un archivo h5py[3] para su posterior uso. Esta implementación puede observarse en el siguiente enlace VGGish UrbanSound8K.

Al igual que MNIST, a la matriz de características de UrbanSound8K antes creada, se le generó la visualización en baja dimensión usando t-SNE para de esta forma, contar con un punto de partida de comparación sobre el aprendizaje alcanzado por las redes VAE y CVAE. La Figura 21 representa la visualización obtenida.

IMPLEMENTACIÓN MODELOS AUTOENCODER

Modelo VAE

La implementación del modelo VAE se basó en la publicación (Van de Kleut, 2020), donde se presenta una arquitectura orientada a objetos con una clase `VariationalEncoder` que adiciona las distribuciones de probabilidad, otra clase `Decoder` y una clase agrupadora llamada `VariationalAutoencoder`. Tomando esto como punto de partida, se construyó un modelo igualmente orientado a objetos compuesto por lo siguiente:

- La clase `VariationalEncoder`: encargada de generar el espacio latente a partir de una capa oculta, que luego pasa por una capa de activación ReLU y a partir de esta, las distribuciones de probabilidad, una correspondiente al vector de medias y otra a la desviación estándar. Así mismo, la divergencia KL, es calculada a partir del resultado de la capa de activación.
- La clase `Decoder`: encargada de proyectar de vuelta la información del espacio latente al espacio original, se compone de dos capas ocultas, el resultado de la primera pasa a la capa de activación ReLU y este resultado, es usado por la segunda capa oculta. Posteriormente el resultado de esta pasa por una nueva capa de activación Sigmoide prediciendo la probabilidad como salida y alcanzando la proyección de los datos.
- Clase `VariationalAutoencoder`: encargada de condensar las clases anteriores, además de orientar el flujo de trabajo *Encoder-Decoder* para el aprendizaje de la red. Esta clase también provee la función de entrenamiento, orientando al aprendizaje de la red por épocas y lotes, definiendo el optimizador Adam con su tasa de aprendizaje y calculando la pérdida de reconstrucción a partir de divergencia KL y MSE, que será retropropagada. Al final del entrenamiento, la función provee el modelo con menor pérdida como también el histórico de pérdidas por época.

[3] Interfaz en python para el formato de datos binarios HDF5, que permite almacenar grandes cantidades de datos numéricos y manipularlos fácilmente con NumPy (*HDF5 for Python — h5py 3.9.0 documentation*, s/f).

En el siguiente enlace podrá accederse al modelo implementado: Modelo VAE.

Modelo CVAE

Para el modelo CVAE se tomó como base la publicación (Raschka, s/f), que cuenta con una colección de varias arquitecturas y modelos de aprendizaje profundo, entre ellos el de CVAE. La implementación consta de una clase llamada `ConditionalVariationalAutoencoder`, la cual condensa como funciones el `encoder` y `decoder`. Adicional a esta clase, también implementa una función para convertir las etiquetas de clase como características onehot, llamada `to_onehot`. Partiendo de esto, se llevó esta implementación a un diseño orientado a objetos, similar a lo realizado con VAE y de esta forma crear similitud entre en los modelos y mantenibilidad. A continuación, se mencionan las diferencias respecto a VAE:

- Se genera una clase llamada `ConditionalVariationalEncoder`, que cuenta con las características ya definidas para `VariationalAutoencoder` en VAE, agregando a esta la concatenación de las características con las etiquetas aplicando onehot. Para CVAE se usó la capa de activación Leaky ReLU.
- La clase `ConditionalDecoder`: se define con las mimas características del `Decoder` en VAE, complementándose con la concatenación de las etiquetas onehot y cambiando igualmente la función de activación por Leaky ReLU.
- La clase `ConditionalVariationalAutoencoder`: al igual que las anteriores, implementa las características ya definidas para la clase `VariationalAutoencoder` en VAE. Con la diferencia en el cálculo de la pérdida de reconstrucción, pues esta lo efectúa a partir de la divergencia KL y la entropía cruzada binaria, además de que en esta misma clase se implementa la función que convierte las etiquetas de clases a características onehot (`to_onehot`).

La implementación del modelo se encuentra en el siguiente enlace: Modelo CVAE.

OPTIMIZACIÓN MODELOS

En la optimización de hiperparámetros para las redes VAE y CVAE, se usó Optuna. Con esta API se inició la búsqueda exploratoria de los siguientes hiperparámetros, de manera independiente para cada red:

- Dimensiones Latentes o tamaño espacio latente.
- Cantidad de neuronas en la capa oculta.
- Tasa de aprendizaje.
- Número de épocas.

Esta búsqueda exploratoria consistió en la asignación de rangos amplios para cada uno de los hiperparámetros de cada una de las redes, los cuales Optuna a través de la siguiente configuración, se encargaba de buscar el mejor arreglo de hiperparámetros que al mismo tiempo minimicen la

función de pérdida de reconstrucción o función objetivo. A continuación, se detalla la configuración del estudio en Optuna:

- Sampler u objeto muestreador, se usó el que se encuentra por defecto, es decir, TPESampler Tree-structured Parzen Estimator algorithm el cual se basa en el muestreo independiente. En cada ensayo, para cada parámetro, TPE ajusta un Modelo de Mezcla Gaussiana (MMG) $l(x)$ al conjunto de valores de los parámetros asociados a los mejores valores objetivo, y otro MMG $g(x)$ a los restantes valores de los parámetros. Elige el valor de parámetro x que maximiza la relación $l(x)/g(x)$ (*Optuna 3.3.0 Documentation*, s/f).
- Objeto podador, se encarga de decidir la detención anticipada de los ensayos poco prometedores. Para este, se usó MedianPruner, el cual se centra en los resultados anteriores y elimina las combinaciones que se encuentren por debajo de la mediana de los resultados obtenidos hasta ese momento, para el mismo paso. Sobre este, se usó la siguiente configuración:
    - Número de pasos de calentamiento o n_warmup_steps: se configuró con valor igual a 5, es decir, antes de comenzar a aplicar al análisis de poda, se completarán 6 pruebas iniciales (considerando que inicia desde 0), de esta forma se espera crear una base sobre la cual se compararán las siguientes pruebas.
    - Pasos intervalo o interval_steps: configurado con valor igual a 3, representa los intervalos que pasan entre las comprobaciones y aplicaciones de la poda.

  Con la configuración del objeto podador, se busca optimizar los recursos, reduciendo tanto el tiempo como la cantidad de cómputo a realizar para la optimización de cada red.
- Dirección de optimización, parámetro que define la dirección de optimización de la función objetivo, para nuestro caso, se estableció minimizar, cuya función objetivo es la función de pérdida de reconstrucción entregada por los modelos VAE y CVAE.

Una vez configurado el estudio, se procede a definir los parámetros de optimización, para ello se define:

- La función de optimización corresponde a la función de pérdida, esta se calcula al entrenar el modelo con los parámetros recomendados para el ensayo por el algoritmo de Optuna y el conjunto de datos UrbanSound8K. Este proceso se encuentra envuelto en una función llamada `objective` encargada de recibir por parámetro los valores recomendados ya mencionados, de construir el modelo de la red y disparar el entrenamiento; obteniendo de regreso el menor valor de la función de pérdida de reconstrucción, el cual es reportado a Optuna.
- El número de ensayos para cada proceso se definió igual a 100, lo que quiere decir que Optuna probará máximo 100 recomendaciones de hiperparámetros.
- El número de trabajos paralelos igual a 4, es decir, al mismo tiempo se realizará el estudio de 4 configuraciones de hiperparámetros recomendados por el algoritmo de Optuna, este valor varía en función de la capacidad de cómputo, en nuestro caso se usó la GPU disponible

en el entorno de Kaggle (GPU Tesla P100-PCIE-16GB y 2 CPU Intel® Xeon® @ 2.00 GHz 1 Core, cada Core con 2 hilos).

Contando ya con los resultados de la búsqueda inicial exploratoria con Optuna, ejecutada de manera independiente a cada una de las redes VAE y CVAE; se usaron las visuales gráficas que provee el mismo Optuna, con los resultados del estudio e hiperparámetros evaluados, para identificar la tendencia de los valores obtenidos por cada hiperparámetro, y a partir de ello, iniciar una nueva búsqueda de hiperparámetros, explotando estos nuevos rangos y tratando de hallar mayor eficiencia en cada una de las redes a razón de la minimización de la función de pérdida. Las gráficas principalmente usadas en este análisis se describen a continuación:

- plot_slice: traza la relación de parámetros como diagrama de corte en un estudio.
- plot_param_importances: traza las importancias de los hiperparámetros.
- plot_optimization_history: traza el historial de optimización para todos los ensayos de un estudio.
- plot_intermediate_values: representa gráficamente los valores intermedios de todos los ensayos de un estudio.
- plot_timeline: traza la cronología de un estudio.

Nota: para más información respecto a las visuales gráficas ofrecidas por Optuna referirse al enlace que se encuentra como nota al pie[4].

A partir de la gráfica "plot_slice", se lograba identificar patrones de agrupación hacia uno de los costados de la gráfica por cada hiperparámetro. Consiguiendo así un nuevo rango de explotación, por tanto, se configura un nuevo estudio en Optuna con un rango más cerrado. Esta búsqueda de hiperparámetros explotacional se realizó 2 veces para cada una de las redes VAE y CVAE. Las demás gráficas visuales, se usaron principalmente para entender el rendimiento de la búsqueda de hiperparámetros, identificando factores como la efectividad y cantidad de pruebas podadas e incluso la evolución de la búsqueda de la minimización de la función de pérdida.

El tipo de sugerencia solicitada a Optuna alrededor de los hiperparámetros en cada uno de los experimentos se detalla a continuación, en la Tabla 1.

Tabla 1. Configuración de sugerencias en Optuna para la búsqueda de los mejores hiperparámetros

	Tamaño espacio latente	***Neuronas capa oculta***	***Tasa de aprendizaje***	***Número épocas***
Optuna	Suggest_int	Suggest_int	Suggest_float y flag log=True	Suggest_int
Descripción	Sugiere números en el rango de los enteros	Sugiere números en el rango de los enteros	Sugiere en el rango del dominio logarítmico	Sugiere números en el rango de los enteros

[4] https://optuna.readthedocs.io/en/stable/reference/visualization/index.html

De esta misma forma, en la Tabla 2 y Tabla 3, se muestran los rangos sobre los que Optuna realizó la búsqueda para cada hiperparámetro y experimento.

Tabla 2. Configuración de rangos de búsqueda en Optuna para la búsqueda de los mejores hiperparámetros por experimento con VAE

Experimento	***Tamaño espacio latente***	***Neuronas capa oculta***	***Tasa de aprendizaje***	***Número épocas***
Exploración	3 - 203	150 – 400	1e-5 - 1e-2	20 - 100
Explotación 1	10 – 100	300 – 450	1e-4 - 8.3e-4	49 - 150
Explotación 2	15 - 60	390 – 490	2e-4 - 4e-4	90 - 200

Tabla 3. Configuración de rangos de búsqueda en Optuna para la búsqueda de los mejores hiperparámetros por experimento con CVAE

Experimento	***Tamaño espacio latente***	***Neuronas capa oculta***	***Tasa de aprendizaje***	***Número épocas***
Exploración	3 - 203	150 - 400	1e-5 - 1e-2	20 - 100
Explotación 1	6 – 50	300 - 450	3e-4 – 9.3e-4	40 - 150
Explotación 2	7 – 30	400 – 490	5e-4 – 7e-4	100 - 200

Los diagramas de corte o plot_slice con los resultados de los experimentos anteriores, se pueden visualizar en la Figura 22, Figura 23 y Figura 24 para VAE y en la Figura 26, Figura 27 y Figura 28 para CVAE. Los resultados obtenidos por cada experimento junto con los hiperparámetros hallados por Optuna se pueden observar en la Tabla 4 y Tabla 5, para VAE y CVAE, respectivamente.

Las implementaciones realizadas con Optuna para la búsqueda de hiperparámetros antes de descritas se encuentran en los siguientes enlaces:

- Optimización Hiperparámetros VAE
- Optimización Hiperparámetros CVAE

GENERACIÓN Y VALIDACIÓN

Entrenamiento

Para la generación de información acústica sintética, primero se realizó el entrenamiento de las redes VAE y CVAE, tomando uso de los mejores hiperparámetros hallados con la ayuda de Optuna (ver Tabla 4 y Tabla 5) más la matriz de características VGGish de UrbanSound8K, cuyos datos se encuentran en el archivo con formato hdf5. Luego de extraer los datos de este archivo, se separan las etiquetas de clase de las características y a la matriz resultante de características se le realiza normalización de datos al rango de valores 0-1. La matriz de características normalizada y las etiquetas de clase son entregadas a la función de entrenamiento configurada en cada red, obteniendo de regreso el modelo entrenado; a este modelo se le extraen los pesos y sesgos de sus

capas y se almacenan, con la ayuda de PyTorch, en un archivo de datos para posteriormente usarlo en las tareas de generación de datos. La implementación descrita puede observarse en el enlace TSNE UrbanSound8K, donde también se encuentran algunas visualizaciones en baja dimensión de las características VGGish del conjunto de datos UrbanSound8K.

Generación

Para la generación de datos se usaron las dos técnicas mencionadas, una por cada modelo de autoencoder implementado; para VAE se usó la técnica denominada interpolación y para CVAE por etiquetas de clases, logrando con cada una de ellas la generación de datos sintéticos dentro del dominio de la información.

El primer paso, fue usar los modelos ya entrenados de VAE y CVAE, los cuales se cargaron con la ayuda de PyTorch, a partir de cada archivo guardado con sus pesos y sesgos, regenerando y dejando listo cada modelo para su uso. También se realizó la carga del archivo con las características VGGish de UrbanSound8K, donde sus datos nuevamente fueron normalizados a la escala 0-1. Tanto los modelos y la matriz de características mencionada se usaron para la generación de datos, aplicando las técnicas correspondientes a cada modelo, así:

- Para VAE, se generaron datos mediante interpolación a partir de la selección, dentro la matriz de características VGGish, de aquellas características pertenecientes a cada una de las etiquetas de clase. Sobre esta sub-matriz de características por etiqueta, se toman de manera aleatoria los valores x_1 x_2 y a partir de estos, se generan los respectivos vectores en espacio latente con el encoder z_1 z_2. Dentro de este rango de espacios latentes se generaron 12 puntos usando expresión de la Ecuación (6), para los cuales antes de generar su reconstrucción con el decoder, se remueve la primera y última posición, que corresponden a z_1 z_2, dejando como resultado 10 puntos intermedios correspondientes a los datos sintéticos generados. Este mismo proceso, se repite por cada una de las etiquetas de clase 8 veces, generando un total de 800 nuevos datos sintéticos.
- Con CVAE y la técnica por etiquetas de clase, se generaron datos a partir de la construcción de un tensor de tamaño 10, cuyos valores representan la etiqueta de clase de los datos que se desean generar, luego se crea un segundo tensor a partir de una distribución aleatoria normal estándar de tamaño 10 x 8, este último es el tamaño del espacio latente obtenido de la optimización con Optuna (ver Tabla 5); este tensor representa el espacio latente aleatorio para las características que se desean generan basadas en la etiqueta del primer tensor. Este par de tensores es entregado al decoder del modelo CVAE pre-entrenado, creando la reconstrucción de las características de la etiqueta entregada, es decir, se generan 10 nuevos datos sintéticos. Este mismo proceso se repite para cada una de las etiquetas de clase y al igual que VAE, por cada una de las etiquetas se repite 8 veces, lo que genera 800 nuevos datos sintéticos.

Las implementaciones descritas en los párrafos anteriores se pueden observar en el enlace Data Generation UrbanSound8K.

Validación

La validación de los datos generados para ambas redes VAE y CVAE se basó en la visualización a baja dimensión de las características VGGish del conjunto original, que las denominaremos base, junto con las características generadas por cada una de las redes previamente entrenadas. Estas matrices fueron apiladas de manera tal que se obtiene una matriz del tamaño [7877, 512] representando 7077 datos base más 800 generados por cada modelo. Las etiquetas de clase para las características base, se conservan tal como su origen, sin embargo, para los datos generados a su etiqueta de clase se le suma la base 10, creando etiquetas en el rango 10-19, todo ello para diferenciar con facilidad los datos base de los datos generados. Con la técnica t-SNE se genera la visualización de baja dimensión 2D de este conjunto de datos compuesto. A la gráfica generada se le adicionan características visuales que ayudan a facilitar la detección y comparación, estas mejoras se describen a continuación:

- Para los nuevos datos generados, se ajusta el marcador estrella (☆), mientras que los datos base conservan el marcado por defecto
- Se ajusta el mapa de color a *nipy_spectral*, que ofrece una amplia variedad de colores, bastante discriminantes entre sí para identificar con mayor claridad las regiones. Además, se modifica la transparencia a 0.5 sobre el color de fondo de las características base, aumentando el contraste entre los datos base y los generados
- Sobre las cajas de texto dentro del gráfico, que representan el valor de las etiquetas de clase, se conserva la distribución 0-9 para los datos base y 10-19 para los datos nuevos generados. Su ubicación es determinada por el centroide de la región de datos que representa
- También se genera una leyenda, donde se presentan cada una de las clases, tanto las base como las generadas, en un rango de 0-19 junto con su marcador y color asignado.

Esta alternativa de validación entre las características base y las generadas mediante su visualización a baja dimensión con t-SNE, ofrece una manera ágil de observar el aprendizaje alcanzado por las redes como también la efectividad de la generación de datos, al observar cómo los datos generados conservan los patrones de agrupaciones por regiones de acuerdo con su clase base. Los resultados de estas representaciones en baja dimensión se pueden observar en la Figura 31 para VAE y Figura 32 para CVAE.

4. RESULTADOS Y DISCUSIÓN

Los resultados presentados a continuación se dividen en tres partes como se muestra en la Figura 15. La primera corresponde a los experimentos realizados con MNIST, con el fin de implementar y validar visualmente los resultados iniciales de los modelos. En la segunda parte, se presentan los resultados de optimización de estos. Finalmente, en la tercera parte se presentan los resultados de aprendizaje, generación y validación sobre el conjunto de datos UrbanSound8K.

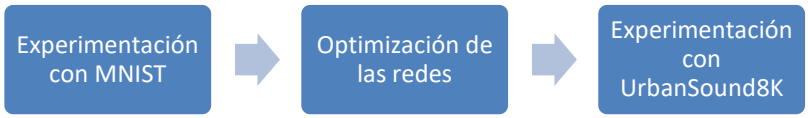


Figura 15. Esquema de resultados

EXPERIMENTACIÓN CON MNIST

El conjunto de datos MNIST y los experimentos realizados con este, se realizaron para obtener una validación visual rápida de los resultados alrededor del aprendizaje de las redes y su comportamiento en la generación de dígitos nuevos, esto último mediante las técnicas de interpolación o por etiquetas de clases según cada red autoencoder implementada.

Se puede observar en la Figura 16, cómo para el conjunto de datos MNIST se define cada una de las regiones correspondientes a los dígitos en la representación de baja dimensión alcanzada con t-SNE. Este punto de partida permitió corroborar el aprendizaje de las redes cuyo resultado, en este mismo conjunto de datos, deberá ser muy similar o incluso superior. Para esta visualización se realizó una reducción desde 784 características (dimensiones) a 2 dimensiones, donde las cajas de texto con los dígitos MNIST, representan los centroides de las regiones de cada clase.

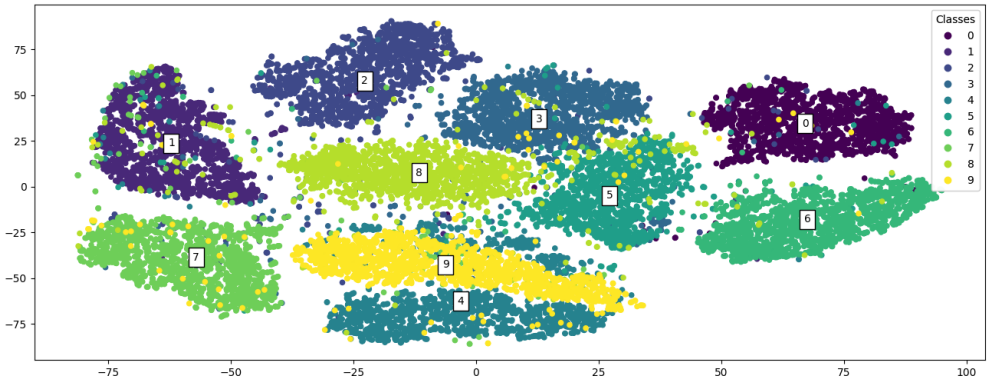


Figura 16. t-SNE Base MNIST

Luego de realizar el entrenamiento de la red VAE con el conjunto de datos MNIST y los hiperparámetros recomendados en la publicación de (Van de Kleut, 2020), se observar en la Figura 17 la representación del espacio latente y cómo en este se notan mejoras en la definición de las regiones, demostrando el aprendizaje y la capacidad de las redes VAE en la identificación de los patrones en comparación a la Figura 16.

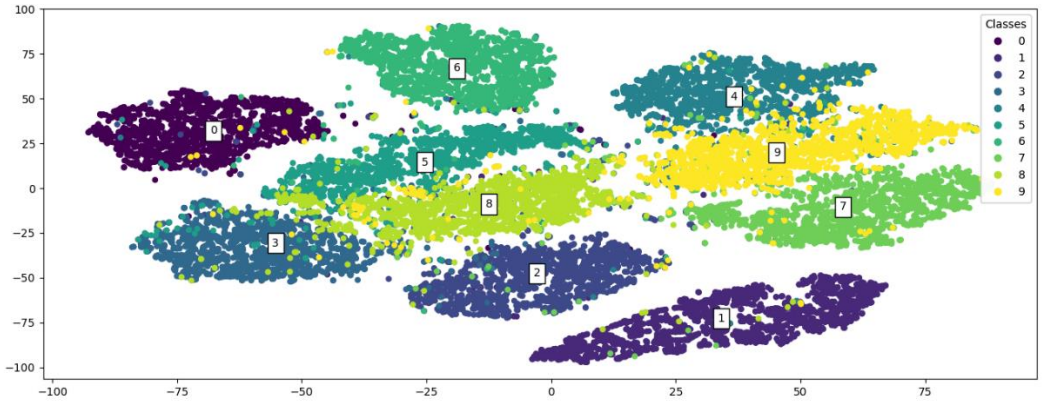


Figura 17. t-SNE VAE MNIST

Partiendo del resultado anterior, se procede a generar dígitos nuevos mediante la técnica de interpolación, los cuales se encuentran representados en la Figura 18. Es importante destacar que los extremos corresponden a los valores x_1 y x_2, proveídos para interpolar 8 datos intermedios completamente nuevos. La Figura 18 se compone de 4 muestras de generación de datos enumeradas del 1 al 4, donde la primera busca generar números 1, la segunda números 3; en cambio para la tercera y cuarta se buscó generar dígitos en la transición de los números 3 a 8 y de 1 a 7, respectivamente. Procesos que VAE completó con bastante efectividad y en los cuales, los valores generados tienen sentido visualmente.

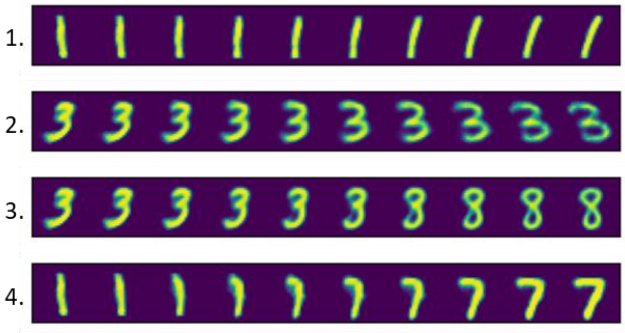


Figura 18. Generación dígitos MNIST con VAE por interpolación

Con CVAE se realizó un proceso muy similar tomando el conjunto de datos MNIST, pero en este caso usando los hiperparámetros expuestos en la publicación de (Raschka, s/f). De esta forma, se obtuvieron los resultados que se encuentran en la Figura 19, donde se observa en la representación t-SNE el aprendizaje alcanzado por la red y cómo los patrones de los dígitos MNIST se agrupan en

una super región, prácticamente sin distinción y cuyos centroides se encuentran muy cercanos los unos a los otros.

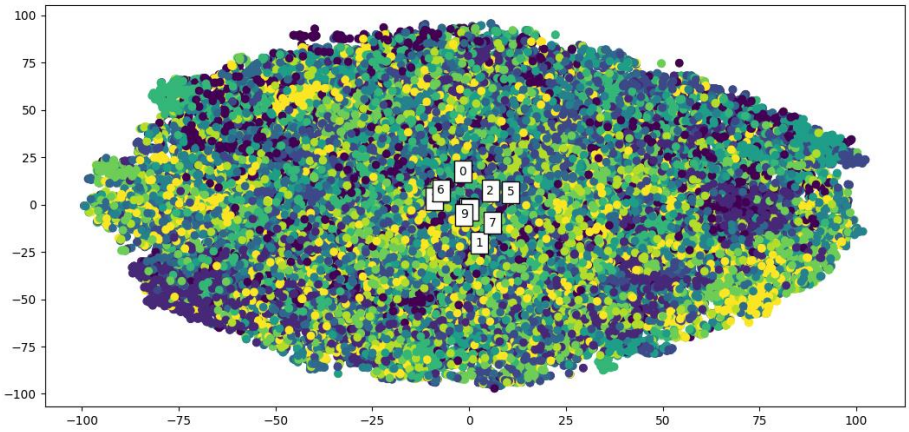


Figura 19. t-SNE CVAE MNIST

A pesar de los resultados de la visualización anterior, CVAE conserva bastante bien la posibilidad de generar dígitos nuevos, como se presenta en la Figura 20. En esta se generaron 4 muestras basadas en las etiquetas de clase 0, 1, 3 y 8, produciendo 10 nuevos dígitos para cada uno de ellos. Podemos observar cómo los dígitos generados son correspondientes, mostrando cómo esta red también se destaca en la generación sintética de información y dejando un desafío alrededor de la visualización de su aprendizaje, pues se podría considerar que la proyección a baja dimensionalidad con la técnica t-SNE puede verse limitada en estas implementaciones.

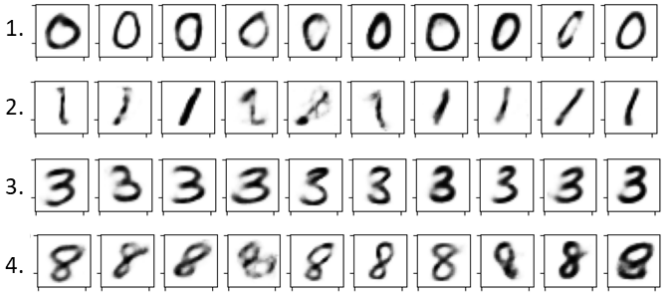


Figura 20. Generación de dígitos MNIST CVAE por Etiquetas de Clase

OPTIMIZACIÓN DE LAS REDES

Como punto de comparación en la Figura 21, se representan en baja dimensión las características base extraídas con VGGish para UrbanSound8K, es decir, se muestra una reducción desde 512 características a 2 dimensiones. A partir de esta representación, se busca generar dos nuevas visualizaciones que muestren el aprendizaje alcanzado por las redes VAE y CVAE, respectivamente. Las cajas de texto dentro de la figura representan las etiquetas de clase de los audios y están ubicadas en los centroides de cada una de las regiones.

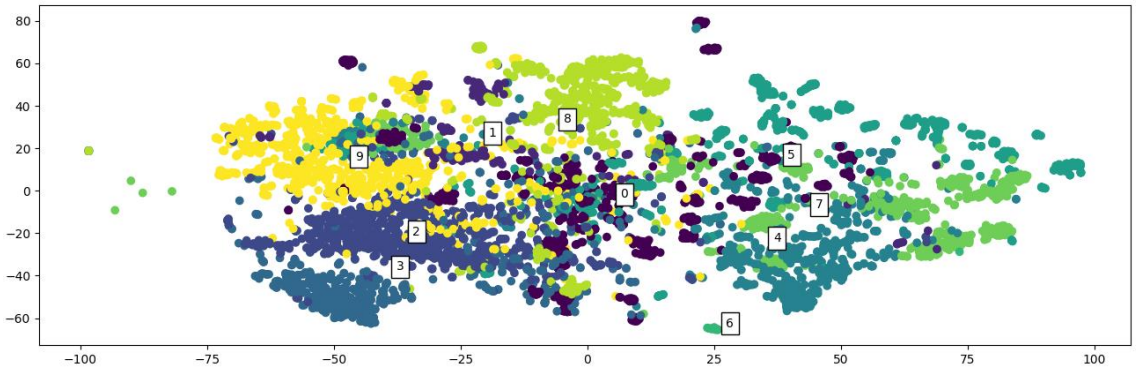


Figura 21. t-SNE Base UrbanSound8K VGGish

Optimización VAE

Los resultados del primer experimento de exploración de hiperparámetros con VAE se presentan en las Figura 22, en ella se puede notar la tendencia y agrupaciones de los valores adquiridos por cada uno de los hiperparámetros, estas dos características nos permiten configurar el siguiente experimento de explotación con el objetivo de encontrar una mejora en los hiperparámetros, como es el caso de las dimensiones latentes, pues se observa que los valores mínimos objetivo usan valores inferiores a 100, diferente al caso del número de épocas y la cantidad de neuronas en la capa oculta, cuyos valores objetivo se minimiza al aumentar sus valores. En este caso también se obtuvo un 75 % de poda sobre el total de pruebas, es decir, que en la Figura 22 se visualiza el resultado de 25 pruebas completadas sobre 100.

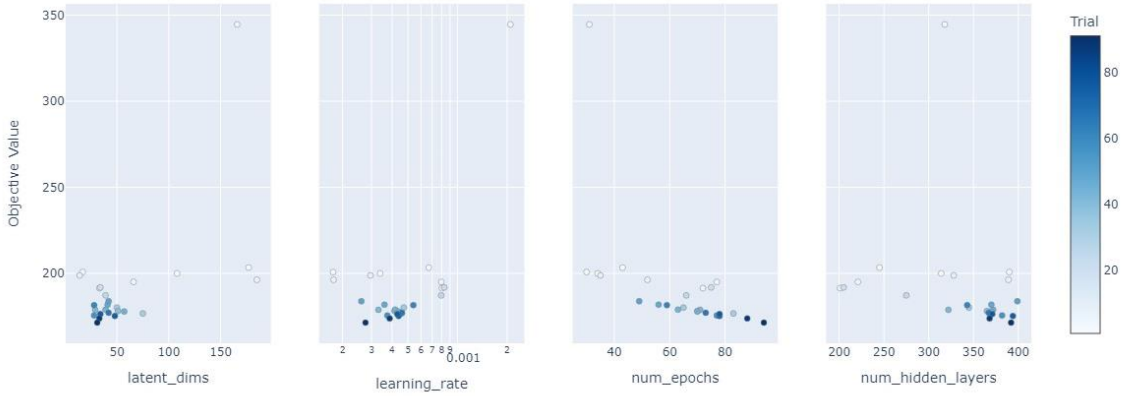


Figura 22. Diagrama de corte - Exploración hiperparámetros VAE

En una segunda iteración de la búsqueda de hiperparámetros para VAE, se configura el experimento de explotación #1, cuyos resultados se presentan en la Figura 23, con un resultado del 80 % de poda. En este segundo experimento se identifican agrupaciones de valores muy cercanas a los extremos, similares al experimento de exploración anterior, por lo que se identifica una nueva configuración de rangos para los hiperparámetros, buscando hallar una mayor minimización de la función objetivo.

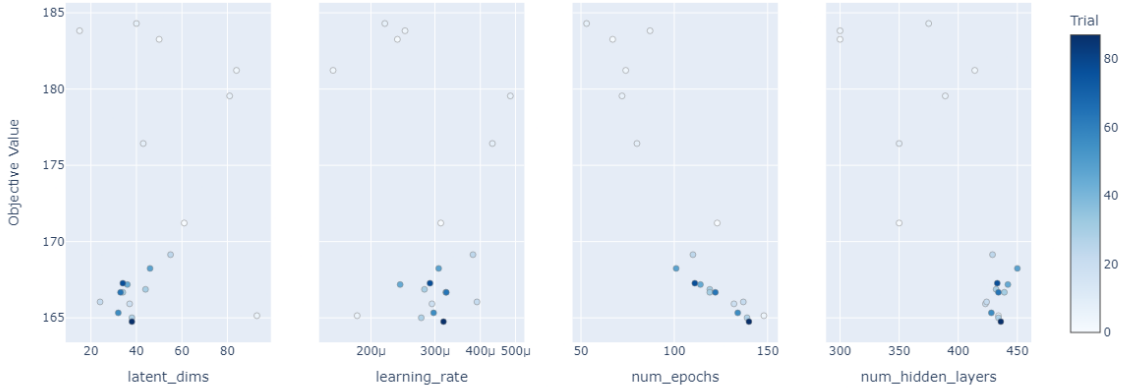


Figura 23. Diagrama de corte - Explotación 1 hiperparámetros VAE

Finalmente, en el experimento de explotación #2 obtenemos un resultado del 88 % de poda, demostrando un nivel alto de optimización y por ende los resultados de los hiperparámetros de este experimento se consideran óptimos para el alcance de este trabajo. Además, se puede observar en la Figura 24, cómo los valores de los hiperparámetros se ubican más hacia el centro de cada gráfico; exceptuando los valores del número de épocas y la cantidad de neuronas en la capa oculta, hiperparámetros que observamos con un comportamiento de minimización de la función objetivo en la medida que ellos también aumentan. A pesar de ello, se consideró no elevar aún más el valor de estos dos hiperparámetros ya que de un lado, con el aumento del número de épocas, implica mayor consumo de tiempo y recursos y, por otro lado, el aumento de neuronas de la capa oculta representaría un valor muy cercano al tamaño original de la matriz de características, perdiendo de esta forma el objetivo de codificación de esta capa.

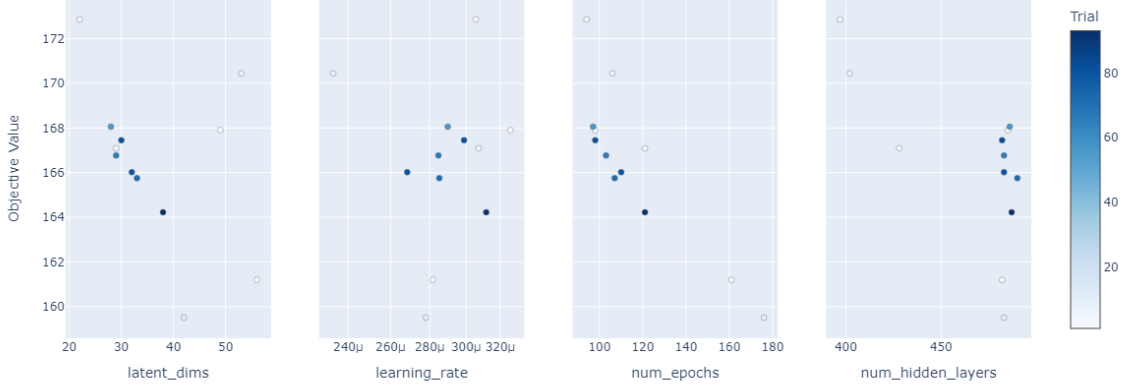


Figura 24. Diagrama de corte - Explotación 2 hiperparámetros VAE

A continuación, en la Tabla 4, se presentan los resultados por experimento de las optimizaciones ejecutadas con Optuna para la búsqueda de los mejores hiperparámetros de la red autoencoder VAE, representando una mejora respecto a la pérdida de reconstrucción del 6,96 % entre el

experimento inicial y el final, este último define los mejores hiperparámetros sobre el conjunto de datos UrbanSound8K para la red autoencoder VAE.

Tabla 4. Resultados de la optimización para VAE

Experimento	*Tamaño espacio latente*	*Neuronas capa oculta*	*Tasa de aprendizaje*	*Número épocas*	*Pérdida Reconstrucción*	*% Poda*
Exploración	31	392	~ 0,000276	94	~ 171,434280	75 %
Explotación 1	38	436	~ 0.000317	140	~ 164.752526	80 %
Explotación 2	42	483	~ 0.000278	176	~ 159.506409	88 %

Tomando estos mejores hiperparámetros, se genera la representación t-SNE de la red VAE (ver Figura 25) sobre el conjunto de datos UrbanSound8K, generando una reducción desde 42 dimensiones a 2 y visualizándose mejoras sutiles respecto a la agrupación de regiones en contraste con la visualización base de la Figura 21 donde las regiones se encontraban algo más dispersas.

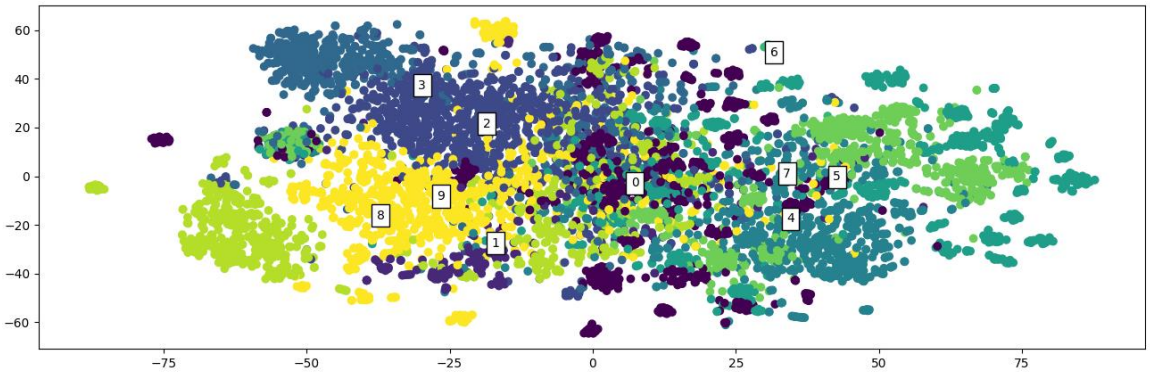


Figura 25. t-SNE VAE Optimizado UrbanSound8K

Optimización CVAE

Para la red autoencoder CVAE, se realizaron experimentos muy similares respecto a lo antes descrito para VAE. En la Figura 26 se observa la tendencia de los hiperparámetros en el primer experimento exploratorio, donde las dimensiones latentes y el número de neuronas en la capa oculta tienden a los extremos, la primera hacia su extremo izquierdo lo que propone reducir para el siguiente experimento el espacio de búsqueda a menos de 100; para el caso de la segunda, su tendencia va en dirección derecha proponiendo un espacio de búsqueda mayor a 300. Sobre los hiperparámetros restantes se identifica posibles reducciones de los rangos hacia los espacios más centrales como es el caso de la tasa de aprendizaje. Sin embargo, para el caso del número de épocas se considera la experiencia obtenida con la optimización de VAE, por lo que solo se movió el rango inferior limitando la búsqueda del siguiente experimento a valores superiores a 50 y extendiendo a su vez el límite superior. Este experimento representó un 68 % de poda sobre 100 pruebas configuradas a ejecutar por Optuna.

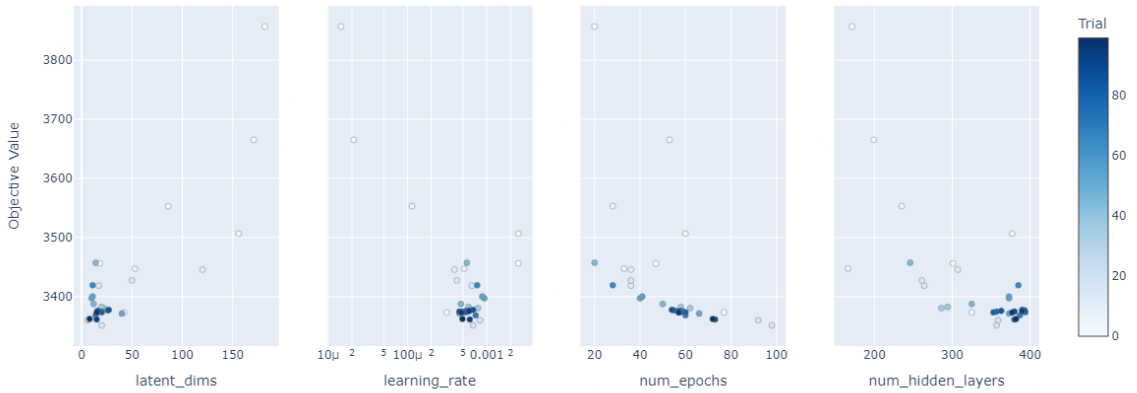


Figura 26. Diagrama de corte - Exploración hiperparámetros CVAE

En la explotación #1 con un 75 % de poda y que se observa sobre la Figura 27, el hiperparámetro de las dimensiones latentes y número de neuronas en la capa oculta, conservan su tendencia hacia los mismos extremos antes vistos, por lo que se extiende el espacio de búsqueda a estos nuevos rangos en el siguiente experimento, mientas que el número de épocas muestra un efecto positivo en la reducción del valor objetivo al aumentar su valor, comportamiento similar a VAE, por lo tanto, en el siguiente experimento se amplía su rango superior, en pro de hallar valores que minimicen aún más la función objetivo, así mismo se aplicó para el número de neuronas en la capa oculta.

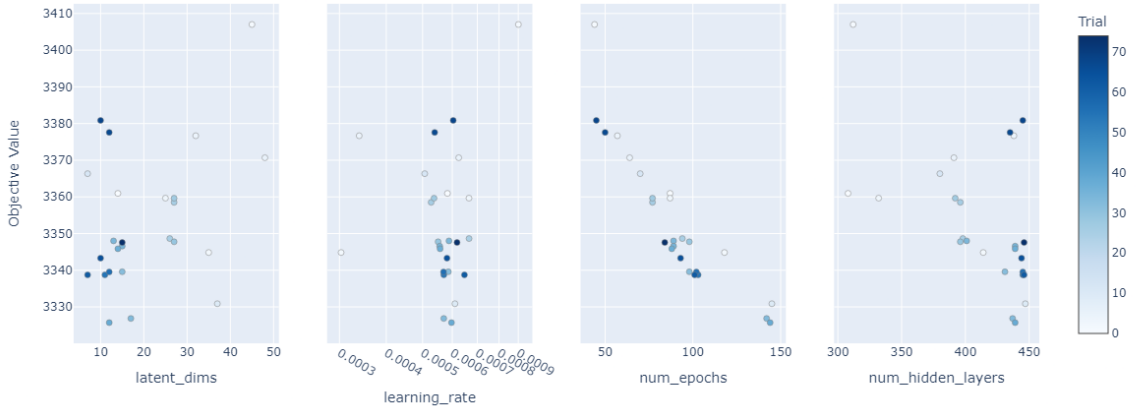


Figura 27. Diagrama de corte - Explotación 1 hiperparámetros CVAE

Sobre el tercer y último experimento de explotación #2 para CVAE cuyo resultado de poda fue del 89 %, nuevamente se halla un nivel alto de optimización, de modo que se fijan estos hiperparámetros como los óptimos para CVAE con el conjunto de datos UrbanSound8K. Este último experimento también comparte similitudes con los resultados obtenidos con VAE, donde nuevamente el número de épocas y la cantidad de neuronas en la capa oculta apuntan a que a mayor valor mayor se alcanza la minimización de la función objetivo, para estos, se toman los mismos criterios que en VAE de limitar el aumento de sus valores en función de la optimización de recursos y evitar la pérdida de sentido sobre capa oculta por esta acción.

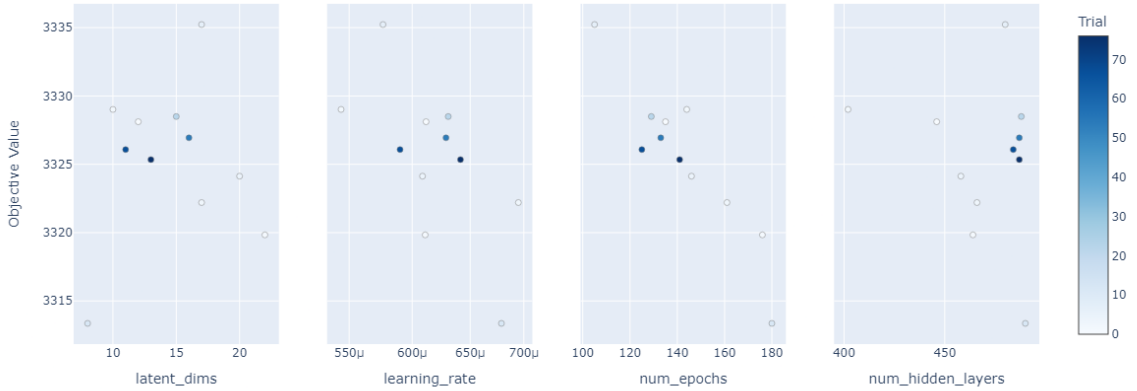


Figura 28. Diagrama de corte - Explotación 2 hiperparámetros CVAE

La Tabla 5 contiene los resultados por experimento de las optimizaciones para la red CVAE sobre el conjunto de datos UrbanSound8K, alcanzando una reducción sobre la función de pérdida inicial del 1.13 % y mostrando un menor resultado en comparación a los alcanzados previamente con VAE.

Tabla 5. Resultados de la optimización para CVAE

Experimento	***Tamaño espacio latente***	***Neuronas capa oculta***	***Tasa de aprendizaje***	***Número épocas***	***Pérdida Reconstrucción***	***% Podas***
Exploración	20	357	~ 0.000670	98	~ 3351.246395	68 %
Explotación 1	12	439	~ 0.000597	144	~ 3325.720622	75 %
Explotación 2	8	490	~ 0.000679	180	~ 3313.357958	89 %

Tomando los mejores hiperparámetros obtenidos del experimento de exploración #2, se genera la Figura 29 como la representación en baja dimensión del aprendizaje alcanzado por CVAE sobre el conjunto de datos UrbanSound8K, y sobre esta visualización podemos identificar cómo conserva el comportamiento ya detectado con el conjunto de pruebas MNIST, donde las regiones tienden formar parte de una super región con los centroides muy cercanos entre sí y una aparente baja clasificación de los datos o poca identificación de cada una de las regiones.

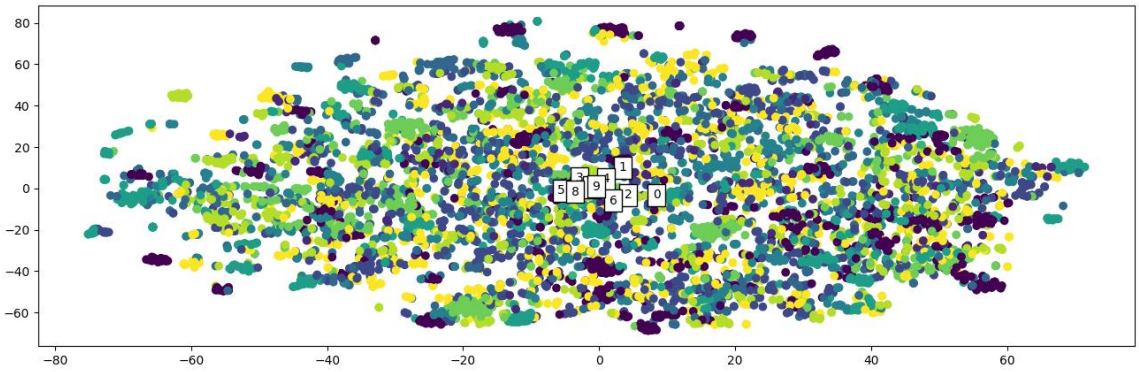


Figura 29. t-SNE CVAE Optimizado UrbanSound8K

EXPERIMENTACIÓN CON URBANSOUND8K

Con fines de comparación de los resultados en la experimentación y generación de datos con UrbanSound8K, la Figura 21 antes presentada se regenera sobre un nuevo mapa de color, ya que este nos permite mayor contraste al momento de comparar los datos base sobre los datos generados de manera sintética. La Figura 30 a continuación representa este cambio.

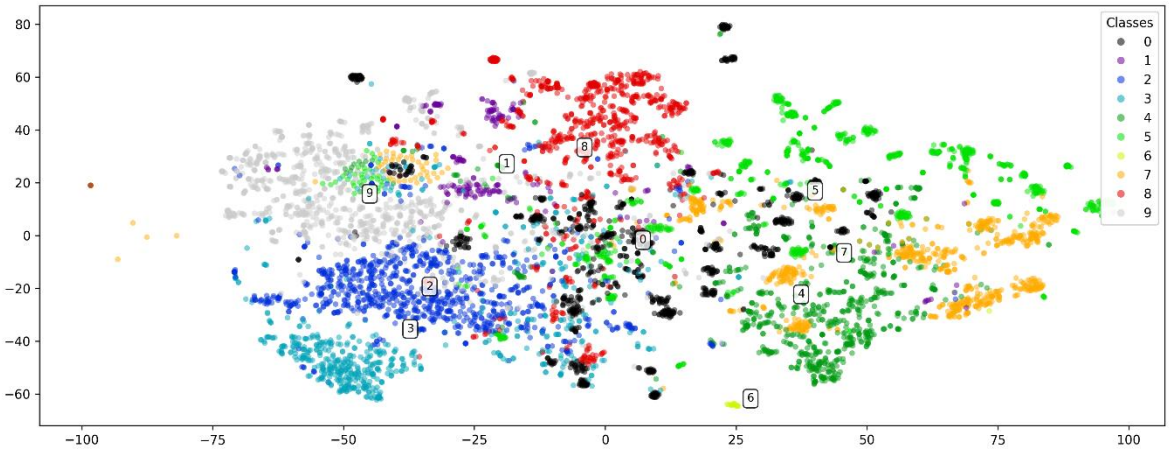


Figura 30. t-SNE Base UrbanSound8K VGGish – Cambio mapa de color

Una vez usado el modelo VAE pre-entrenado con los mejores hiperparámetros para el conjunto de datos UrbanSound8K, se genera la Figura 31, donde se compara la generación de los nuevos datos por interpolación contra los datos base. Se puede apreciar cómo los datos nuevos representados por el marcador (☆) conserva las regiones de las etiquetas de clase base y así mismo los centroides de las etiquetas base y las generadas conservan alta relación y proximidad.

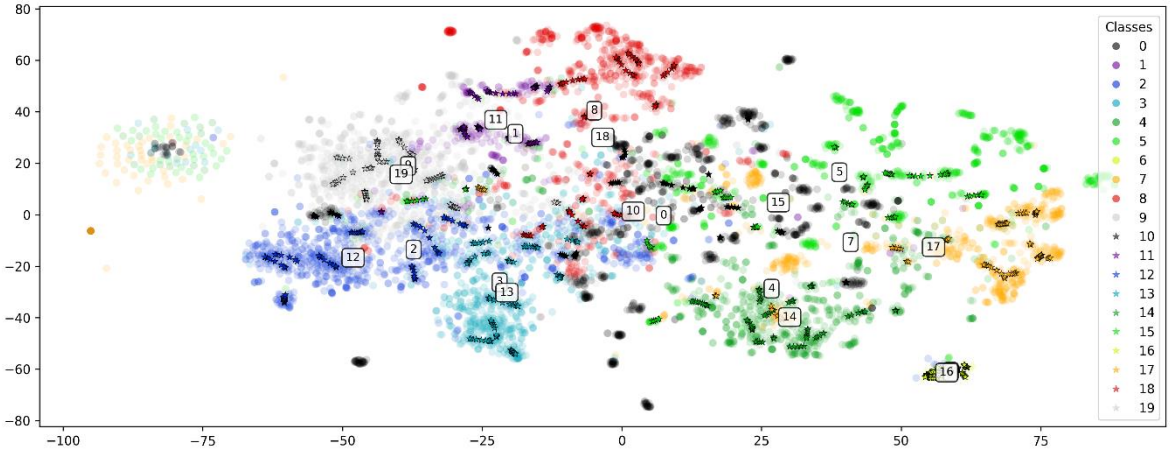


Figura 31 Visualización datos generados VAE

Para el caso de CVAE, podemos apreciar en la Figura 32 cómo los resultados de generación son sobresalientes y correlacionados los datos base, a pesar de la baja diferenciación que previamente se podía notar sobre Figura 29.

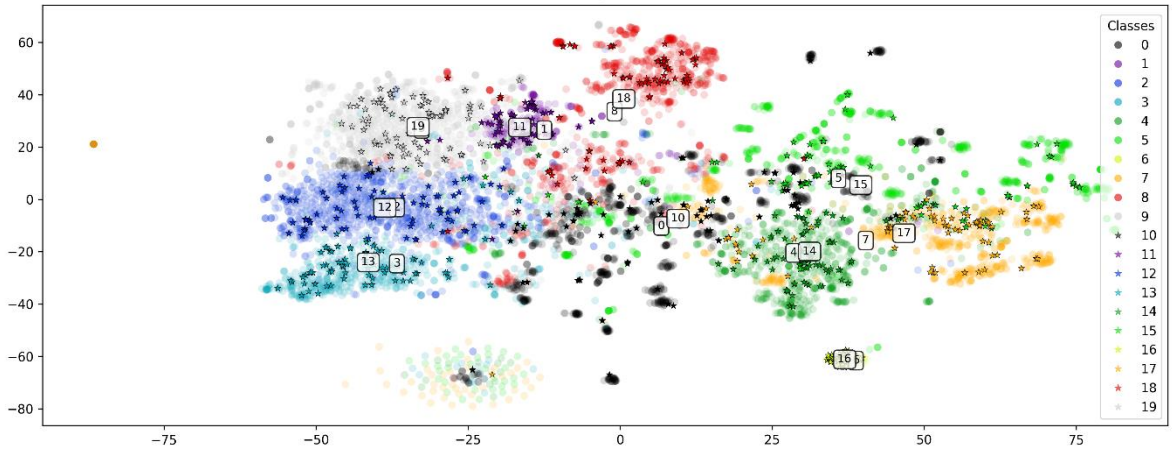


Figura 32 Visualización datos generados CVAE

5. CONCLUSIONES, RECOMENDACIONES Y TRABAJO FUTURO

CONCLUSIONES

Tomando los resultados antes mencionados, se pueden considerar las técnicas basadas en redes neuronales autoencoder VAE y CVAE como modelos con potencial para la generación de información acústica ambiental sintética. Por una parte, VAE mediante su técnica de interpolación presenta una buena alternativa para generar datos de manera controlada a través de rangos que pueden incluso mezclarse. Por otra parte, CVAE mediante la técnica de etiquetas de clase, puede ser una buena alternativa para la generación sintética de datos aprovechándose del control que ofrece parar generar múltiples datos de manera aleatoria sobre una clase determinada.

El uso de modelos pre-entrenados, como VGGish, permite la caracterización de los datos acústicos, que a su vez reduce la cantidad de información para su posterior uso en el entrenamiento de redes neuronales VAE y CVAE, potenciando el trabajo de aprendizaje de patrones e impactando positivamente en la generación de nuevos datos a partir de lo aprendido de este dominio caracterizado.

Las redes neuronales VAE y CVAE, basadas en autoencoders, son entrenadas en un principio con el conjunto de datos MNIST, el cual permite la puesta a punto de las implementaciones, así como una evaluación rápida y visual de los resultados obtenidos. Posteriormente, estas redes se entrenan con las características VGGish del conjunto de datos real UrbanSound8K, compuesto de 10 diferentes clases de sonidos urbanos, permitiendo una codificación de 512 características en 42 para VAE y en 8 para CVAE.

Con el fin de validar la generación de datos se usa la técnica de visualización en baja dimensión t-SNE, obteniendo resultados sobresalientes de comprobación sobre la generación de datos sintéticos de las redes autoencoders. El uso de la técnica t-SNE permite generar evidencias agiles sobre las regiones formadas por los datos generados y visualizar que estos conservan su relación respecto a los datos base. Estas representaciones en baja dimensión nos permiten de forma visual, representar la efectividad de las redes autoencoder en la tarea de generar datos en el dominio de estudio, tanto en el conjunto de datos experimentales como en los reales.

RECOMENDACIONES

La técnica t-SNE ofrece una alternativa visual muy buena para la validación del aprendizaje alcanzado por las redes, como también para la comprobación de la generación de datos. Sin embargo, para la red autoencoder CVAE, se encuentran limitaciones al observar las representaciones de su aprendizaje sobre la proyección de los espacios latentes, pues en los

diferentes experimentos observamos cómo el aprendizaje de esta red se mostraba como una súper región que en apariencia carece de patrones de aprendizaje, pero que luego fueron corroborados durante la generación de nuevos datos. Así pues, identificamos esta técnica como limitante para entender el aprendizaje de la red CVAE y en su caso deberá plantearse una técnica diferente que cubra con mayor eficiencia este tipo de validación.

En este trabajo fue relevante usar, previo a la experimentación con datos reales, un conjunto de datos experimental de manera tal que nos permitiese probar y validar de manera ágil las implementaciones realizadas de las redes autoencoders VAE y CVAE, lo que facilitó la identificación temprana de inconvenientes y su pronta solución. Así mismo, la optimización de hiperparámetros cumple una labor importante a la hora de trabajar con redes neuronales, reflejándose, una vez se identifique estos mejores hiperparámetros, en mejoras en el aprendizaje y optimización de recursos.

Contar con el recurso de GPU mediante las plataformas Google Colab y Kaggle representó gran valor en el desarrollo de este trabajo reflejándose en aspectos como mejoras en la capacidad y velocidad de cómputo, en el rendimiento y tiempos durante la puesta a punto de las redes como a la hora de generar datos sintéticos. A pesar de ello, estos recursos son limitados, principalmente por tiempo de uso, lo que los hace desfavorables en labores prolongadas como la optimización de hiperparámetros o el entrenamiento, por lo que contar con equipos de cómputo que no presenten este tipo de limitaciones, permitirá un desarrollo continuo de las diferentes actividades e incluso repercusiones en la reducción de tiempos.

TRABAJO FUTURO

Técnicas como el monitoreo acústico pasivo, que estudia el paisaje sonoro basado en el monitoreo y análisis de sonido como herramienta para la conservación de la biodiversidad, presentan grandes desafíos a la hora de recopilar dicha información acústica, pues el hecho de requerir equipos lo más poco invasivos, también los limita en temas de almacenamiento y consumo energético. Parte de la solución a esta situación está en la técnica de la grabación asíncrona de por ejemplo los 5 primeros minutos de cada hora o un minuto cada 14 minutos, pero esto deja una brecha de tiempo sin información capturada. Como trabajo futuro, se presenta la oportunidad del uso de las técnicas generativas mencionadas en este trabajo para producir datos acústicos de manera sintética, que se encuentren dentro del dominio de estudio y que permitan ampliar el volumen del conjunto de datos, lo que impactaría positivamente los entrenamientos de redes neuronales, muy usadas en labores de análisis de datos e identificación de patrones, y que dependen de un gran número de datos para generar resultados con mayor precisión.

Los resultados de este trabajo se basaron en evaluaciones sobre aspectos cualitativos a través de la visualización en baja dimensionalidad de los espacios latentes y las características base, a esta última se le adicionaron los datos generados logrando su fácil comparación sobre la conservación de las regiones según la etiqueta de clase. A pesar de obtener resultados tan significativos, es importante considerar la implementación de técnicas que permitan evaluar estos desempeños de manera

cuantitativa. Para trabajo futuro, se observa como alternativa el entrenamiento de modelos de aprendizaje supervisado que permitan evaluar la capacidad de generación de datos con las redes VAE y CVAE, a través de la construcción de conjuntos de datos de entrenamiento y prueba sobre una matriz de características que incluya los datos generados, y la comprobación cuantitativa a través de métricas de desempeño.

REFERENCIAS

Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). *Optuna: A Next-generation Hyperparameter Optimization Framework*. https://arxiv.org/pdf/1907.10902.pdf

Bedi, P., & Gole, P. (2021). Plant disease detection using hybrid model based on convolutional autoencoder and convolutional neural network. *Artificial Intelligence in Agriculture*, *5*, 90–101. https://doi.org/10.1016/J.AIIA.2021.05.002

Chen, L., Li, S., Bai, Q., Yang, J., Jiang, S., & Miao, Y. (2021). Review of Image Classification Algorithms Based on Convolutional Neural Networks. *Remote Sensing 2021, Vol. 13, Page 4712*, *13*(22), 4712. https://doi.org/10.3390/RS13224712

Dahmani, S., Colotte, V., Girard, V., & Ouni, S. (2019). *Conditional Variational Auto-Encoder for Text-Driven Expressive AudioVisual Speech Synthesis*. 2175776. https://inria.hal.science/hal-02175776

Deng, L. (2012, enero). The MNIST Database of Handwritten Digit Images for Machine Learning Research. *IEEE SIGNAL PROCESSING MAGAZINE*. https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/MNIST-SPM2012.pdf

Di, N., Sharif, M. Z., Hu, Z., Xue, R., & Yu, B. (2023). Applicability of VGGish embedding in bee colony monitoring: comparison with MFCC in colony sound classification. *PeerJ*, *11*, e14696. https://doi.org/10.7717/PEERJ.14696/SUPP-1

Doshi, K. (2021, febrero 11). *Audio Deep Learning Made Simple (Part 1): State-of-the-Art Techniques*. https://towardsdatascience.com/audio-deep-learning-made-simple-part-1-state-of-the-art-techniques-da1d3dff2504

Gibb, K. A., Eldridge, A., Sandom, C. J., & Simpson, I. J. A. (2023). Towards interpretable learned representations for Ecoacoustics using variational auto-encoding. *bioRxiv*, 2023.09.07.556690. https://doi.org/10.1101/2023.09.07.556690

HDF5 for Python — h5py 3.9.0 documentation. (s/f). Recuperado el 6 de octubre de 2023, de https://docs.h5py.org/en/stable/index.html

IBM. (s/f). *¿Qué son las redes neuronales convolucionales?* Recuperado el 31 de agosto de 2023, de https://www.ibm.com/mx-es/topics/convolutional-neural-networks

Kingma, D. P., & Lei Ba, J. (2014). *ADAM: A Method For Stochastic Optimization*. https://arxiv.org/pdf/1412.6980.pdf

Moebs, W., Ling, S. J., & Sanny, J. (2021). *Fisica Universitaria Vol 1* (Vol. 1). OpenStax. https://openstax.org/

Naranjo Torres, J., Mora, M., Hernández-García, R., Barrientos, R. J., Fredes, C., & Valenzuela, A. (2020). A Review of Convolutional Neural Network Applied to Fruit Image Processing. *Applied Sciences 2020, Vol. 10, Page 3443*, *10*(10), 3443. https://doi.org/10.3390/APP10103443

Optuna 3.3.0 Documentation. (s/f). Recuperado el 23 de septiembre de 2023, de https://optuna.readthedocs.io/en/stable/reference/samplers/generated/optuna.samplers.TPESampler.html#optuna.samplers.TPESampler

Paper, D. (2021). Convolutional Neural Networks. *TensorFlow 2.x in the Colaboratory Cloud*, 153–181. https://doi.org/10.1007/978-1-4842-6649-6_7

Qiu, Z., Wang, H., Liao, C., Lu, Z., & Kuang, Y. (2023). Sound Recognition of Harmful Bird Species Related to Power Grid Faults Based on VGGish Transfer Learning. *Journal of Electrical Engineering and Technology*, *18*(3), 2447–2456. https://doi.org/10.1007/S42835-022-01284-Z/FIGURES/8

Raschka, S. (s/f). *Deep Learning Models*. Recuperado el 23 de septiembre de 2023, de https://github.com/rasbt/deeplearning-models/blob/master/pytorch_ipynb/autoencoder/ae-cvae.ipynb

Rojano Aguilar, A., Salazar Moreno, R., Miranda, L., & Ojeda Bustamante, W. (2021). *Algoritmo Adam En La Inteligencia Artificial*. https://www.riego.mx/congresos/comeii2021/files/ponencias/extenso/COMEII-21005.pdf

Salamon, J., Jacoby, C., & Bello, J. P. (s/f). *UrbanSound8K - Urban Sound Datasets*. Recuperado el 15 de septiembre de 2023, de https://urbansounddataset.weebly.com/urbansound8k.html

Salamon, J., Jacoby, C., & Bello, J. P. (2014). A Dataset and Taxonomy for Urban Sound Research. *Proceedings of the 22nd ACM international conference on Multimedia*, 1041–1044. https://doi.org/10.1145/2647868.2655045

Saldanha, J., Chakraborty, S., Patil, S., Kotecha, K., Kumar, S., & Nayyar, A. (2022). Data augmentation using Variational Autoencoders for improvement of respiratory disease classification. *PLOS ONE*, *17*(8), e0266467. https://doi.org/10.1371/JOURNAL.PONE.0266467

Tian, Y. (2020). Artificial Intelligence Image Recognition Method Based on Convolutional Neural Network Algorithm. *IEEE Access*, *8*, 125731–125744. https://doi.org/10.1109/ACCESS.2020.3006097

Van de Kleut, A. (2020). *Variational AutoEncoders (VAE) with PyTorch*. https://avandekleut.github.io/vae/

Van Der Maaten, L., & Hinton, G. (2008). Visualizing Data using t-SNE. *Journal of Machine Learning Research*, *9*, 2579–2605.

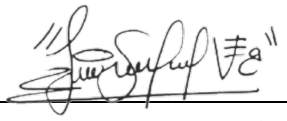
Wu, Z., Wang, S., Qian, Y., Yu, K., & Key, M. (2019). *Data Augmentation using Variational Autoencoder for Embedding based Speaker Verification*. https://doi.org/10.21437/Interspeech.2019-2248

Yalçın, O. G. (2021a). *Applied Neural Networks with TensorFlow 2*. Apress. https://doi.org/10.1007/978-1-4842-6513-0

Yalçın, O. G. (2021b). Convolutional Neural Networks. *Applied Neural Networks with TensorFlow 2*, 145–160. https://doi.org/10.1007/978-1-4842-6513-0_7

Zhao, K., Ding, H., Ye, K., & Cui, X. (2021). A Transformer-Based Hierarchical Variational AutoEncoder Combined Hidden Markov Model for Long Text Generation. *Entropy 2021, Vol. 23, Page 1277*, *23*(10), 1277. https://doi.org/10.3390/E23101277

FIRMA ESTUDIANTE 

FIRMA ASESORES 

Andrés Eduardo Castro O.

FECHA ENTREGA: 04/12/2023

FIRMA COMITÉ TRABAJO DE GRADO DE LA FACULTAD _____

RECHAZADO __ ACEPTADO ____ ACEPTADO CON MODIFICACIONES _____

ACTA NO. _____

FECHA ENTREGA: _____

FIRMA CONSEJO DE FACULTAD _____

ACTA NO. _____

FECHA ENTREGA: _____