Deep Deconvolution for Seismic Waves with Real and Simulated Distributed Acoustic Sensing Data: Final Report

JUAN C. AGUILERA C.[1], **DIEGO A. BADILLO S.**[1], **MAURICIO A. ARAVENA C.**[2], **AND RICARDO E. MARDONES T.**[2]

[1] *Master's Student in Electronic Engineering, UTFSM*
[2] *Bachelor's Student in Electronic Engineering, UTFSM*
[*] *juan.aguileraca@sansano.usm.cl, diego.badillo@sansano.usm.cl, maurio.aravena@sansano.usm.cl, ricardo.mardones@sansano.usm.cl*

July 27, 2022

Distributed Acoustic Sensing (DAS) is a method that allows the use of an optical fibre as an array of discrete sensors. Its implementation in already placed optical fibres for telecommunications can be useful for monitoring vibrations in structures and detection of earthquakes, among other things. In this work, a particular application is studied where an optical fibre is placed below a highway, and it is used for estimating the velocity of vehicles. A so called *Deconvolutional Autoencoder* is used to process the measured signals for a better estimation of the vehicles' velocity.
In this report, the Deconvolutional Autoencoder is retrained with the data consisting on the vibrations caused by vehicles on a highway to compare the results given in the original work. This implementation is compared with a classic minimisation algorithm for deconvolution called FISTA. Once the Deconvolutional Autoencoder is trained with the original data, it is adapted to new data by changing the activaction function of the output layer and the convolution method used for feedback.

1. INTRODUCCIÓN

La tecnología *Distributed Acoustic Sensing*[1], también conocida como *DAS* por sus siglas en inglés, está basada en el concepto de *Rayleigh scattering*, fenómeno físico que ocurre al propagarse luz por una fibra óptica. Esta es una de las variadas formas que existen de usar una fibra óptica para la medición de deformación y temperatura a lo largo de una fibra, y consiste en enviar luz coherente desde un interrogador por la fibra y medir las variaciones de la fase en la luz reflejada producto de *Rayleigh scattering*, las que son proporcionales a variaciones de temperatura o deformación que pueden ser mapeadas a distintos puntos de la fibra según el tiempo en que se detecten en el interrogador. De esta forma, una fibra óptica que ya haya sido instalada para propósitos de telecomunicaciones puede ser utilizada como un sensor distribuido que permite la medición de vibraciones acústicas en distintas posiciones de la fibra producidas por sismos u otras perturbaciones.

Para este proyecto se replicará la red propuesta en [2] para el aprendizaje no supervisado de deconvolución de vibraciones producidas por automóviles en una carretera, con el propósito de estimar su velocidad con una fibra óptica instalada longitudinalmente bajo el asfalto. La arquitectura usada es un autoencoder compuesto por una U-Net [3] y es entrenada retroalimentando la salida convolucionada por una respuesta a impulso conocida. Esta retroalimencación hace que la salida del autoencoder no sea una copia estimada de la entrada, sino su deconvolución. Los autores de [2] llaman a esta arquitectura particular un *Deconvolutional Autoencoder* o DAE (no confundir con *Denoising Autoencoder*). En el entrenamiento se usa una función loss que permite minimizar la disimilitud entre la reconstrucción de la señal de entrada, y a su vez penalizar la salida con una norma L_1, con el objetivo de obtener soluciones de deconvolución *sparse*.

Además de replicar los resultados entregados en el artículo de referencia [2], se propone expandir el uso del *Autoencoder Deconvolucional* a nuevos datos generados a partir de una respuesta impulso basada en un chirp lineal en frecuencias. La razón de la elección de este kernel para simular los datos es poder evaluar la posibilidad del uso del autoencoder para datos usados en la Tesis [4], que a su vez ocupa datos obtenidos en el estudio Poro-Tomo [5], en donde una señal acústica con chirp en frecuencias es generada en un terreno, y se obtienen mediciones reverberantes de manera distribuida con un sensor de fibra óptica basado en Distributed Acoustic Sensing.

El resto del manuscrito está estructurado de la siguiente forma: en la Sección 2 se describe el marco teórico de la deconvolución y autoencoder de manera resumida. En la Sección 3 se expone la metodología de trabajo para la recreación de los resultados en el artículo de referencia [2], con los respectivos resultados en la Sección 4. Las conclusiones del trabajo desarrollado son expuestas en la Sección 5.

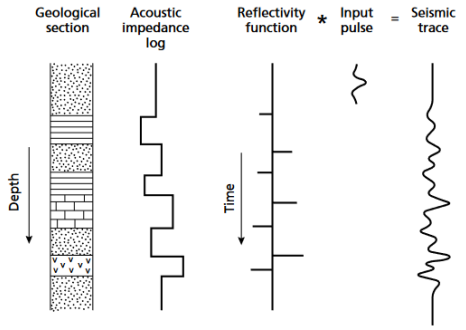

Figura 1. Modelo del esquema convolucional de reflexiones sísmicas. Adaptado de [8].

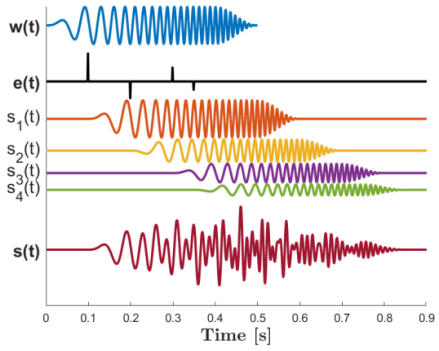

Figura 2. Modelo convolucional con $w(t)$ una señal con un chirp lineal en frecuencias y ruido aditivo $v(t) = 0$. La señal $s(t)$ medida presenta interferencias, por lo que las ubicaciones de las reflexiones son irreconocibles a simple vista. [4].

2. MARCO TEÓRICO

A. Deconvolución

Para propósitos de evaluación no destructiva de un sistema, resulta conveniente enviar una señal y medir las distintas reflexiones que el sistema produce para identificar averías o características de este. Bajo este contexto, es conveniente modelar el canal de propagación con un modelo convolucional [6], como se describe en Ec. (1).

$$s(t) = w(t) * e(t) + v(t) \tag{1}$$

En donde la señal medida $s(t)$ es el resultado de la convolución de un kernel $w(t)$ (correspondiente a la señal enviada) con una secuencia de impulsos *sparse* que representan reflexiones en el medio $e(t)$, más un ruido aditivo $v(t)$. La secuencia de impulsos $e(t)$ está relacionada con los cambios en impedancia acústica en el camino de propagación (ver Fig. 1).

El problema de la deconvolución en contextos geofísicos consiste en obtener un estimador de los impulsos dados por la señal *sparse* $e(t)$ que permiten obtener información de los cambios de impedancia acústica en un terreno. En teoría, es posible obtener una deconvolución directa en el dominio de Fourier mediante una división de espectros, y luego volver al dominio temporal. La presencia de ruido y errores de medición hacen que esto no entregue resultados óptimos, usualmente por problemas numéricos.

A partir del modelo convolucional propuesto surgen dos problemas iniciales [7]: 1) cuando dos pulsos en $e(t)$ están muy cerca entre sí, relativo al largo de $w(t)$, la señal $w(t)$ es superpuesta al medirla en $s(t)$, por lo que no es trivial identificar los instantes de tiempo en los que se produce la reflexión (ver Fig. 2); 2) cuando las diferencias de impedancia acústica son muy pequeñas relativos al ruido $v(t)$, el impulso asociado a la reflexión es pequeño, lo que implica que la reflexión en $s(t)$ relacionada a ese impulso presenta un SNR muy bajo, y es indistinguible del ruido.

B. Autoencoders

Los autoencoders, o AE, son una clase de redes neuronales de una única *hidden layer* entrenada de manera no supervisada, cuyo objetivo es lograr una copia de la entrada en la salida de la red [9]. La estructura se puede dividir en dos secciones: *encoder* y *decoder*.

Para lograr su objetivo, se busca que los AE aprendan una función, que se aproxime a la función identidad, de manera que se puedan obtener en la salida datos similares a los datos de entrada de la red. En este proceso, los AE intentan generar un vector que contenga, de manera comprimida, las características relevantes de la función de entrada, logrando una representación eficiente de los datos de entrada, propagando la información a través de la red hasta llegar a la *hidden layer* también conocida como *espacio latente* o el *bottle-neck*, dado que la cantidad de neuronas en esta capa es mucho menor a la cantidad presente en las capas de entrada o de salida.

B.1. Autoencoders sparse

El objetivo en este tipo denominado autoencoders sparse, o SAE [9, 10], es el mismo que los AE estándar, pero se logra de una manera diferente. En lugar de (o además de) provocar el *bottle-neck* característico de la estructura de los AE reduciendo el número de neuronas de las *hidden layers*, los SAE generan activaciones y desactivaciones de neuronas en las capas, generando una especie de *dropout*, y se construye la función de pérdida o *loss function* de modo que se penalicen dichas activaciones de las neuronas dentro de ciertas capas dadas, normalmente en la etapa del *encoder*. Se estimula la red para que aprenda a codificar y decodificar utilizando solo una pequeña cantidad de neuronas activas.

Existen diversas maneras de imponer las restricciones de *sparsity* a la red, pero la más ampliamente utilizada corresponden a la **penalización por norma L1**, donde se agrega un término a la función de pérdida, el cual penaliza la magnitud del vector de activaciones a en la capa h para cada iteración del entrenamiento, escalado por el hiper-parámetro λ tal como en la Ec. (2).

$$L'(x, \tilde{x}) = L(x, \tilde{x}) + \lambda \sum_i |a_i^{(h)}| \tag{2}$$

3. METODOLOGÍA

A. Deconvolutional Autoencoder

La arquitectura de la red presentada por el artículo de referencia [2] se muestra en la Fig. 3, la cual corresponde a una *light-weight* [1] U-Net Autoencoder con 305.089 parámetros entrenables, que toma como entrada un set de $N_q = 24$ formas de onda consecutivas de $N_t = 1024$ muestras temporales, organizada en una matriz de $N_q \; x \; N_t$. Esta entrada es entregada al Autoencoder, el cual se compone de tres capas convolucionales, seguida de

[1] Una arquitectura es *light-weight* cuando se propone como solución para hacer el uso de *deep neural networks* viable en dispositivos pequeños.

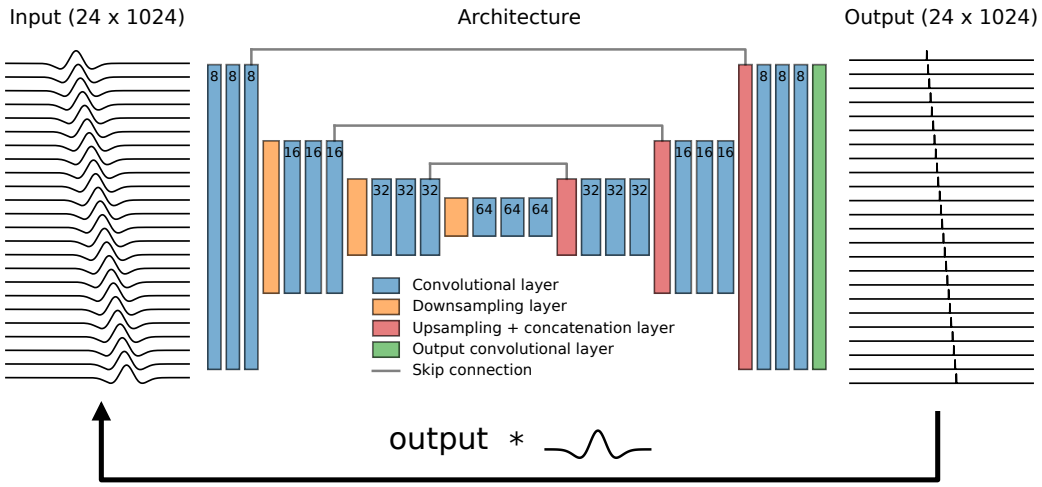

Figura 3. Esquema del Auto-Encoder de Deconvolución (DAE). Los datos DAS se introducen en una U-Net Auto-Encoder, cuya salida se convoluciona con una respuesta a impulso conocida (en el dominio temporal) para obtener una reconstrucción de la entrada. [2].

tres bloques de codificación, cada uno de estos bloques aplica una operación de *downsampling* y tres capas convolucionales. El *decoder* revierte la operación de codificación con tres bloques de *bilinear upsampling*. El rasgo característico de una U-Net, es la presencia de *skip-connections*, las cuales son conexiones que conectan directamente la salida de un bloque de codificación con su bloque de decodificación correspondiente. Esta concatenación es seguida de tres capas convolucionales. Finalmente la capa de salida es una simple capa convolucional con un canal de salida y una activación ReLU, que impone positividad en la salida del modelo. Cada capa convolucional a excepción de la capa de salida en el modelo es seguida de una función de activación *Swish* [11].

El factor clave para transformar esta arquitectura en un algoritmo de deconvolución auto-supervisado, es convolucionar la salida del modelo e con la respuesta impulso w a lo largo del eje temporal. De esta forma, para un *batch* de entradas $\{s_1, s_2, ..., s_{N_b}\}(e, s \in R^{N_q x N_t}, k \in R^{1 x N_k}, N_b = 128)$, la siguiente función objetivo es minimizada:

$$\mathcal{L} = \frac{1}{N_b} \sum_{i=1}^{N_b} (|| [w * e_i]_t - s_i ||_2^2 + \rho ||e_i||_1) \quad \textbf{(3)}$$

Donde $[w * e_i]_t$ denota la convolución entre w y e en el eje temporal, y ρ es un parámetro que pondera el regularizador l_1 de e. En el artículo de referencia se utiliza como hiper-parámetro $\rho = 10$. El regularizador $||e_i||_1$ induce a soluciones *sparse*, puesto que minimiza la norma l_1 de la salida, mientras que al mismo tiempo el término $||[w * e_i]_t - s_i||_2^2$ induce a la similitud de la convolución de la salida con la respuesta a impulso a la entrada, mediante la minimización de la norma l_2.

Esta arquitectura está disponible de manera pública por parte de los autores del artículo de referencia [2]. La implementación está hecha en TensorFlow como un objeto en Python, que contiene métodos para cada acción ejecutable sobre la red, ya sea en entrenamiento o validación, como la convolución con el kernel para la retroalimentación, y la definición de la función de minimización de la Ec. (3).

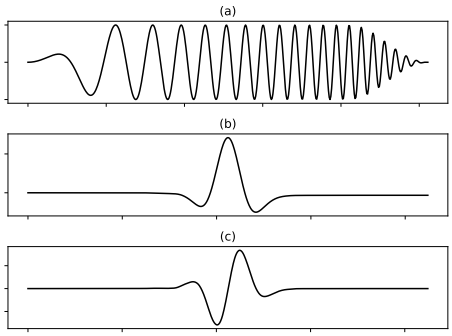

Figura 4. (a) Kernel simulado basado en un chirp lineal en frecuencias, basado en el chirp usado en el experimento PoroTomo [5], (b) Kernel integrado e invertido verticalmente a partir de (c) Kernel empírico obtenido a partir de las vibraciones causadas por vehículos [2].

B. Entrenamiento con los datos del artículo de referencia

Para el entrenamiento de la red, los autores proponen integrar tanto el kernel como los datos medidos directamente por la fibra óptica, ya que así obtuvieron empíricamente una mejor convergencia. No se detalla una explicación justificada del por qué, pero se da una hipótesis de que puede deberse a la simetría que se obtiene en las señales al realizar este paso (ver Fig. 4(b)).

Una vez integrados los datos y el kernel, se debe dividir el total de datos capturados, ordenados como una matriz de $24 \times N_{tot}$, con N_{tot} la cantidad total de muestras temporales capturadas. Esta división se hace para tener datos de entrenamiento separados de los datos de validación, y se hace en razón $50 - 50$. Además, se debe dividir cada sub-set en ventanas de 24×1024, de manera tal que puedan ser usadas como entrada de la red. Estas ventanas son extraídas aleatoriamente en lotes de 128 a partir del sub-set de entrenamiento. Se define una epoch para este caso cuando han sido usados 78 lotes. Esta implementación está disponible en el archivo `train.py` en nuestro repositorio github.com/Juanx65/DeepDeconvV2, para el que adaptamos la arquitectura facilitada por los autores.

Para la validación se procede de manera similar, separando los datos totales en dos sub-sets. El sub-set de validación es separado en ventanas de 24×1024, las cuales son entregadas como entrada a la red entrenada de manera secuencial, y luego concatenadas para formar la salida consistente de todas las ventanas del sub-set de validación.

C. Entrenamiento con datos simulados

Como una prueba de concepto para una próxima implementación a mediciones del estudio PoroTomo [5], cuyos datos son también usados en la Tesis [4] y que consisten de mediciones distribuidas por una fibra óptica de las vibraciones generadas arbitrariamente como un chirp lineal en frecuencias, se propone en este trabajo simular datos que consistan en la convolución del kernel de la Fig. 4(a) con impulsos distribuidos aleatoriamente en ventanas de 1024 muestras, cuyas amplitudes son también aleatorias en el intervalo $[-1, 1]$. De igual manera que para el caso de los datos de los autores del artículo de referencia, una epoch es considerada cuando han sido usados 78 lotes de 128 ventanas.

En la implementación original, en la última capa es usada la función de activación ReLU, cuya salida es mayor o igual a cero. Debido a que los nuevos datos son generados con impulsos que pueden tomar valores negativos, es necesario cambiar la función de activación de la última capa para poder obtener estimadores de los impulsos dentro de su recorrido. Para esto, utilizamos la función tangente hiperbólica como activación.

Además, en la implementación original, la deconvolución obtenida es *no-causal*. Esto quiere decir que en la convolución se consideran entradas futuras. Para los nuevos datos es necesario adaptar la retroalimentación del autoencoder deconvolucional de manera que calcule una convolución causal. Esto debido al contexto del problema, en donde cada impulso representa un punto de reflexión, por lo que no tiene sentido considerar entradas futuras si una onda incidente aún no llega a dicho punto de reflexión. Esto se ilustra de mejor forma en la Fig. 2.

D. FISTA

Los autores del trabajo detallan un algoritmo convencional de deconvolución, el cual fue usado como referencia para comparar los resultados obtenidos con DAE.

La tarea de deconvolucionar los datos DAS se adopta usando la versión acelerada del algoritmo ISTA, conocida como Fast-ISTA o FISTA [12]. A diferencia del autoencoder deconvolucional propuesto, este método procesa cada canal de manera independiente, sin considerar posibles correlaciones espaciales entre los canales.

4. RESULTADOS

A. Datos del artículo de referencia integrados

Una vez entrenadas 1000 epochs usando los híper-parámetros descritos en el artículo de referencia [2], y habiendo guardado la mejor epoch, procedemos a comparar nuestros resultados con el algoritmo tradicional FISTA y los pesos entregados por los autores. Para esto escogimos dos secciones del dataset de validación que contengan comportamientos interesantes y distintivos. En la Fig. 5, columna izquierda, se muestra un caso en que hay vehículos moviéndose en ambos sentidos, generando interferencias, y en la columna derecha solamente en un sentido. Puntos importantes a destacar de esta figura es que para FISTA se obtienen impulsos tanto negativos como positivos, mientras que el DAE entrega solo impulsos positivos por la función de activación ReLU en la última capa. En los casos (e),(i) se puede observar un impulso con mayor amplitud que el resto entre los segundos 20 y 25, incluso cuando en los datos originales no hay una vibración de mayor amplitud en este punto. Esto es debido al procesamiento hecho en ventanas de 1024 muestras temporales. En este caso ese impulso corresponde a los extremos de una de las ventanas usadas en la evaluación. En el caso de FISTA

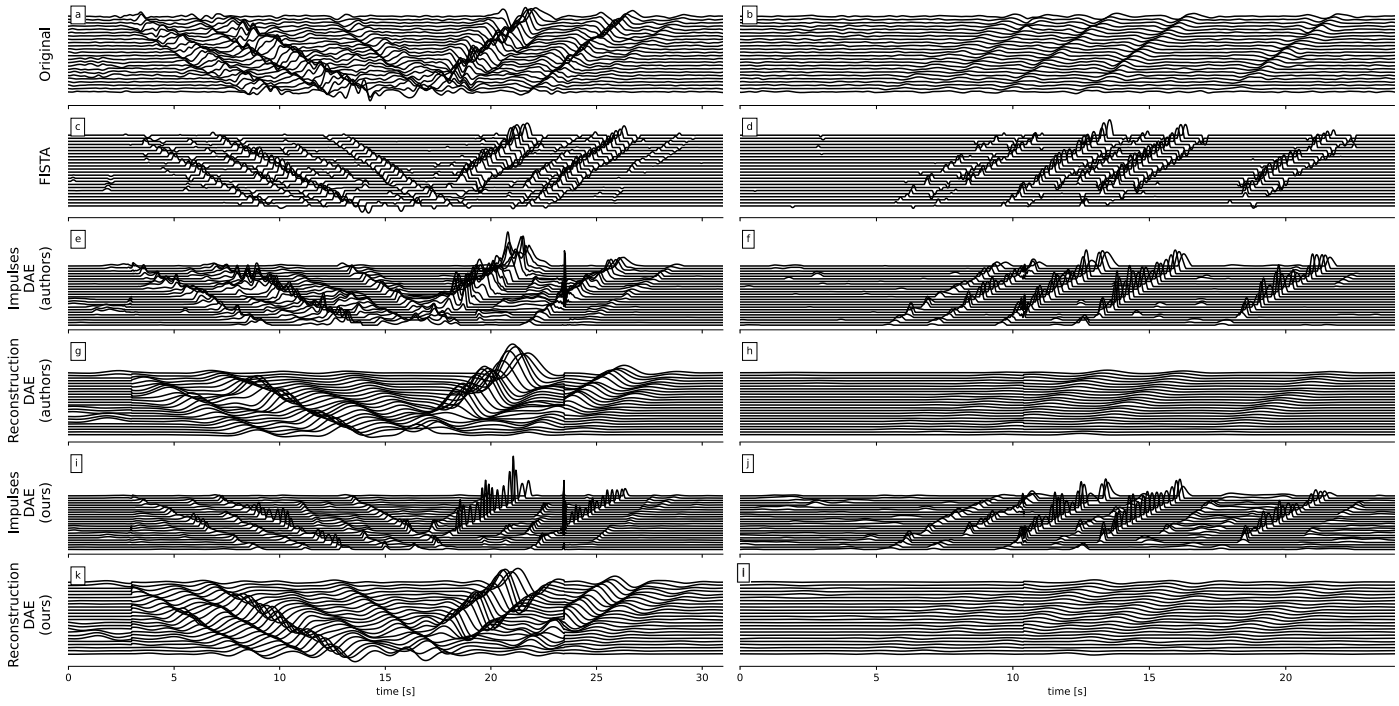

Figura 5. Comparación de resultados con distintos métodos par dos ejemplos del dataset de validación.

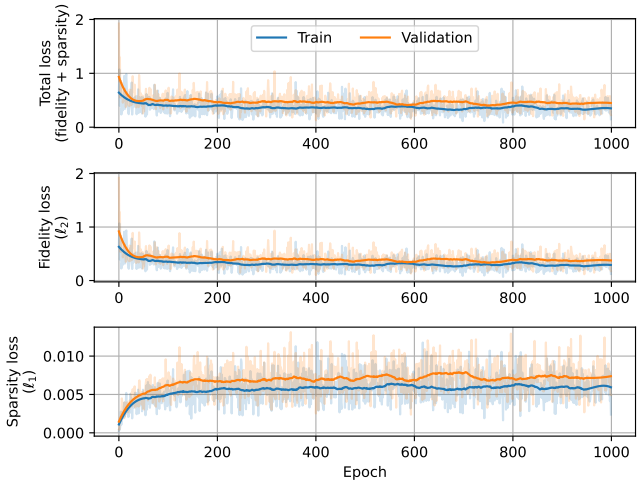

Figura 7. Funciones de pérdida para el entrenamiento con 1000 épocas usando la metodología y datos del artículo de referencia [2].

(c),(d), el procesamiento también es hecho en ventanas, pero el largo de la ventana usado en este caso es mayor (5000 muestras temporales), razón por la cuál no se encuentra este ejemplar.

Al hacer la comparación para los casos (e),(i) correspondientes a pesos de los autores y a los pesos resultantes de nuestro entrenamiento, observamos que nuestros pesos entregan impulsos más angostos para este ejemplo en particular, lo que a su vez genera una leve diferencia en la reconstrucción para los casos (g),(k). Esta *mejora* en desempeño (impulsos más angostos) para este caso particular puede deberse a la forma aleatoria en que se separan los datos para la etapa de entrenamiento. Es posible que en nuestro entrenamiento haya habido un sesgo en el que la red haya observado casos similares al mostrado, produciendo un mejor resultado para este ejemplo en particular. Por su parte, para el caso de los resultados en el ejemplo de la columna derecha, obtenido con los pesos de los autores resulta ser más *sparse* que el obtenido con nuestros pesos.

En la Fig. 7 se adjuntan los resultados de entrenamiento para cada época, en donde las curvas resaltadas representan un suavizado de las funciónes de pérdida para cada época. Aquí se observa que la disminución de total loss por época tiene una pendiente baja, al punto de que no vale la pena entrenar las 1000 épocas para obtener resultados útiles. En nuestro caso, se llega a una pérdida total de alrededor de 0.4, mientras que en el artículo de referencia se llega a una pérdida total menor de aproximadamente 0.2.

B. Datos simulados, un canal

Una vez hechos los cambios propuestos a la arquitectura, esto es, cambiar la función de activación de la última capa por una tangente hiperbólica y cambiar el método de convolución a uno causal, se obtienen resultados para un único canal de medición, estos se muestran en la Fig. 6. En un principio, los resultados obtenidos directamente después de hacer estos cambios son los de la columna izquierda, que no logran obtener una deconvolución adecuada. Además, las pérdidas para las épocas de entrenamiento se adjuntan en la Fig. 9, en donde se puede ver que no se logra generalizar al obtener una pérdida de validación ascendente después de la época 50. Con esto, y al observar que en la reconstrucción obtenida en la Fig. 6(e) presenta altas frecuencias al comienzo, concluimos que había un error en la orientación del kernel.

En la columna derecha de la Fig. 6 se muestran los resultados obtenidos una vez volvimos a entrenar pero esta vez invirtiendo el kernel. Los resultados ahora sí son adecuados, lo que se refleja también en las funciones de pérdida que se muestran en la Fig. 10.

C. Datos simulados, 24 canales

Una vez comprobado el funcionamiento del DAE con un canal, simulamos retardos proporcionales entre los canales para evaluar la capacidad del DAE de aprender correlaciones espaciales con un el kernel chirp. Los resultados para este caso se encuentran adjuntos en la Fig. 8, en donde se observa que la red logra generalizar para distintos desfases sintéticos, en ambos casos para datos del dataset de validación.

D. Datos del artículo de referencia sin integrar

En la Sección 3 señalamos que los autores propusieron integrar los datos y el kernel para obtener mejores resultados de convergencia. Su hipótesis es que la simetría del kernel afecta en ello. Sin embargo, al probar con un kernel chirp para datos simulados y obtener buenos resultados, podemos descartar asimetría como razón para peor convergencia.

La razón probable por la que los autores obtuvieron mejores resultados para un kernel simétrico se debe a que en su arquitectura propuesta, en el método en donde se ejecuta la retroalimentación con convolución fue aplicada la correlación cruzada en vez de la convolución, en donde la diferencia es la inversión horizontal del kernel. La función `Conv2D` es usada para este propósito.

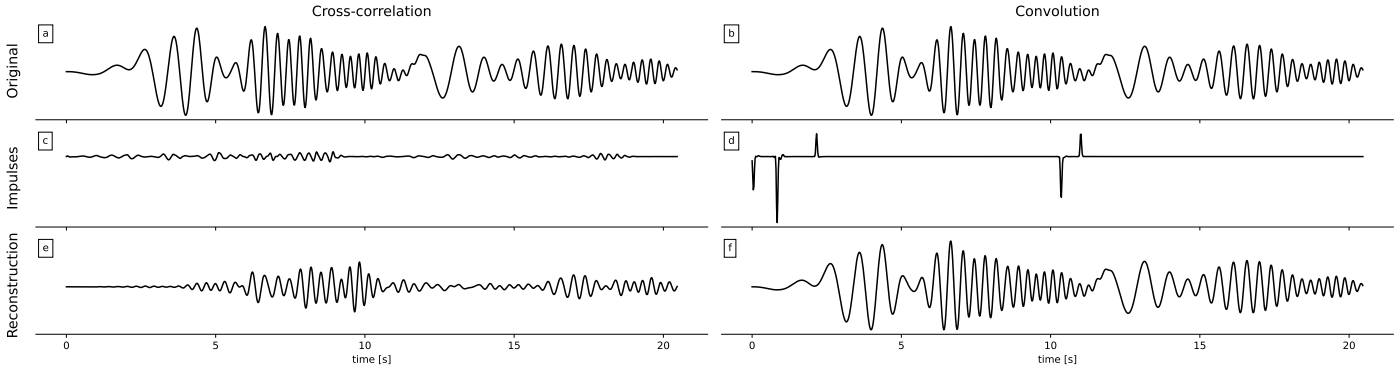

Figura 6. Ejemplos para resultados con datos simulados de un canal con un kernel basado en un chirp lineal de frecuencias.

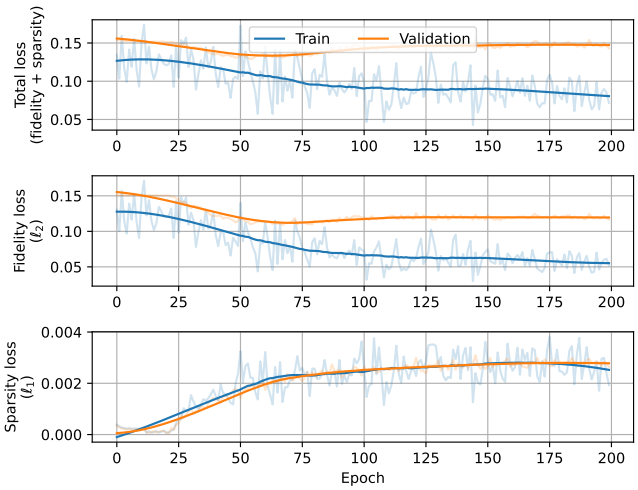

Figura 9. Funciones de pérdida para el entrenamiento de datos simulados chirp sin inversión del kernel (correlación cruzada).

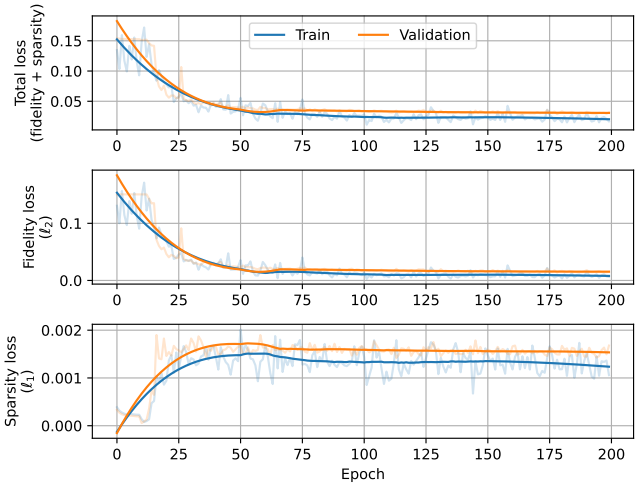

Figura 10. Funciones de pérdida para el entrenamiento de datos simulados chirp con inversión del kernel (convolución).

En la Fig. 12 muestra los resultados al entrenar los datos sin integración en el kernel ni en datos medidos (ver Fig. 4(c)) usando la red de los autores sin modificaciones, lo que calcula una correlación cruzada en la retroalimentación del DAE; y para la red modificada, que realiza correctamente la convolución en la retroalimentación.

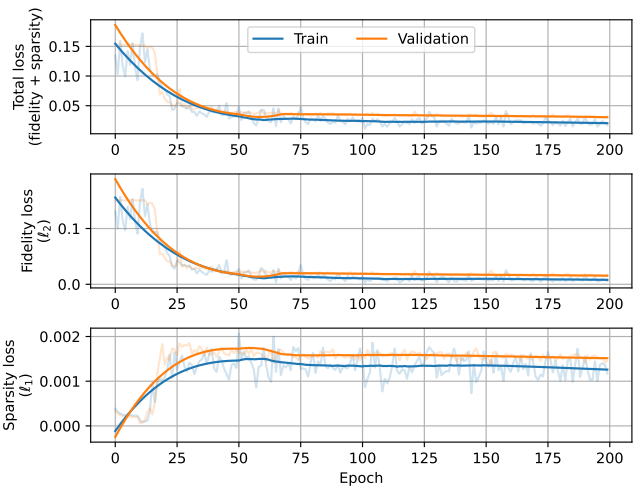

Figura 11. Funciones de pérdida para el entrenamiento de datos simulados chirp con 24 canales desfasados entre sí.

En la Fig. 12(d), en el intervalo $[10, 17]$ segundos, se observa que un impulso central resulta como resultado de la deconvolución, mientras que en la Fig.12(c), en el mismo intervalo, se observan dos impulsos para representar la misma entrada. Además, en el intervalo $[0, 5]$ segundos, en donde se visualiza más claramente a la entrada la forma del kernel de la Fig. 4(c), se observa cómo para el caso de la correlación cruzada se representa con dos impulsos mientras que para la convolución solo con uno.

E. Desempeño

Para evaluar el desempeño de la red implementada, se utilizaron dos métrica: el error de los resultados respecto a los datos *ground truth* , considerando MSE y desviación estándar de este mismo y por otro lado, el tiempo requerido para completar la inferencia por parte de la red. Para ello procesan 10.000 ventanas de 1024 datos. Los resultados obtenidos para la medición de error se encuentran en el Cuadro 1 mientras que los tiempos de inferencia se encuentran en el Cuadro 2.

Se puede ver que los errores obtenidos son bajos en promedio, sin embargo, se debe tener en consideración que estos se obtuvieron a partir de datos generados mediante simulación, los cuales no presentan ninguna clase de ruido ni dispersión entre las muestras. En el caso de probar el DAE con datos reales, se

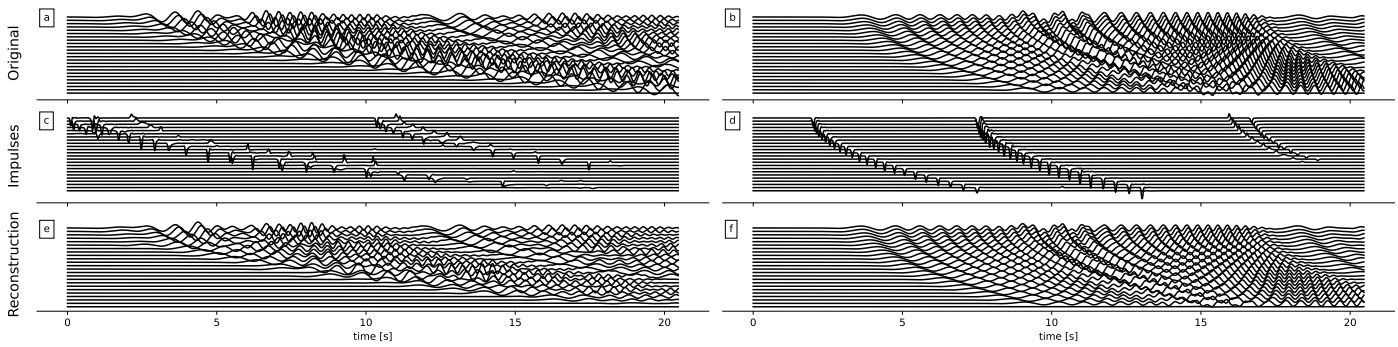

Figura 8. Ejemplos para resultados con datos simulados de 24 canales con un kernel basado en un chirp lineal de frecuencias.

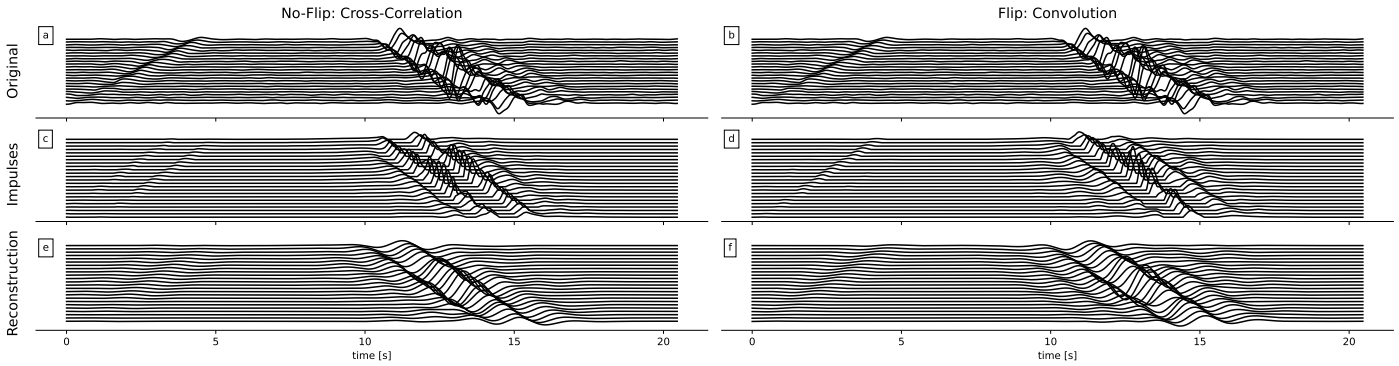

Figura 12. Comparación de resultados para los datos del artículo de referencia sin integración. En la columna izquierda se entrena con la arquitectura sin modificar de los autores, y en la columna derecha se modifica la retroalimentación de tal manera que se ejecute una convolución

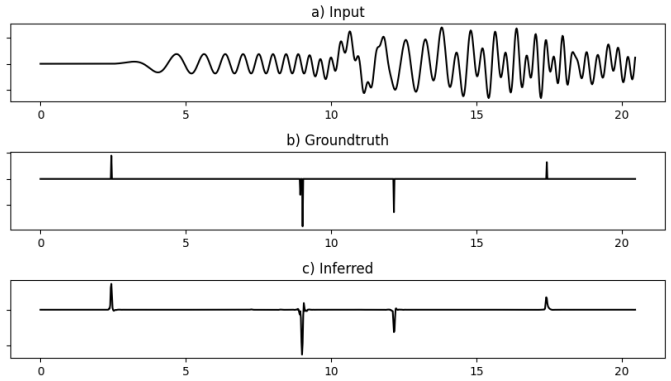

Figura 13. Comparación de groundtruth y resultados por DAE para datos simulados.

espera que exista dispersión (ensanchamiento temporal) en el kernel debido a la propagación en el medio, lo que significaría que el kernel no sería capaz de representar perfectamente estos casos con pocos impulsos, y por lo tanto haya un mayor error de inferencia.

MSE promedio	0.0013
Desviación estándar MSE	0.0190

Cuadro 1. Evaluación de resultados frente a *groundtruth* obtenidos para la red asociada al kernel *chirp* y los datos simulados. Los datos se obtienen luego de procesar 10.000 ventanas de 1024 datos.

Tiempo de inferencia promedio	0.0169 s
Desviación estándar en los tiempos	0.0357 s

Cuadro 2. Resultados de tiempos de inferencia, para la red implementada y entrada para los datos asociados al kernel *chirp*. Los datos se obtienen luego de procesar 10.000 ventanas de 1024 datos.

5. CONCLUSIONES

La arquitectura de la red U-Net encoder-decoder resulta útil por su esencia para representaciones sparse [10]. Considerando el modelo convolucional para señales geofísicas (Ec. (1) y Fig. 1), bajo los supuestos de que los cambios de impedancia acústicas en una subsuperficie son discretos, y por lo tanto las reflexiones de una onda enviada se producirían en ubicaciones puntuales, resulta conveniente pensar en la posibilidad del uso de un autoencoder para la deconvolución en este contexto.

La red DAE (*Deconvolutional Autoencoder*) propuesta en [2] fue utilizada para mejorar la resolución de captura de vibraciones de vehículos sobre una carretera para poder estimar con mayor precisión la velocidad de los vehículos especialmente cuando hay superposición de vibraciones (dos vehículos produciendo vibraciones en el mismo instante en el mismo punto espacial de medición). La implementación de un método basado en *Deep Learning* en contraste al método clásico de optimización permite una inferencia más de cuatrocientas veces más rápida que el método de optimización FISTA, el cual se utiliza como métrica de comparación, con el mismo hardware. Otra ventaja del uso de la red DAE propuesta para la deconvolución de varios canales correlacionados, es que en la etapa de entrenamiento, debido a que las entradas son matrices de altura equivalente a la cantidad de canales, y ancho equivalente a la cantidad de muestras, se infiere una correlación espacial entre canales, que para el caso del algoritmo de comparación FISTA es inexistente, puesto que se debe aplicar la deconvolución para cada canal de manera independiente.

Al aplicar el DAE a los nuevos datos pudimos observar que la arquitectura usada por los autores de [2] realiza una *de-correlación cruzada* en vez de una deconvolución, al no invertir el kernel en la retroalimentación. Este error es debido a que las bibliotecas de aprendizaje de máquinas suelen usar indistintamente el término de convolución aplicado a correlación cruzada y convolución [9]. Al utilizar la inversión correcta del kernel, se obtiene una mejora en en la convergencia.

El computador usado para el procesamiento cuenta con las siguientes especificaciones: Intel® Core™ i7-10750H, Nvidia RTX 2060 Max-Q 6GB, 32GB DDR4 3200MHz. Con este equipo se desarrolló un entrenamiento que tardó aproximadamente 13 horas para 1000 épocas con el objetivo de lograr resultados similares a los presentados en el paper de referencia, sin embargo, se observó que el *loss* bastaba con únicamente 200 épocas para lograr la convergencia esperada, por lo que se estableció esta cantidad co-

mo límite de entrenamiento para las siguientes evaluaciones de la red, para la cual el entrenamiento tomaba aproximadamente 2 horas. El repositorio para este proyecto se encuentra disponible públicamente en github.com/Juanx65/DeepDeconvV2.

REFERENCIAS

1. A. H. Hartog, *An Introduction to Distributed Optical Fibre Sensors* (CRC Press, 2017).
2. M. van den Ende, A. Ferrari, A. Sladen, and C. Richard, "Deep Deconvolution for Traffic Analysis with Distributed Acoustic Sensing Data," EarthArXiv (2021).
3. O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," (2015).
4. F. Muñoz Carmona, "Blind near-field array signal processing to enhance fibre-optic distributed acoustic sensing capabilities," Master's thesis (2022).
5. K. L. Feigl and L. M. Parker, "PoroTomo Final Technical Report: Poroelastic Tomography by Adjoint Inverse Modeling of Data from Seismology, Geodesy, and Hydrology," (2019).
6. M. O'Brien, T. Sinclair, and S. Kramer, "Recovery of a Sparse Spike Time Series by L1 Norm Deconvolution," Signal Process. IEEE Transactions on **42**, 3353 – 3365 (1995).
7. Y. Chang, Y. Zi, J. Zhao, Z. Yang, W. He, and H. Sun, "An adaptive sparse deconvolution method for distinguishing the overlapping echoes of ultrasonic guided waves for pipeline crack inspection," Meas. Sci. Technol. **28**, 035002 (2017).
8. P. Kearey and M. Brooks, *An introduction to geophysical exploration /* (Wiley Blackwell„ 2002.).
9. I. J. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT Press, Cambridge, MA, USA, 2016). http://www.deeplearningbook.org.
10. D. Arpit, Y. Zhou, H. Ngo, and V. Govindaraju, "Why Regularized Auto-Encoders learn Sparse Representation?" (2015).
11. P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for Activation Functions," (2017).
12. A. Beck and M. Teboulle, "A fast Iterative Shrinkage-Thresholding Algorithm with application to wavelet-based image deblurring," in *2009 IEEE International Conference on Acoustics, Speech and Signal Processing,* (2009), pp. 693–696.