UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE ELECTRÓNICA
VALPARAÍSO - CHILE

"Estimación del flujo glótico a partir de señales de acelerómetro mediante aprendizaje profundo"

Juan Pablo Riquelme Romero

MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELECTRÓNICA

PROFESOR GUIA: **Juan Pablo Cortés**

PROFESOR CORREFERENTE: **Matías Zañartu**

CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE MONOGRAFÍA A REPOSITORIO ACADÉMICO

1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

Tipo de monografía (marcar una opción): ☒ Memoria o trabajo de título ☐ Tesis de Postgrado

Título del trabajo: Estimación del flujo glótico a partir de señales de acelerómetro mediante aprendizaje profundo

Nombre del candidato(a): Juan Pablo Riquelme Romero

Carrera / Grado: Ingeniería Civil Electrónica

Campus: Casa Central **Departamento:** Electrónica

2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, ___Matías Zañartu_____, en mi calidad de profesor(a) guía/director(a) del trabajo académico mencionado anteriormente **DEJO CONSTANCIA** que:

- He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.
- El trabajo cumple con los requisitos académicos y de formato establecidos por la institución.

3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL (marcar una opción)

☒ El trabajo **NO contiene** información que amerite confidencialidad y puede ser publicado de inmediato en repositorio con acceso abierto.

☐ El trabajo **CONTIENE** información con potenciales implicancias de propiedad industrial o intelectual y requiere un periodo de confidencialidad (**embargo**) por **(marcar una opción)**:

☐ 6 meses ☐ 12 meses ☐ 2 años ☐ 3 años ☐ 5 años ☐ 10 años

Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):

4.- FIRMAS

Profesor(a) guía o director(a) de memoria o tesis:

Fecha: ___20/10/2025___ **Firma:** _____

Estudiante o Candidato(a):

Fecha: 15/10/2025 **Firma:** _____

Este formulario debe ser insertado como página 2 de la memoria o tesis, completado y firmado por estudiante y profesor(a) antes de la entrega en portal PRISMA de Biblioteca USM.

Resumen

La medición del flujo glótico es fundamental para el estudio de la voz pero su obtención directa es invasiva y compleja. Una alternativa no invasiva es el uso de acelerómetros en el cuello, aunque la estimación del flujo a partir de estas señales también presenta desafíos. El objetivo de este trabajo es desarrollar un método basado en aprendizaje profundo para estimar la forma de onda del flujo glótico a partir de las señales de un acelerómetro captadas en la superficie del cuello. Para lograr esto, se utilizó una base de datos existente con señales de voz y acelerometría para generar un conjunto de datos para el entrenamiento. Las señales de flujo glótico de referencia se obtuvieron aplicando el método de filtrado inverso de fase cuasi-cerrada (QCP) a las señales de voz grabadas con micrófono. Se implementó un sistema automatizado para procesar y validar las señales. Posteriormente, se diseñó, entrenó y optimizó una Red Convolucional Temporal (TCN) para aprender a transformar las secuencias de la señal de acelerómetro en las secuencias de flujo glótico correspondientes. Los resultados de la evaluación muestran que el modelo puede estimar la morfología general del flujo glótico. El rendimiento del modelo depende del género y de la condición vocal del hablante, observándose un mayor grado de error en las voces femeninas con patología. El trabajo demuestra la viabilidad de utilizar el modelo para realizar un filtrado inverso sin necesidad de calibración por sujeto.

Abstract

The measurement of glottal flow is fundamental for the study of the voice, but its direct acquisition is invasive and complex. A non-invasive alternative is the use of accelerometers on the neck, although estimating the flow from these signals also presents challenges. The objective of this work is to develop a method based on deep learning to estimate the waveform of the glottal flow from accelerometer signals captured on the neck surface. To achieve this, an existing database with voice and accelerometry signals was used to generate a dataset for training. The reference glottal flow signals were obtained by applying the quasi-closed phase (QCP) inverse filtering method to the voice signals recorded with a microphone. An automated system was implemented to process and validate the signals. Subsequently, a Temporal Convolutional Network (TCN) was designed, trained, and optimized to learn how to transform the accelerometer signal sequences into the corresponding glottal flow sequences. The evaluation results show that the model can estimate the general morphology of the glottal flow. The model's performance depends on the speaker's gender and vocal condition, with a higher degree of error observed in female voices with pathology. The work shows the feasibility of using the model to perform inverse filtering without the need for subject-specific calibration.

Glosario

ACC Acelerómetro.

BiLSTM Red neuronal de memoria a corto y largo plazo bidireccional (Bidireccional Long Short-Term Memory).

CCD Descomposición basada en Cepstrum Complejo (Complex Cepstrum-based Decomposition).

CP Análisis de fase cerrada (Closed phase).

GCI Instante de cierre glotal (Glottal Closure Instant).

GIF Filtrado inverso glotal (Glottal Inverse Filtering).

IAIF Filtrado inverso adaptativo iterativo (Iterative Adaptative Inverse Filtering).

IBIF Filtrado inverso basado en impedancia subglotal (Impedance-Based Inverse Filtering).

LP Predicción lineal (Linear Prediction).

LSTM Red neuronal de memoria a corto y largo plazo (Long Short-Term Memory).

MAE Error Absoluto Medio (Mean Absolute Error).

MIC Micrófono.

MSE Error Cuadrático Medio (Mean Squared Error).

QCP Filtrado inverso de fase cuasi-cerrada (Quasi-Closed Phase).

RMSE Raíz del Error Cuadrático Medio (Root Mean Squared Error).

RNN Red neuronal recurrente (Recurrent Neural Network).

TCN Red Convolucional Temporal (Temporal Convolutional Network).

WLP Predicción lineal ponderada (Weighted Linear Prediction).

Índice

1. Introducción

La producción de la voz humana es el resultado de un proceso coordinado que se origina en los pulmones, los cuales suministran un flujo de aire que atraviesa la laringe. En esta estructura se encuentran las cuerdas vocales, cuya vibración modula dicho flujo y lo convierte en una fuente de energía acústica [1]. Esta señal de excitación, conocida como flujo glótico, es posteriormente filtrada por las cavidades del tracto vocal, que le otorgan sus características resonantes. Finalmente, la señal acústica es radiada desde los labios hacia el exterior [2].

El flujo glótico contiene información fundamental sobre los patrones vibratorios de las cuerdas vocales. Por esta razón, su análisis permite caracterizar la producción de la voz en distintas aplicaciones. En el ámbito clínico, por ejemplo, el estudio del flujo glótico es útil para identificar diferentes tipos de fonación, como la modal, soplada o tensa [3]. Esto es relevante, ya que ciertos modos de fonación pueden estar asociados al desarrollo de trastornos vocales [4]. En el campo de la tecnología, el flujo glótico se utiliza para mejorar la naturalidad de los sistemas de síntesis de voz y para extraer características que complementan a las del tracto vocal en tareas de reconocimiento de hablantes [5].

A pesar de su importancia, la medición directa del flujo glótico es compleja. La vibración de las cuerdas vocales ocurre a una velocidad muy alta y en una zona de difícil acceso, lo que exige el uso de equipamiento especializado y a menudo invasivo, como la videoendoscopía de alta velocidad [1]. Para superar estas dificultades, se han desarrollado métodos no invasivos para estimar el flujo glótico de manera indirecta. Una de las técnicas más extendidas es el filtrado inverso glotal (Glottal Inverse Filtering, GIF).

El filtrado inverso se basa en la idea de revertir computacionalmente el proceso de producción de la voz. Por ejemplo, a una señal de voz, se le busca cancelar los efectos del tracto vocal y de la radiación en los labios para obtener una estimación de la señal de origen: el flujo glótico [1]. Aunque estos métodos son atractivos por su simplicidad, presentan limitaciones. Su rendimiento se degrada en voces con una frecuencia fundamental alta y su precisión depende de supuestos como la linealidad del tracto vocal y la independencia entre la fuente y el filtro, los cuales no siempre se cumplen en la producción de voz real [5].

Una alternativa al uso de micrófonos es registrar las vibraciones de la superficie del cue-

llo mediante un acelerómetro. Estas señales son menos sensibles al ruido ambiental, no contienen información inteligible del habla (resguardando la privacidad del usuario) y, por lo tanto, son adecuadas para el monitoreo ambulatorio de la función vocal en condiciones cotidianas[6]. Sin embargo, el filtrado inverso de señales de acelerómetro también enfrenta desafíos. Los métodos existentes, como el filtrado inverso basado en impedancia subglotal (IBIF), dependen de modelos físicos que requieren una calibración específica para cada sujeto y pueden ser sensibles a factores como la posición del sensor o las propiedades mecánicas de la piel.

En los últimos años, el aprendizaje profundo ha surgido como una alternativa a los enfoques basados en modelos físicos. Las redes neuronales profundas tienen la capacidad de aprender relaciones complejas directamente a partir de los datos, sin necesidad de asumir un modelo explícito del sistema fonador [7]. Esta propiedad las convierte en una herramienta prometedora para abordar la tarea de filtrado inverso glotal, ya que podrían aprender a mapear la señal de un acelerómetro al flujo glótico correspondiente, superando algunas de las limitaciones de los métodos tradicionales.

2. Objetivos

El objetivo general de esta memoria es desarrollar un método basado en aprendizaje profundo para estimar la forma de onda del flujo glótico a partir de señales de acelerómetro captadas en la superficie del cuello.

Para alcanzar este objetivo, se definen los siguientes objetivos específicos:

- Implementar un sistema automatizado para procesar y validar señales de voz y acelerometría provenientes de una base de datos, con el fin de generar un conjunto de datos apto para el entrenamiento de un modelo de aprendizaje profundo.
- Obtener señales de flujo glótico de referencia aplicando el método de filtrado inverso de fase cuasi-cerrada (QCP) a las señales de voz registradas con micrófono.
- Diseñar, entrenar y optimizar una Red Convolucional Temporal (TCN) para aprender a transformar secuencias de la señal de acelerómetro en las secuencias de flujo glótico de referencia correspondientes.

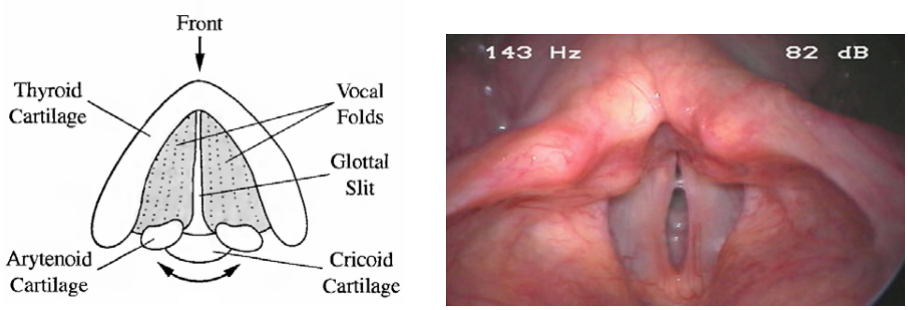
- Evaluar el rendimiento del modelo entrenado mediante métricas cuantitativas y comparaciones visuales entre las estimaciones y las señales de referencia en un conjunto de datos no utilizado durante el entrenamiento.

3. Marco Teórico

En esta sección se detallan los conceptos fundamentales necesarios para comprender el problema, considerando las herramientas y enfoques disponibles que pueden contribuir al logro del objetivo propuesto.

3.1. Flujo Glotal

La velocidad volumetrica del flujo glotal, o simplemente flujo glotal, se refiere al volumen de aire que atraviesa la glotis por unidad de tiempo durante la vibración de las cuerdas vocales, medido en litros por segundo (L/s) [1]. En la Figura 1a, se presenta un esquema anatómico de la laringe en el que se identifican la glotis y las cuerdas vocales, mientras que la Figura 1b muestra una imagen real de las cuerdas vocales en vibración.



(a) Esquema de la laringe humana visto desde arriba. Tomado de [8].

(b) Vista superior real de las cuerdas vocales en vibración. Tomado de [1].

Figura 1: Vista esquemática y real de la laringe humana desde arriba.

La glotis, el espacio entre las cuerdas vocales, cambia de tamaño según el movimiento de estas, abriéndose o cerrándose dependiendo de su separación o aducción.

Las cuerdas vocales pueden encontrarse principalmente en tres estados básicos: respiración, sonoridad y no sonoridad. En el estado de respiración se mantiene una glotis amplia, donde el aire proveniente de los pulmones fluye libremente a través de la glotis [8].

Por otra parte, en el estado de sonoridad, las cuerdas vocales se aducen, es decir, se aproximan para reducir o cerrar la glotis. La contracción de los pulmones inicia el flujo de aire, lo que provoca una acumulación de presión por debajo de la glotis. Cuando esta presión subglótica supera un cierto umbral, las cuerdas vocales entran en un estado de vibración autosostenida. Esta vibración modula el flujo de aire glotal, generando una serie de pul-

sos cuasiperiódicos caracterizados por ciclos de apertura y cierre de las cuerdas vocales. Como resultado, el flujo glotal adquiere una estructura cuasiperiódica que actúa como la fuente sonora que excita el tracto vocal [9]. Una representación del flujo glotal se puede observar en la Figura 2.

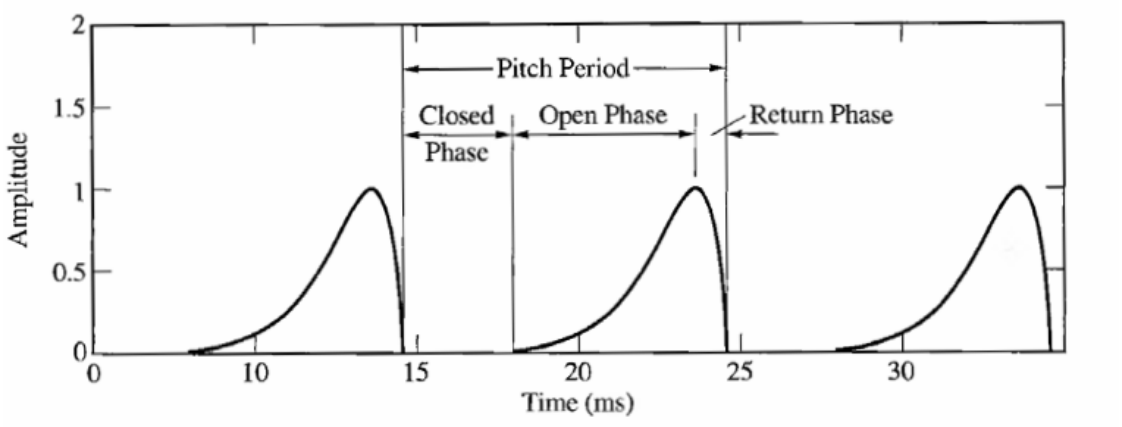


Figura 2: Representación de la forma de onda del flujo glotal. Tomado de [8].

A partir de la Figura se identifican tres fases: la fase glotal cerrada, correspondiente al intervalo en el que las cuerdas vocales permanecen completamente cerradas, impidiendo el paso del flujo de aire. La fase glotal abierta, que inicia cuando el flujo de aire deja de ser nulo y aumenta progresivamente hasta alcanzar su velocidad máxima. Y la fase de retorno, que abarca desde el momento en que se alcanza dicha velocidad máxima hasta el cierre completo de las cuerdas vocales. Cabe señalar que esta secuencia representa una generalización del comportamiento del flujo glotal. En algunos casos, como se verá más adelante, puede no observarse un cierre completo de las cuerdas vocales.

La duración de un ciclo glotal se denomina período de tono, y su recíproco corresponde a la frecuencia fundamental. Diversos factores influyen en la velocidad de oscilación de las cuerdas vocales, tales como la tensión muscular, la masa de las cuerdas vocales y la presión del aire detrás de la glotis, proveniente de los pulmones y la tráquea. El rango típico de frecuencia fundamental en humanos varía entre aproximadamente 60 Hz y 400 Hz. Generalmente, los hombres presentan una frecuencia fundamental más baja que las mujeres debido a que sus cuerdas vocales son más largas y más masivas [8].

El estado de no sonoridad es similar al de respiración, en tanto que las cuerdas vocales permanecen separadas. Sin embargo, en algún punto del tracto vocal se produce una constricción que genera turbulencia [5], lo que da lugar a los sonidos fricativos. Por ejemplo, en la producción de /s/, las cuerdas vocales permanecen separadas y se produce una

constricción con la punta de la lengua, lo que genera la turbulencia característica de este sonido [10].

Como se explicó anteriormente, el flujo glotal se define como el volumen de aire que atraviesa la glotis por unidad de tiempo durante la vibración de las cuerdas vocales. Por ello, el análisis de filtrado inverso glotal se enfoca específicamente en el estado sonoro de las cuerdas vocales, es decir, en la fonación, entendida como la producción de sonido a partir de las vibraciones de las cuerdas vocales generadas por cambios en la presión del aire [11]. En particular, se estudian las vocales, ya que se producen en este estado de sonoridad y, acústicamente, presentan mayor duración y energía en comparación con otros segmentos del habla [1], lo que las hace especialmente adecuadas para el estudio del flujo glotal.

La fonación puede variar tanto en intensidad como en tipo (lo que se conoce también como calidad vocal), en función de la activación de los músculos laríngeos y del esfuerzo respiratorio [12]. Dado que el flujo glotal contiene información relevante sobre los patrones vibratorios de las cuerdas vocales, cada tipo de fonación se asocia con una forma de onda característica y con propiedades espectrales específicas [13]. Es importante tener en cuenta los tipos de fonación, ya que la forma de onda del flujo glotal asociada a cada uno puede comprometer la eficacia de los métodos de filtrado inverso.

Algunos de los tipos de fonación que puede producir el ser humano son la modal, la soplada, la tensa, la crujiente, el falsete, entre otros. Estas variantes se sitúan a lo largo de un continuo de calidad vocal, en el cual, la fonación soplada y la fonación presionada (o tensa) representan los extremos opuestos. La fonación modal se considera el punto de referencia al comparar distintos tipos de fonación, dado que se caracteriza por una vibración periódica de las cuerdas vocales, baja tensión laríngea y un cierre glotal abrupto. En el extremo, la fonación soplada implica una menor tensión laríngea y un cierre glotal incompleto, lo que genera ruido turbulento y, a diferencia de la fonación modal, hace que la estructura armónica sea más prominente en las bajas frecuencias. En cambio, la fonación presionada presenta una mayor tensión aductiva y longitudinal, con un cierre glotal más brusco, lo que acentúa los armónicos de alta frecuencia [12]. La Figura 3 muestra los espectrogramas de los flujos glotales estimados correspondientes a las fonaciones soplada, modal y presionada, respectivamente.

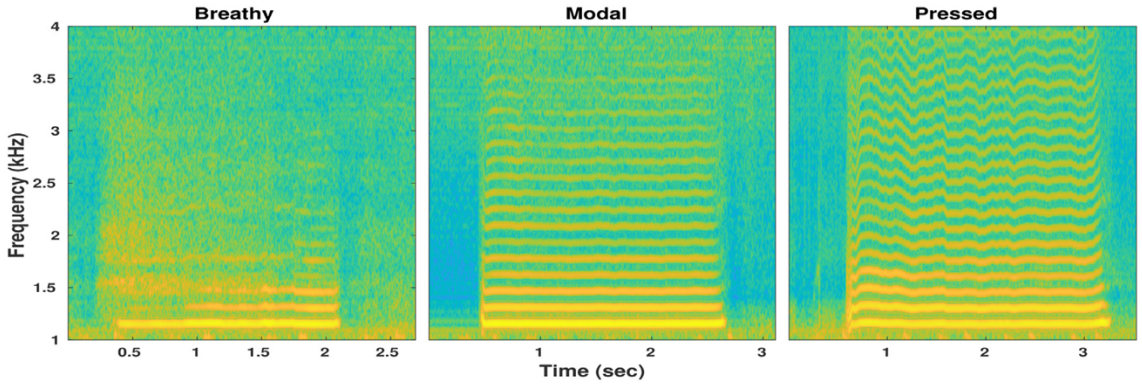


Figura 3: Espectrogramas de flujos glotales estimados correspondientes a las fonaciones soplada (breathy), modal y presionada (pressed), respectivamente. Se observa un aumento progresivo en la energía de los armónicos superiores desde la fonación soplada hacia la presionada, reflejando diferencias en el grado de cierre glotal y la tensión laríngea. Tomado de [12].

Una forma complementaria de analizar esta señal es mediante su derivada temporal, la cual está estrechamente asociada a la señal acústica radiada en los labios [1], aspecto que se abordará con mayor detalle en la siguiente sección. La Figura 4 muestra una representación de un ciclo del flujo glotal junto con su derivada temporal.

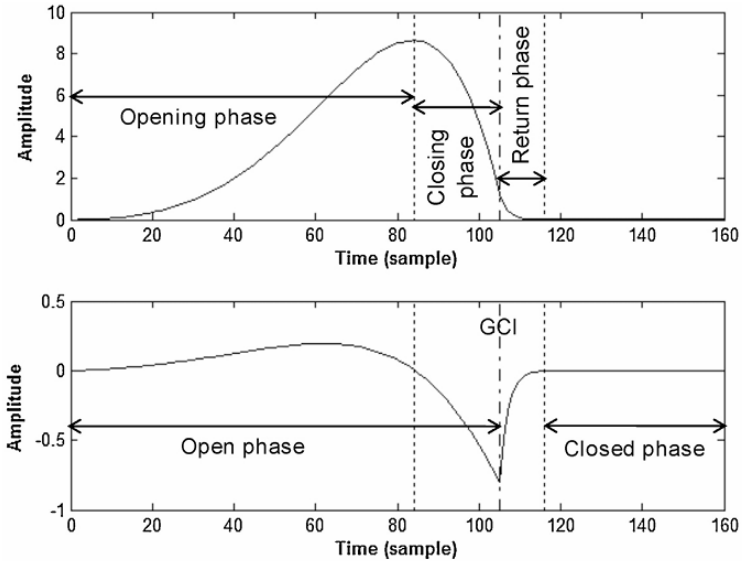


Figura 4: Representación de un ciclo del flujo glotal y su derivada. Tomado de [5].

La derivada del flujo glotal resalta los eventos de apertura y cierre de las cuerdas vocales. En particular, el instante de cierre glotal (GCI por sus siglas en ingles) corresponde al instante en que las cuerdas vocales se cierran bruscamente, generando una excitación similar a un impulso en el sistema del tracto vocal [14]. Este instante permite, por ejemplo, sincronizar el análisis de la señal con los ciclos glotales, estimar parámetros del flujo glotal

o incorporar información de la fuente glotal en aplicaciones como el reconocimiento de hablantes y el análisis expresivo del habla [5].

Con el flujo glotal, su derivada y los GCIs, uno de los pasos fundamentales es la etapa de parametrización, que consiste en describir cuantitativamente la forma del flujo glotal mediante un conjunto de parámetros representativos. Estos incluyen tanto parámetros temporales como espectrales, los cuales capturan distintas propiedades dinámicas y acústicas del flujo. Estos parámetros permiten, por ejemplo, caracterizar el tipo de fonación [12], detectar alteraciones asociadas a trastornos de la voz [4] o evaluar métodos de filtrado inverso [15]. En la Tabla 1, se resumen algunos parámetros glotales utilizados junto con una descripción.

Tabla 1: Parámetros temporales y espectrales del flujo glotal.

Parámetro	**Tipo**	**Descripción**
Open Quotient (OQ)	Temporal	Proporción del ciclo glotal durante el cual la glotis está abierta.
Closing Quotient (ClQ)	Temporal	Duración relativa de la fase de cierre del ciclo glotal.
Speed Quotient (SQ)	Temporal	Relación entre las fases de apertura y cierre del ciclo glotal.
Normalized Amplitude Quotient (NAQ)	Temporal	Relación entre la amplitud peak del flujo glotal y la velocidad de cierre, normalizada por el periodo.
H1–H2	Espectral	Diferencia de amplitud entre el primer y segundo armónico.
Harmonic Richness Factor (HRF)	Espectral	Proporción de energía entre los armónicos superiores y el fundamental.
Spectral Tilt (ST)	Espectral	Pendiente del espectro de magnitud.

El flujo glotal constituye una señal de alto interés por su aplicación en contextos clínicos y tecnológicos, pero acceder directamente a esta señal resulta complejo. La vibración de las cuerdas vocales ocurre en una región dificil de acceder sin técnicas invasivas y equipamiento especializado [1], y el flujo glotal en sí no puede ser medido de forma directa durante el habla natural. Esta limitación ha motivado el desarrollo de métodos

indirectos de estimación, siendo el filtrado inverso glotal (Glottal Inverse Filtering, GIF) una alternativa no invasiva para obtener el flujo glotal.

3.2. Filtrado Inverso Glotal

El filtrado inverso es una técnica que, a partir de una señal adquirida por un sensor (como un micrófono o un acelerómetro), busca estimar la señal original que dio lugar a dicha medición. Para ello, se modela el sistema que modificó la señal original durante su propagación hasta el sensor, y se invierten sus efectos mediante la aplicación del modelo inverso. De esta forma, se obtiene una estimación de la señal original. A continuación, se revisan en detalle los enfoques de filtrado inverso aplicados tanto a señales acústicas captadas por micrófono como a señales de aceleración superficial registradas mediante sensores en el cuello, considerando sus fundamentos, modelamientos y limitaciones particulares.

3.2.1. Filtrado inverso glotal para señales de micrófono

El micrófono es una opción directa y accesible para capturar la señal necesaria para la estimación de la fuente glotal mediante filtrado inverso. En este contexto, el objetivo es estimar la señal de la fuente glotal a partir de la señal acústica capturada por el micrófono. Según la teoría acústica del habla, esta señal puede modelarse como el resultado de aplicar un filtro, que incluye tanto el efecto del tracto vocal como la radiación en la abertura bucal, a una fuente sonora (la señal glotal) [2]. Bajo este enfoque, las técnicas de filtrado inverso buscan eliminar las contribuciones del tracto vocal y de la radiación presentes en la señal registrada por el micrófono, con el fin de obtener una estimación de la fuente glotal.

De esta manera, el proceso de filtrado inverso depende de cómo se modela el tracto vocal y la radiación en la abertura bucal. Por lo tanto, es fundamental comprender su funcionamiento en la producción de voz.

El tracto vocal incluye la laringe, la faringe, las cavidades orales y nasales. En la producción de voz, este sistema actúa como un filtro resonante que “colorea” la fuente sonora (flujo glotal), otorgándole las características acústicas que permiten al ser humano transmitir información. Desde un enfoque simplificado, puede modelarse como un filtro lineal con resonancias, cuyas frecuencias se denominan formantes. Estos formantes correspon-

den a los polos de la función de transferencia del sistema, y sus frecuencias (F1, F2, etc.) dependen de la configuración articulatoria adoptada para producir cada vocal. La posición de la lengua, la mandíbula y los labios modifica las cavidades resonantes del tracto vocal, generando variaciones en los formantes según la vocal emitida. La variación de los formantes se visualiza en diagramas como el de la Figura 5, que representa el espacio de formantes F1-F2 para diferentes vocales del inglés. Además, la frecuencia de los formantes disminuye a medida que aumenta la longitud del tracto vocal, lo que explica por qué, en promedio, los hombres presentan formantes más bajos que las mujeres para las mismas vocales[8].

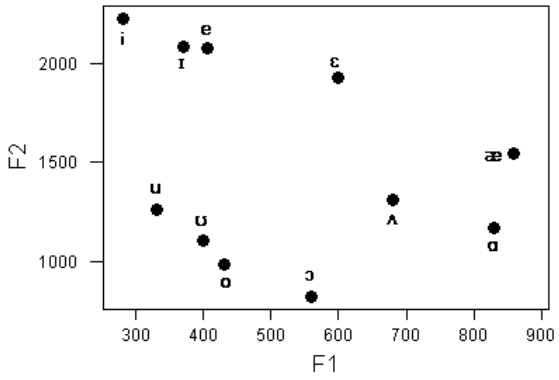


Figura 5: Diagrama de formantes F1-F2 para vocales del inglés. Tomado de [16]

Desde la perspectiva del filtrado inverso, el modelado del tracto vocal se centra en estimar un filtro que elimine las resonancias (formantes) y anchos de bandas del tracto vocal, con el fin de recuperar la fuente glotal original. Una de las técnicas más utilizadas para este propósito es la predicción lineal (LP, por sus siglas en inglés), que modela el tracto vocal como un filtro de solo polos de orden p.

LP modela cada muestra de la señal de voz como una combinación lineal de las p muestras anteriores, ajustando los coeficientes de dicha combinación para minimizar el error de predicción dentro de una ventana temporal corta (típicamente de 20 a 30 ms), en la que se asume estacionariedad [17]. Así, se estima un conjunto de coeficientes que representa de manera óptima las características resonantes del tracto vocal. El parámetro p permite controlar la complejidad del modelo, un valor mayor permite capturar más detalles del espectro, pero incrementa el riesgo de sobreajuste.

Por otra parte, el flujo de aire, modulado por el tracto vocal, se transforma en una señal de presión sonora al salir por la boca hacia el campo libre. Esta transformación, conocida como radiación por la boca, está regida por principios de dinámica de fluidos. En el con-

texto del filtrado inverso glotal, este fenómeno se modela como un diferenciador, ya que se asume que las variaciones de presión son proporcionales a la derivada del flujo de aire [18]. Como consecuencia de esta operación, se pierde la componente DC del flujo glotal al propagarse hacia el campo libre [19]. Para compensar el efecto de la radiación durante el filtrado inverso, es necesario integrar la señal registrada.

A continuación, se describen algunos de los métodos clásicos utilizados en el estado del arte para realizar filtrado inverso en señales de micrófono, seguidos de una evaluación comparativa basada en dos artículos relevantes.

3.2.2. Métodos de filtrado inverso para señales de micrófono

- **Closed phase (CP) analysis [20, 1]**:Calcula un modelo del tracto vocal mediante LP sobre las muestras correspondientes a la fase cerrada del ciclo glotal (flujo nulo). Esta fase se considera óptima para la estimación del filtro vocal, ya que la influencia del flujo es mínima en la señal de voz. Sin embargo, la detección precisa de dicha fase puede verse comprometida cuando su duración es muy breve y no se cuenta con un número suficiente de muestras, como ocurre en fonaciones con frecuencia fundamental elevada o en fonaciones de tipo soplado, donde la fase cerrada puede ser inexistente.

- **Iterative adaptative inverse filtering(IAIF) [21]**: Este método estima el flujo glotal mediante un proceso de dos iteraciones. En la primera, se modela el efecto del flujo glotal en la señal de voz utilizando LP de primer orden. Esta estimación se elimina de la señal mediante filtrado inverso. Luego, sobre la señal ya filtrada, se aplica un LP de orden superior para obtener un primer modelo del tracto vocal. Con este modelo, se obtiene un primer estimado del flujo glotal al cancelar los efectos del tracto vocal y la radiación labial de la señal original. Este estimado inicial sirve de base para la segunda iteración, en la cual se vuelve a modelar el flujo glotal usando un LP de orden dos o cuatro. Nuevamente, se elimina su contribución por filtrado inverso, y se estima otra vez el tracto vocal con un modelo de orden mayor. Finalmente, se aplica filtrado inverso sobre la señal original para cancelar la radiación labial y el tracto vocal, obteniendo así el flujo glotal estimado.

- **Quasi-Closed Phase (QCP) analysis [22]:** Este método, basado en CP, utiliza el Weighted Linear Prediction (WLP) para modelar el tracto vocal. El uso de WLP

permite enfatizar determinadas contribuciones de la señal, lo que facilita resaltar las muestras que corresponden a la fase cerrada del ciclo glotal, donde la señal de voz está dominada por el tracto vocal. Para lograrlo, aplica una función de ponderación llamada Attenuated Main Excitation (AME), que construye una forma de onda temporal diseñada para reducir la contribución de las muestras de voz ubicadas en la vecindad de la excitación principal del tracto vocal, es decir, en torno a los GCIs.

QCP estima directamente el flujo glotal mediante el filtrado inverso de la señal de voz, utilizando un modelo del tracto vocal de alto orden obtenido a través de WLP y cancelando el efecto de la radiación labial. El método requiere conocer los instantes de cierre glotal y definir tres parámetros que controlan la función AME:

- Position Quotient (PQ): determina la posición relativa donde comienza la sección no atenuada.
- Duration Quotient (DQ): controla la duración relativa de la sección no atenuada.
- Ramp Quotient (RQ): define la duración relativa de la rampa de transición que conecta las zonas atenuada y no atenuada en cada ciclo de voz.

En la Figura 6 se puede observar una representación de la derivada del flujo glotal y la función AME entre dos GCIs consecutivos.

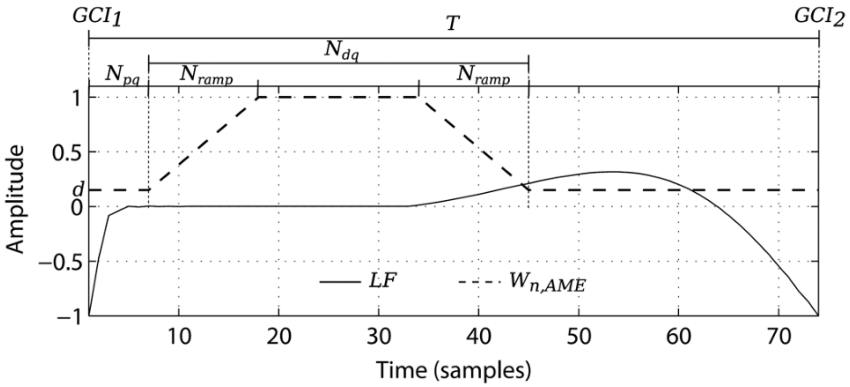


Figura 6: Representación de la derivada del flujo glotal (LF) y de la función AME ($W_{n,AME}$) entre dos instantes de cierre glotal consecutivos (GCIs). Tomado de [22].

- **Complex Cepstrum-based Decomposition (CCD) [23, 15]**: Este método aplica una descomposición en cepstrum complejo a segmentos de señal de voz con el fin de separar sus componentes de mínima y máxima fase, asociadas respectivamente al tracto vocal (respuesta causal) y al flujo glotal durante la fase abierta (respuesta

anticausal). El procedimiento consiste en aplicar la transformada discreta de Fourier a un segmento de señal de voz centrado en el GCI, seguida del logaritmo complejo y su transformada inversa. Luego, se extrae únicamente la parte anticausal del cepstrum correspondiente al flujo glotal durante la fase abierta. Finalmente, se aplica la exponencial compleja e inversión de Fourier para recuperar la estimación temporal del flujo glotal.

Los métodos de filtrado inverso presentan limitaciones que deben tenerse en cuenta al estimar el flujo glotal a partir de señales de micrófono, especialmente bajo condiciones que afectan la estructura del ciclo glotal o la calidad de la señal. Debido a la dificultad de acceder experimentalmente al flujo glotal real, la validación de estos métodos se realiza mediante simuladores de voz que permiten generar señales sintéticas con referencias controladas. A continuación, se describen dos evaluaciones relevantes reportadas en la literatura, que permitirán justificar la selección del método utilizado en este trabajo para obtener las referencias de flujo glotal. Cabe mencionar que estos estudios analizan algunas variantes de los métodos mencionados, dichas variantes no se describen ya que no forman parte del desarrollo principal de este trabajo.

Uno de estos estudios es el realizado por Chien et al. (2017) [15], quienes evaluaron los métodos CP, IAIF, CCD, Sparse Linear Prediction (SLP) y Weighted Linear Prediction (WLP), siendo estos dos últimos variantes del enfoque QCP. Para ello, generaron un conjunto de 750 vocales sostenidas (correspondientes a /i/, /e/, /ɛ/, /ä/, /o/ y /u/) y 125 oraciones sintéticas, estas últimas derivadas de una única oración en alemán (Lea und Doreen mögen Bananen) utilizando VocalTractLab, un sintetizador articulatorio que implementa un modelo tridimensional del tracto vocal y modelos biomecánicos de la laringe para generar señales de voz a partir de configuraciones articulatorias [24].

Las señales se sintetizaron variando la frecuencia fundamental, la presión subglótica y la calidad vocal, de modo de evaluar el desempeño de los métodos bajo distintas condiciones fisiológicas y fonatorias. La detección de los GCIs (y de los instantes de apertura glotal, necesarios en CP para marcar el fin de la fase de cierre) se realizó mediante el algoritmo YAGA para los métodos CP, SLP y WLP, mientras que el método CCD utilizó el algoritmo de SEDREAMS. Ambos algoritmos permiten estimar estos instantes a partir de la señal de voz. IAIF, por su parte, no requiere detección de GCIs. Además, los autores evaluaron la robustez de los métodos frente a errores en la detección de los GCIs.

Las formas de onda estimadas fueron alineadas en el tiempo y normalizadas en amplitud respecto a la referencia, y se calculó el Mean Absolute Error-Waveform (MAE-WAVE) el cual mide error considerando las diferencias entre cada ciclo de las señales. Además, se emplearon métricas basadas en parámetros glotales como NAQ, H1-H2 y HRF. Las conclusiones más relevantes para el presente trabajo son:

- Los algoritmos evaluados presentan errores MAE-WAVE del orden de 30 % en vocales sostenidas y 40 % en habla continua. El método CP demostró el mejor desempeño global en vocales sostenidas, mientras que SLP fue el más efectivo para el habla continua.
- El desempeño de los algoritmos se ve afectado en las vocales cuyo primer formante presenta una frecuencia baja. Los autores explican que esta formante baja se superpone con la energía de baja frecuencia del flujo glotal, lo que dificulta el proceso de filtrado inverso. En particular, las vocales cerradas y redondeadas (/o/, /u/) registran errores superiores a 0.4 en MAE-WAVE en todos los casos evaluados.
- Todos los métodos mostraron un rendimiento aproximadamente constante frente a variaciones en la calidad vocal y la presión subglótica, con la excepción del CCD, cuyo desempeño se deterioró notablemente en condiciones de voz susurrante o de baja presión subglótica. Los autores explican que esto se debe a que, bajo dichas condiciones, las suposiciones en las que se basa el CCD dejan de ser válidas.
- Los métodos muestran una degradación en el desempeño a medida que aumenta la frecuencia fundamental. Los autores explican que esto ocurre porque los armónicos se vuelven más dispersos en el espectro, lo que dificulta la estimación del contorno espectral.
- SLP presenta la menor dependencia del error introducido en los GCIs, mientras que CP, WLP y CCD muestran aumentos pronunciados del MAE-WAVE a medida que el error aumenta. Cabe notar que IAIF no depende de los CGIs.

Más recientemente, Freixes et al. (2023) [25] evaluaron los métodos IAIF, IOP-IAIF, GFM-IAIF, QCP y ST-QCP. Para ello, utilizaron el primer repositorio del conjunto de datos OPENGLOT. OPENGLOT, desarrollado en 2019 por Alku et al. [26], es una plataforma de acceso abierto diseñada para la evaluación de algoritmos de filtrado inverso

aplicados a señales de voz. Esta plataforma se compone de cuatro repositorios que integran tanto datos sintéticos generados mediante modelos computacionales como grabaciones reales. El primer repositorio contiene vocales sintéticas generadas utilizando el modelo computacional de flujo glotal de Liljencrants-Fant (LF). Estas vocales fueron filtradas mediante tractos vocales modelados como filtros de solo polos de octavo orden, configurados con cuatro formantes. El conjunto incluye seis vocales (/a/, /e/, /i/, /o/, /u/, /æ/), cuatro tipos de fonación (whispery, breathy, normal y creaky). También se consideró un rango progresivo de frecuencias fundamentales. Para representar ambos géneros, los autores utilizaron configuraciones específicas de frecuencias formantes correspondientes a voces masculinas y femeninas. En total, se generaron 672 muestras.

Para evaluar los métodos, los autores propusieron una metodología de análisis en dos etapas. En la primera, se realizó una optimización de parámetros para cada algoritmo con el objetivo de minimizar el MAE-Wave, métrica utilizada en la evaluación previamente presentada. En la segunda etapa, utilizando los parámetros óptimos, se calcularon distintas métricas de error sobre las señales de flujo glotal estimadas. Estas incluyeron la distancia RMS relativa entre la señal glotal estimada normalizada y la señal de referencia para cada período, determinada a partir de los GCIs, así como cuatro métricas adicionales basadas en parámetros glotales: NAQ, H1-H2, HRF y ST. Las conclusiones más relevantes para este trabajo son:

- Los métodos basados en QCP superaron significativamente a las variantes de IAIF en la mayoría de las métricas y condiciones analizadas.
- Al igual que en el trabajo anterior, el desempeño de todos los métodos disminuyó con el aumento de la frecuencia fundamental. Esta degradación fue más pronunciada en voces femeninas. Drugman et al. [27] señalan la dificultad adicional de aplicar el filtrado inverso en este tipo de voces y explican que, debido a la menor duración de los ciclos glotales, el tracto vocal no tiene tiempo suficiente para volver a su estado inicial entre excitaciones consecutivas. A esto se le suman las frecuencias formantes más altas, resultado de un tracto vocal más corto, como se explicó en la sección 3.2.1. A pesar de esto, los métodos QCP demostraron un rendimiento más estable en el rango de frecuencias fundamentales estudiado.
- Las vocales /i/, /e/ y /æ/ (en voces femeninas) y /i/ y /u/ (en voces masculinas) presentaron los mayores errores, especialmente en los métodos IAIF. Al igual que

en el trabajo anterior, los autores atribuyen este comportamiento a la cercanía del primer formante a la región espectral del flujo glotal, dificultando la separación fuente-filtro.

- Aunque las variantes IOP-IAIF y GFM-IAIF mejoraron en algunos aspectos al IAIF original, siguieron siendo superadas por los métodos QCP. Cabe destacar que, en el corpus femenino, QCP supera a QCP-ST incluso en el error ST, mientras que en el corpus masculino ocurre lo contrario, donde QCP-ST obtiene mejores resultados en algunas métricas específicas.

Los estudios presentados resultan relevantes para este trabajo no solo porque permiten comparar el desempeño de distintos métodos de filtrado inverso bajo diversas condiciones, sino también por las metodologías empleadas, las cuales aportan elementos clave sobre cómo implementar y ajustar los algoritmos. En particular, es importante destacar que el segundo estudio optimizó los parámetros de los métodos evaluados, lo cual naturalmente conduce a mejores resultados. Sin embargo, este tipo de optimización no es posible en este trabajo, ya que no hay una referencia verdadera del flujo glotal. En contraste, el primer estudio utilizó parámetros predefinidos, un enfoque que se alinea con lo que deberá adoptarse aquí.

Debido a las limitaciones de los métodos de filtrado inverso, en los últimos años se han explorado nuevas técnicas para estimar el flujo glotal a partir de señales de voz. Si bien estas propuestas no se analizarán en detalle, por no constituir el enfoque central de este trabajo, es relevante mencionar que recientemente se han desarrollado métodos basados en redes neuronales profundas para este propósito [28]. Estas estrategias serán retomadas en la sección 3.3, dado que guardan una estrecha relación con la metodología empleada en el presente estudio.

Por otro lado, el uso de señales de acelerómetro ha surgido como una alternativa viable para la estimación del flujo glotal, especialmente en contextos donde las señales acústicas captadas por micrófonos se ven afectadas por el ruido ambiental o presentan limitaciones prácticas para su obtención. A continuación, se describe en detalle el enfoque IBIF, uno de los métodos más relevantes para aplicar filtrado inverso glotal sobre señales de acelerómetro, con énfasis en sus fundamentos físicos y en los principios que permiten modelar la relación entre la señal registrada en el cuello y el flujo glotal. . Se incluyen además algunas extensiones y mejoras propuestas sobre este enfoque. Posteriormente, se presentan

otras estrategias que, si bien fueron desarrolladas originalmente para señales acústicas, han sido utilizadas también con señales de acelerómetro.

3.2.3. Filtrado inverso glotal para señales de acelerómetro

Esta modalidad presenta varias ventajas. Por un lado, las señales de acelerómetro son menos sensibles al ruido ambiental, lo que las vuelve especialmente útiles en entornos no controlados o aplicaciones ambulatorias. Además, al no contener información inteligible de la voz, estas señales garantizan un mayor nivel de privacidad para el hablante. Finalmente, su facilidad de implementación en dispositivos portátiles facilita el monitoreo continuo de la función vocal en condiciones naturales del habla [6].

A diferencia del filtrado inverso para señales de micrófono, cuyo objetivo es cancelar las resonancias del tracto vocal, este enfoque se centra en eliminar las resonancias subglotales captadas en la superficie del cuello. Desde un punto de vista fisiológico, la aceleración de la superficie del cuello está acoplada mecanoacústicamente con la presión subglótica y el flujo de aire glotal [29]. El sistema subglotal, compuesto por las vías respiratorias del árbol traqueobronquial y los tejidos circundantes, suministra el flujo de aire para la fonación y se comporta como un filtro acústico con sus propias frecuencias naturales. Estas frecuencias, conocidas como resonancias subglotales, interactúan con la fuente glotal y el tracto vocal superior. A diferencia de los formantes, que varían significativamente con la configuración articulatoria para producir distintas vocales, las resonancias subglotales son más estables y están fuertemente ligadas a las características fisiológicas del hablante, principalmente a la altura, ya que esta se relaciona con la longitud del árbol traqueobronquial [30].

Uno de los métodos de filtrado inverso más relevantes para el sistema subglótico es el filtrado inverso basado en impedancia subglotal (IBIF, por sus siglas en inglés) [6]. Este método propone una representación computacional del sistema subglótico. La Figura 7 presenta un esquema del sistema subglótico junto con el modelo eléctrico utilizado en IBIF.

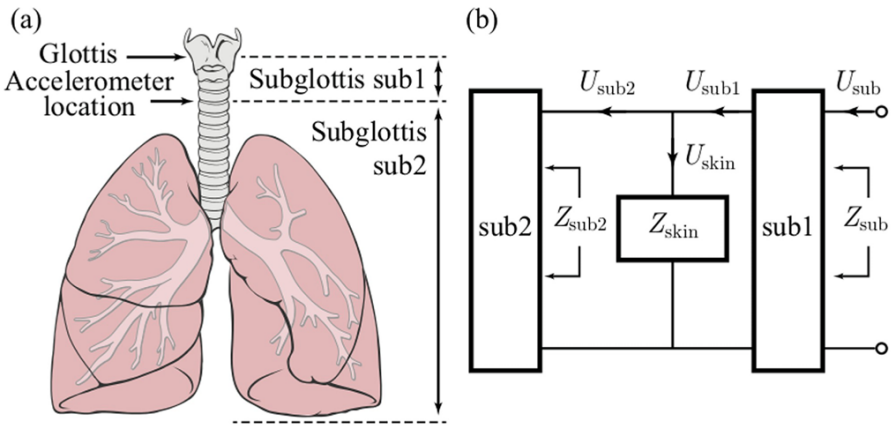


Figura 7: (a) Representación del sistema subglotal. El acelerómetro se posiciona sobre la piel, en la región de la escotadura supraesternal, aproximadamente a 5 cm por debajo de la glotis. Esta ubicación divide el tracto subglotal en dos segmentos que representan la porción de la tráquea extratorácica situada por encima (sub1) y por debajo del acelerómetro (sub2). (b) Modelo eléctrico del sistema subglotal. Tomado de [6].

El método IBIF utiliza un modelo de líneas de transmisión para obtener la representación de las impedancias en frecuencia de las secciones sub1 y sub2 que se observan en la Figura 7 (a). Este modelo divide la tráquea en pequeñas secciones que relacionan la presión subglótica y la velocidad del flujo volumétrico de aire mediante elementos acústicos concentrados. De este modo, las secciones sub1 y sub2 se modelan como una cadena de estas secciones. A esta red se incorpora en paralelo una impedancia adicional (Z_{skin}) que representa las propiedades mecánicas de la piel y la carga del acelerómetro. Por esta impedancia pasa U_{skin}, que denota la velocidad de la piel. Una vez que se definen las impedancias equivalentes de cada sección, como se puede ver en la Figura 7 (b), se calcula la respuesta en frecuencia del sistema desde la glotis hasta la ubicación del sensor. Dado que la entrada capturada es la aceleración y no la velocidad, se incluye un diferenciador ideal que permite obtener un modelo que relaciona la aceleración registrada en la piel ($\dot{U}_{skin}$) y el flujo de aire subglotal. Finalmente, el flujo glotal se define como el negativo del flujo subglotal.

El procedimiento de filtrado inverso consiste en tomar ventanas de la señal de aceleración registrada en el cuello, transformarlas al dominio de la frecuencia mediante FFT y multiplicarlas por la inversión de la función de transferencia obtenida por el modelo.

Finalmente, dado que el modelo depende de parámetros anatómicos y mecánicos que varían entre individuos, se implementa una etapa de calibración específica para cada sujeto. Esta consiste en ajustar los parámetros del modelo (resistencia, masa y rigidez de la piel,

longitud traqueal y posición del sensor) mediante un proceso de optimización que minimiza el error entre el flujo estimado por IBIF y una referencia obtenida por métodos tradicionales, como el filtrado inverso de flujo oral.

Por otro lado, la investigación reciente ha comenzado a explorar enfoques complementarios basados en el aprendizaje automático, los cuales pueden aprender relaciones complejas directamente a partir de los datos sin depender de un modelo físico explícito. Por ejemplo, en un problema distinto pero relacionado, Sepúlveda et al. [31] utilizaron una red bayesiana para estimar parámetros fisiológicos internos, como la presión subglótica, a partir de características extraídas de la señal de aceleración.

Los modelos físicos a menudo requieren simplificaciones que no logran capturar las no linealidades y la compleja interacción entre múltiples componentes del sistema fonador. Además, estos modelos pueden ver su rendimiento degradado cuando sus suposiciones de base no se cumplen, dependen de calibraciones sujetas a error o pueden requerir la elección de parámetros subjetivos. Es por esta razón que el desarrollo de un modelo basado en datos, como los que ofrece el aprendizaje profundo, emerge como una alternativa viable para la tarea de filtrado inverso y la estimación del flujo glotal. A continuación, se detallan los principios del aprendizaje profundo, se justifica su aplicabilidad en el sistema de producción de voz, y se presentan algunas arquitecturas de red neuronal que pueden ser ideales para este trabajo.

3.3. Aprendizaje profundo

El aprendizaje profundo es una rama del aprendizaje automático que utiliza redes neuronales profundas para tomar decisiones imitando el funcionamiento de las neuronas biológicas. Estas redes se componen de capas de neuronas artificiales interconectadas, cada una con un peso y un umbral de activación [32]. En la Figura 8 se presenta un esquema de una red neuronal.

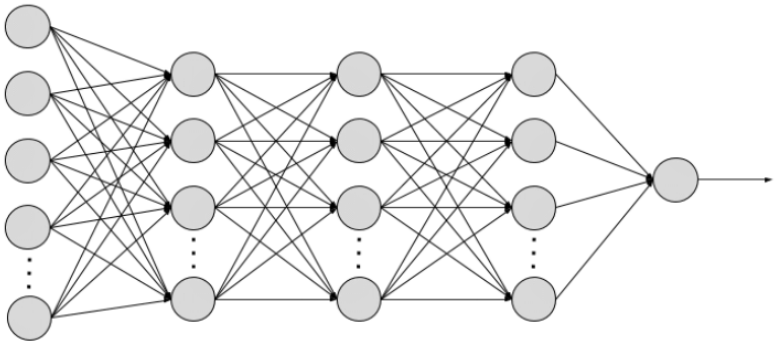


Figura 8: Modificado de [33].

Las redes neuronales se organizan en tres tipos de capas: una capa de entrada, una o más capas ocultas y una capa de salida. La información ingresa a través de la capa de entrada, donde los datos se procesan y analizan. Estos datos luego se pasan a las capas ocultas, donde se realizan cálculos complejos. Finalmente, la capa de salida proporciona el resultado final de este procesamiento[33]. Las conexiones entre los nodos de una red neuronal se representan con un peso, que indica la influencia que tiene un nodo sobre otro. Si el resultado de un nodo supera un umbral de activación, se activa y envía los datos a la siguiente capa [32].

El proceso de aprendizaje de las redes neuronales, llamado entrenamiento, se basa en principios de optimización. Su objetivo es ajustar los pesos de la red para minimizar la función de pérdida, que cuantifica la diferencia entre las predicciones del modelo y los valores reales en un conjunto de datos. Este ajuste se realiza mediante un bucle de retroalimentación, cuyo algoritmo más común es la retropropagación del error. En este proceso, el error calculado en la capa de salida se propaga hacia atrás a través de la red, y los pesos de cada conexión se actualizan en proporción a su contribución al error total, reduciendo así las predicciones incorrectas en iteraciones futuras [33, 34].

El éxito del entrenamiento de una red neuronal depende fundamentalmente de la calidad y cantidad de los datos disponibles. En el enfoque de aprendizaje supervisado, la red aprende a mapear una entrada a una salida a partir de un gran número de ejemplos etiquetados. Por ello, un conjunto de datos extenso y representativo del problema es crucial para que el modelo pueda generalizar su conocimiento a casos no vistos durante el entrenamiento [7].

En este sentido, las redes neuronales han demostrado ser una herramienta especialmente eficaz para trabajar con señales fisiológicas. Su principal ventaja es que pueden transfor-

mar los datos de entrada en representaciones más abstractas combinando módulos simples pero no lineales. Esto les permite aprender patrones y modelar funciones complejas directamente de los datos, sin necesidad de una extracción manual de características [7].

Cabe mencionar que esta capacidad de aprendizaje conlleva la dificultad de interpretar su funcionamiento interno. La compleja red de cálculos que ocurre en las capas ocultas hace muy difícil explicar cómo el modelo llega a una determinada salida. Por esta razón, a menudo se describe a estos modelos como una caja negra [35].

3.3.1. Redes neuronales utilizadas en el sistema de voz

Aplicar este enfoque al filtrado inverso glotal presenta un obstáculo importante. La imposibilidad de medir directamente el flujo glotal dificulta la obtención de datos para el entrenamiento supervisado. A pesar de esta dificultad, se han podido utilizar las redes neuronales con éxito para estimar la fuente glotal o parámetros relacionados. La falta de datos de referencia ha impulsado diversas estrategias para generar datos de entrenamiento.

Narendra et al. [36], por ejemplo, utilizaron redes neuronales profundas para estimar la fuente glotal a partir de señales de voz degradadas por códecs telefónicos. Para generar los datos de entrenamiento, utilizaron una base de datos de voz de alta calidad, de la cual derivaron dos conjuntos de señales. Por un lado, estimaron un flujo glotal de referencia aplicando QCP sobre las señales limpias. Por otro, procesaron esas mismas señales con un códec de compresión para obtener una versión codificada, de la cual extrajeron características acústicas. Finalmente, entrenaron el modelo para mapear las características de la señal codificada al flujo glotal de referencia. Los autores demostraron que su sistema era capaz de estimar el flujo glotal desde la voz codificada con mayor precisión que los métodos IAIF y CP.

Uno de los principales desafíos al trabajar con redes neuronales en esta área es la falta de acceso al flujo glotal real. En el caso anterior, se utilizó QCP para obtener una referencia. Por su parte, Langheinrich et al. [28], propusieron entrenar una red bidireccional de memoria a corto y largo plazo (BiLSTM) exclusivamente con datos sintéticos. Los pares micrófono-flujo glotal fueron generados a partir de un sintetizador articulatorio. El modelo fue capaz de generalizar este conocimiento a señales de voz naturales, logrando estimar el flujo glotal de manera comparable a IAIF.

Finalmente, en un trabajo que también utiliza la señal de acelerometría del cuello, Sepúlveda et al. [31] emplearon una red neuronal para estimar parámetros fisiológicos como la presión subglótica. El entrenamiento se basó principalmente en datos sintéticos generados por un modelo físico de los pliegues vocales. Su red fue entrenada para mapear un conjunto de siete características aerodinámicas y acústicas a los parámetros fisiológicos de interés. Posteriormente, para alinear el conocimiento del modelo con mediciones reales, aplicaron una etapa de "Transfer Learning", que consistió en reentrenar la red con un conjunto de datos clínicos. El enfoque de red bayesiana les permitió, además, realizar las estimaciones y cuantificar la incertidumbre asociada a ellas.

Estos trabajos revelan la viabilidad práctica de realizar un filtrado inverso glotal a partir de la señal de acelerómetro. Tanto la señal de aceleración como el flujo glotal son series temporales, donde el valor en un instante depende de valores pasados. Para modelar la relación entre ellas, como se requiere en el filtrado inverso, la arquitectura de la red neuronal debe ser capaz de procesar información secuencial y capturar estas dependencias temporales [37]. La arquitectura utilizada por Langheinrich es un ejemplo de este tipo de red, aunque existen otras alternativas que podrían aplicarse a esta tarea. Para implementar un filtrado inverso de este tipo, es necesario conocer las distintas arquitecturas disponibles.

3.3.2. Arquitecturas para el Procesamiento de Secuencias

Existen diversas familias de arquitecturas de redes neuronales diseñadas para el procesamiento de datos secuenciales. Cada una aborda el desafío de capturar dependencias temporales desde un enfoque distinto, ofreciendo un compromiso diferente entre eficiencia computacional, capacidad de modelado y complejidad. A continuación se presentan tres arquitecturas que pueden ser utilizadas para el problema de filtrado inverso:

- **Long Short-Term Memory (LSTM) y BiLSTM**: Las LSTM son un tipo de red neuronal recurrente (RNN) capaz de aprender dependencias a largo plazo mediante un mecanismo de "puertas"que controlan el flujo de información. Suelen procesar los datos en una sola dirección (hacia adelante). Sin embargo, una variante más potente es la **LSTM Bidireccional (BiLSTM)**, que utiliza una segunda capa LSTM para procesar la secuencia en orden inverso. De este modo, las predicciones de la red en cualquier punto temporal se benefician del contexto tanto del pasado (secuencia hacia adelante) como del futuro (secuencia hacia atrás), mejorando significativa-

mente su capacidad de comprensión contextual.[38]

- **Temporal Convolutional Networks (TCN)**: Las TCN usan convoluciones 1D como base, pero adaptadas para procesamiento temporal. Emplean convoluciones causales, asegurando que la salida en un tiempo t solo dependa de entradas en t o antes (nunca del futuro). Además, usan convoluciones dilatadas, que aplican el filtro con saltos, lo que permite un campo receptivo amplio sin aumentar mucho la profundidad. Esto les permite capturar dependencias en el largo plazo eficientemente y entrenarse más rápido que las RNN tradicionales gracias al paralelismo de convoluciones [39].

- **Transformers**: Los Transformers procesan toda la secuencia simultáneamente en lugar de paso a paso. Se basan en el mecanismo de auto-atención (o atención), por el cual cada elemento de la secuencia pondera la importancia de todos los demás al construir su representación. Esto permite capturar dependencias complejas a cualquier distancia. Como no dependen de recursividad, pueden entrenarse en paralelo y suelen ser más rápidos que LSTM. Para retener información sobre la posición de cada elemento, los Transformers incorporan codificación posicional [40].

4. Metodología

El marco teórico presentado establece los conceptos necesarios y herramientas disponibles para abordar el problema. Sobre esta base, se presenta a continuación la metodología diseñada.

4.1. Descripción General del Sistema Propuesto

El objetivo de esta memoria es desarrollar un método de estimación del flujo glotal a partir de señales de acelerómetro captadas en el cuello, empleando un modelo de aprendizaje profundo entrenado con referencias generadas mediante filtrado inverso aplicado a señales de micrófono.

El proceso se divide en las siguientes etapas:

1. Selección y procesamiento de señales de micrófono (MIC) y acelerómetro (ACC) provenientes de una base de datos existente.
2. Estimación y procesamiento del flujo glotal de referencia.
3. Entrenamiento y evaluación de redes neuronales con las señales de acelerómetro y flujo glotal de referencia.

4.2. Datos Disponibles

Para este trabajo se tienen señales provenientes de la base de datos del Hospital General de Massachusetts (MGH, por sus siglas en ingles), especificamente aquellas correspondientes a las grabaciones en laboratorio descritas en [4]. Esta base de datos incluye señales de hombres y mujeres, con voz normal o con hiperfunción vocal fonotraumática (nódulos o pólipos) y no fonotraumática (disfonía por tensión muscular), registrados en diferentes etapas de su tratamiento, que pueden incluir terapia vocal y/o cirugía. La Figura 9 presenta un sujeto con los dispositivos utilizados para la adquisición de datos.

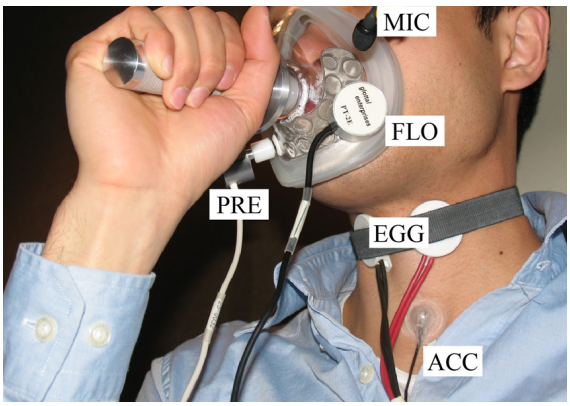


Figura 9: Sujeto con los dispositivos de medición empleados para la adquisición de datos en el laboratorio. Se muestran: micrófono acústico (MIC), electrodos de electroglotografía (EGG), acelerómetro (ACC), máscara neumotacográfica (FLO) y sensor de presión intraoral (PRE), todos sincronizados para registrar de forma simultánea las señales relevantes. Tomado de [4].

Durante las sesiones de grabación en laboratorio, se utilizaron los siguientes dispositivos para la adquisición simultánea de las señales, todas registradas con una frecuencia de muestreo de 20 kHz:

- Un micrófono acústico ubicado a aproximadamente 10 cm de los labios para la captura de la señal de voz radiada.
- Electrodos de electroglotografía colocados a ambos lados del cartílago tiroides para medir la impedancia laríngea variable en el tiempo.
- Un acelerómetro posicionado sobre la superficie del cuello para registrar vibraciones asociadas a la fonación.
- Un sensor de flujo de aire conectado a una mascarilla neumotacográfica para medir el flujo de aire espirado.
- Un sensor de presión conectado a un tubo delgado insertado en la cavidad oral para medir la presión intraoral.

En cada sesión de grabación, los participantes produjeron los siguientes enunciados, con distintas intensidades (suave, media y fuerte) y otros patrones de producción:

- Vocales sostenidas /a/, /e/, /i/, /o/, /u/.

- Secuencias silábicas como /pae pae pae pae pae/, así como otras variantes como /afa/ y /ufu/.
- El *Rainbow Passage*, un texto estándar utilizado en la evaluación clínica de la voz, junto con otras frases y pasajes empleados para la evaluación vocal.

Para la etapa de entrenamiento se utiliza únicamente la vocal /a/ en niveles de intensidad media e intensa, proveniente tanto de sujetos patológicos como no patológicos, de ambos sexos. Esta elección se basa en que el flujo glotal, obtenido mediante filtrado inverso a partir de señales de micrófono, resulta más confiable para esta vocal debido a su primer formante relativamente alto, tal como se presentó en la sección 3.2.2 sobre la evaluación de los métodos de filtrado inverso.

Sin embargo, es importante considerar que esta decisión puede limitar la capacidad de generalización del modelo a otras vocales. Por ejemplo, el enfoque IBIF de Zañartu [6] reconoce que hay una interacción acústica entre la fuente glotal y el tracto vocal que varía con la vocal producida. Por lo tanto, al entrenar el modelo únicamente con la vocal /a/, se corre el riesgo de que el sistema se sobreajuste a esta interacción específica.

Se incluyen sujetos de ambos sexos, así como individuos con y sin patologías, para favorecer la capacidad de generalización del modelo. Las diferencias fisiológicas entre hombres y mujeres, junto con las alteraciones en el flujo glotal asociadas a la presencia de patologías, impactan directamente en su dinámica. Entrenar la red con estas variaciones permite que el modelo aprenda las características comunes del flujo glotal, así como sus desviaciones provocadas por factores anatómicos o patológicos, lo que le confiere una mayor robustez frente a casos no observados durante el entrenamiento.

Se excluyen del conjunto de datos las grabaciones de intensidad suave, ya que presentan muy poca energía en las frecuencias altas, lo que impide que el método de filtrado inverso sea confiable. En la Figura 10 se muestra un ejemplo de este tipo de señal junto con su densidad espectral de potencia.

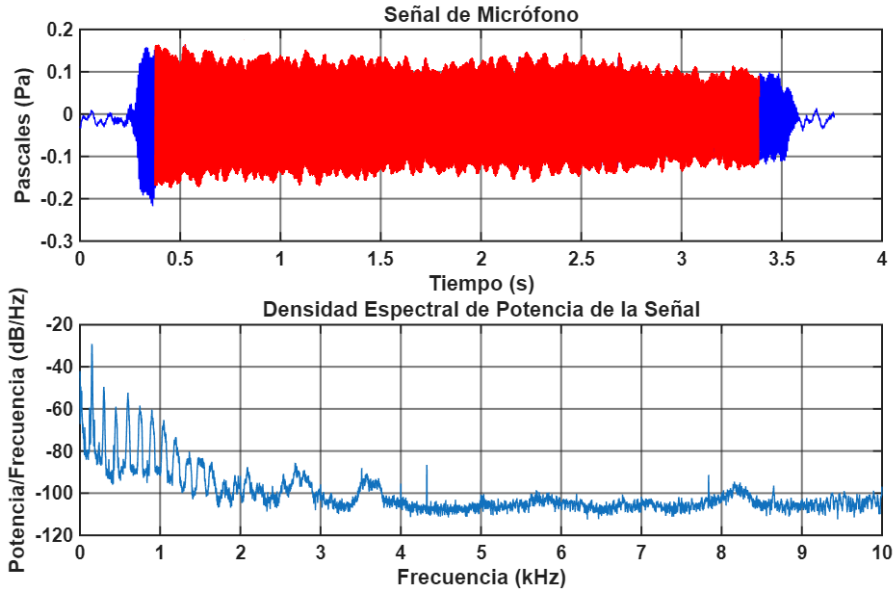


Figura 10: Señal de micrófono de una fonación suave (arriba) y su Densidad Espectral de Potencia (abajo), calculada sobre el segmento en rojo. Se observa que la energía por encima de 2 kHz decae a niveles muy cercanos al piso de ruido.

Se observa que la mayor parte de la energía de la señal se concentra en las frecuencias bajas, por debajo de los 2 kHz aproximadamente. Para frecuencias más altas, la potencia decae a niveles muy cercanos al ruido de fondo. Esta falta de energía en la zona alta del espectro impide identificar correctamente las resonancias del tracto vocal, lo que resulta en una estimación del filtro inverso poco fiable. Los detalles de la implementación de la densidad espectral de potencia se presentan en la sección 5.1.1.

4.3. Obtención del Flujo Glotal de Referencia

Para la obtención del flujo glotal de referencia en este trabajo, se ha seleccionado el método QCP. Esta elección se fundamenta en su robustez y rendimiento superior en condiciones desafiantes.

Tal como se presentó en la sección 3.2.2 sobre la evaluación de los métodos de filtrado inverso, Chien et al. [15] señalan que los algoritmos evaluados presentan errores del orden del 30 % en vocales sostenidas, atribuidos principalmente a voces con frecuencias fundamentales elevadas, algo que ocurre normalmente en voces de mujeres. Por su parte, el estudio de Freixes et al. [25], también incluido en la evaluación, destacó un rendimiento más estable del método QCP en contextos de frecuencias altas. Además, QCP ha sido utilizado exitosamente en la clasificación de sujetos patológicos a partir de las caracterís-

ticas del flujo glotal estimado mediante este método [3]. Finalmente, los flujos glotales estimados mediante QCP se han utilizado para entrenar redes neuronales profundas, como se presentó en la sección 3.3.1. Considerando lo anterior, QCP representa una de las alternativas más robustas disponibles para generar las referencias del modelo.

Para utilizar el método QCP se requiere contar con los GCIs. Existen algoritmos que permiten estimar los GCIs directamente desde la señal de voz, tales como SEDREAMS, YAGA y otros. La estimación de GCIs es un área de estudio en sí misma y, como se describió en la sección 3.2.2, la precisión de estos instantes es crucial para el rendimiento del filtrado inverso. En una revisión cuantitativa, se reportó que SEDREAMS y YAGA destacan en habla limpia, y que SEDREAMS muestra además buena robustez frente al ruido y la reverberación [41].

En este trabajo se detectan los GCIs con SEDREAMS a partir de la señal de micrófono. SEDREAMS calcula una señal "basada en la media" a partir de la señal de voz para determinar intervalos de tiempo aproximados donde se espera que ocurran los eventos de cierre glotal. Luego la ubicación precisa de los instantes se refina identificando la posición del peak más fuerte en la señal de residuo de la predicción lineal [42].

Finalmente, para facilitar la detección de GCIs, obtener una estimación fiable del flujo y mejorar el entrenamiento, todas las señales (micrófono, flujo glotal y acelerometría) se someten a un preprocesamiento. Los detalles de esta etapa se describen en la sección 5.

4.4. Modelo para el Filtrado Inverso Glotal

4.4.1. Arquitectura y Selección del Modelo

Para realizar el filtrado inverso de la señal del acelerómetro, se seleccionó una Red Convolucional Temporal (TCN). Esta decisión se fundamenta en la comparativa realizada por Bai et al. [43], donde se demuestra que las TCN presentan ventajas importantes sobre arquitecturas recurrentes como LSTM y GRU. Específicamente, las TCN requieren menos parámetros, permiten un procesamiento en paralelo y presentan un entrenamiento más estable al evitar los problemas de gradientes, mostrando además una mayor capacidad de retención de memoria.

Por otro lado, los Transformers, en escenarios con ventanas temporales relativamente cor-

tas (como las que se usan en este trabajo), la capacidad de modelado resulta sobredimensionada y con un coste computacional considerablemente mayor. Si bien los transformadores muestran resultados sobresalientes en secuencias largas, para secuencias cortas no muestran los mismos beneficios[44]. Además un modelo basado en transformadores demanda recursos que no se encuentran disponibles en el presente trabajo, lo que vuelve su uso poco realista.

Finalmente, no se ha encontrado que las TCN hayan sido exploradas previamente en la literatura para la tarea de filtrado inverso glotal. Esto convierte su aplicación en este trabajo en un enfoque novedoso dentro del área.

Para la elección de los hiperparámetros de la TCN, tales como el número de capas, el tamaño del filtro (o kernel) y el factor de dilatación, se empleará un método de optimización bayesiana. Esta técnica permite explorar el espacio de hiperparámetros de forma más eficiente que una búsqueda aleatoria [45]. A diferencia de estos métodos, la optimización bayesiana construye un modelo probabilístico que relaciona los hiperparámetros con el rendimiento. En cada iteración, utiliza este modelo para decidir de forma inteligente cuál es la siguiente combinación de parámetros a evaluar, lo que le permite converger más rápidamente a una solución óptima [46].

4.4.2. Entrenamiento y Evaluación del Modelo

El modelo será entrenado bajo un esquema de aprendizaje supervisado, mapeando secuencias de la señal de acelerómetro (X) a secuencias de flujo glotal de referencia (Y). Como función de pérdida, se utilizará el Error Cuadrático Medio (MSE), una elección estándar para tareas de regresión de series temporales. La función de pérdida se define como:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

donde y_i es el valor real de la señal de referencia, $\hat{y}_i$ es el valor predicho por el modelo y N es el número total de muestras en la secuencia. Cabe destacar que minimizar el MSE es equivalente a minimizar la norma L2 del error, función de perdida utilizado en el trabajo de Langheinrich et al. [28].

Para la evaluación del rendimiento del modelo se medirán dos métricas de regresión estándar sobre el conjunto de prueba para cuantificar el error general de la estimación: El Error Absoluto Medio (MAE) y la Raíz del Error Cuadrático Medio (RMSE).

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \quad , \quad \text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$$

El MAE representa la magnitud promedio del error, proporcionando una medida directa de la desviación. Por su parte, el RMSE penaliza en mayor medida los errores grandes debido al término cuadrático, siendo sensible a la presencia de valores atípicos o predicciones muy desviadas.

5. Implementación

En esta sección se muestran los detalles específicos con respecto a la implementación del sistema. Se detallan las decisiones y funciones utilizados para llevar a cabo la metodología, incluyendo la preparación de los datos e implementación de la red neuronal.

5.1. Preparación Automatizada de Datos

Dado el gran volumen de datos disponible, es necesaria una etapa de procesamiento automatizada y robusta. El pipeline se diseño con la intención de generar un conjunto de datos de calidad, y descartar cualquier archivo que pudiera contaminar el entrenamiento de la red neuronal. El proceso implementado consiste en una secuencia de filtrado, validación, estimación y normalización. Los parámetros del proceso, como umbrales, frecuencias de corte y otros, fueron establecidos mediante un proceso iterativo de análisis teórico y validación práctica. Cada etapa será detallada a continuación. El código se puede consultar en el apéndice A.1.

5.1.1. Preprocesamiento de las Señales

El primer paso consiste en filtrar las señales de micrófono (MIC) y acelerómetro (ACC) para eliminar componentes de frecuencia no deseadas. El diseño de los filtros se basa en el análisis de la densidad espectral de potencia (PSD) de las señales. Cada PSD se estima mediante la función *pwelch* de Matlab con una frecuencia de muestreo de 20 kHz, una ventana de Hamming de 0.5 s y un solapamiento del 50 %, lo que proporciona una resolución espectral de 2 Hz.

Para todos los filtros diseñados en este trabajo, se opta por filtros FIR de alto orden ($N = 1000$). Dado que el procesamiento es *offline*, el costo computacional no es una restricción. Esta elección ofrece simplicidad en la implementación, estabilidad numérica y una fuerte atenuación en las bandas de rechazo. Además, se utiliza la función *filtfilt* de Matlab para aplicar los filtros, lo que garantiza una respuesta de fase cero al procesar la señal en ambas direcciones, evitando así la distorsión de la forma de onda.

- **Señal de Micrófono (MIC):** La señal acústica será utilizada tanto para la estima-

ción de los GCIs como para la estimación del flujo glotal. Por ello, es fundamental preservar el contenido frecuencial asociado a las formantes del tracto vocal. Sin embargo, las grabaciones pueden contener ruido ajeno al proceso de fonación, que se encuentra por debajo de la frecuencia fundamental (F_0). En la Figura 11 se observa un ejemplo de la PSD de una señal de micrófono. Para atenuar estas componentes no deseadas, se aplica un filtro pasa altos con una frecuencia de corte de 55 Hz. Esta elección permite eliminar el ruido de baja frecuencia sin afectar las componentes de voz, considerando que el rango de F_0 de interés comienza en 60 Hz. La respuesta en magnitud del filtro se muestra en la Figura 12. Las señales en la base de datos pueden venir invertida. En ese caso, la función *polarity_reskew* de la librería abierta COVAREP [47] se utiliza para detectarlo y corregirlo.

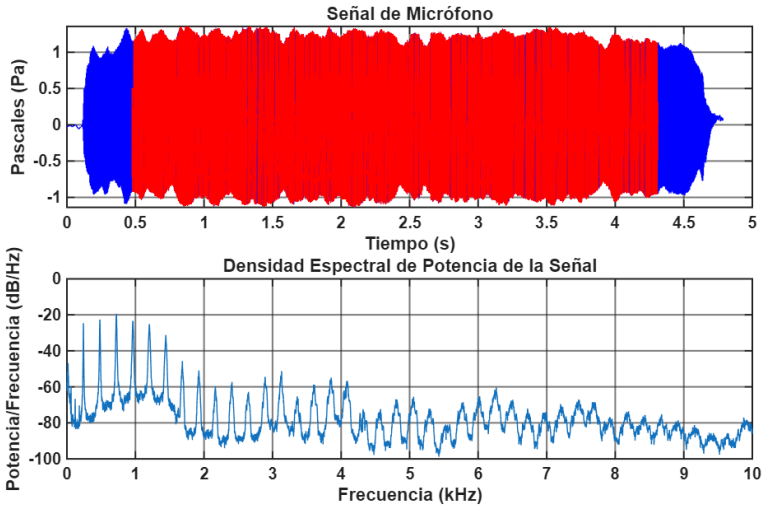


Figura 11: Ejemplo de la densidad espectral de potencia (PSD) para una señal de micrófono de una vocal /a/ sostenida.

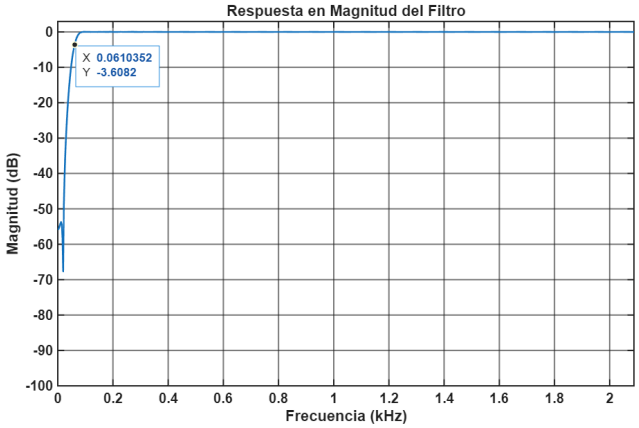


Figura 12: Magnitud del filtro FIR pasa altos ($f_c = 55$ Hz) aplicado a la señal de micrófono. Se señala un punto cercano a la frecuencia de corte

- **Señal de Acelerómetro (ACC):** La señal del acelerómetro captura la vibración

de los tejidos del cuello producida por el movimiento de las cuerdas vocales. El contenido de interés de esta señal se concentra principalmente en bajas frecuencias. La Figura 13 muestra la PSD de una señal de acelerómetro, en la que se observa una caída significativa de energía, ya que la diferencia entre el primer armónico y el octavo armónico, ubicado aproximadamente en 2 kHz, supera los 50 dB. Con el fin de aislar la banda de interés se diseña un filtro pasabanda entre 20 Hz y 2000 Hz. La frecuencia de corte inferior (20 Hz) elimina el posible desplazamiento de DC y los artefactos de movimiento corporal de muy baja frecuencia. La frecuencia de corte superior (2000 Hz) preserva la morfología de la señal relacionada con la vibración de las cuerdas vocales mientras atenúa el ruido de alta frecuencia que no aporta información relevante. Este filtrado permite presentarle al modelo una señal de entrada más limpia, facilitando el aprendizaje de la red. La respuesta en frecuencia de este filtro se presenta en la Figura 14.

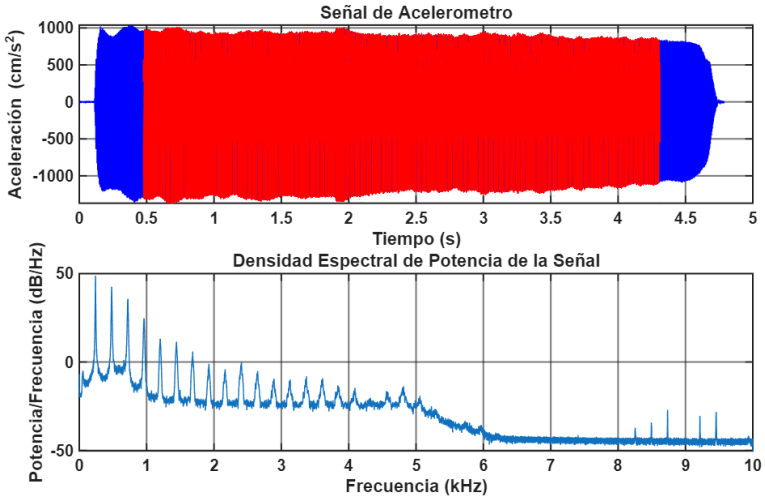


Figura 13: Ejemplo de la densidad espectral de potencia (PSD) para una señal de acelerómetro.

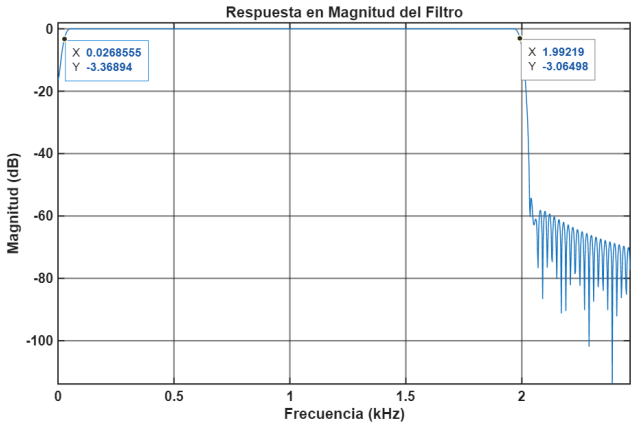


Figura 14: Respuesta en frecuencia y fase del filtro FIR pasabanda (20 Hz - 2000 Hz) aplicado a la señal de acelerómetro. Se señalan puntos cerca de las frecuencia de corte.

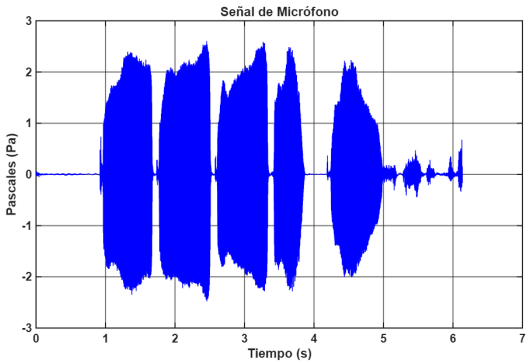
5.1.2. Validación y Descarte Automático de Señales

Para garantizar la calidad de los datos, se implementó un proceso de validación en múltiples etapas. Un archivo es descartado tan pronto como una de las verificaciones falla.

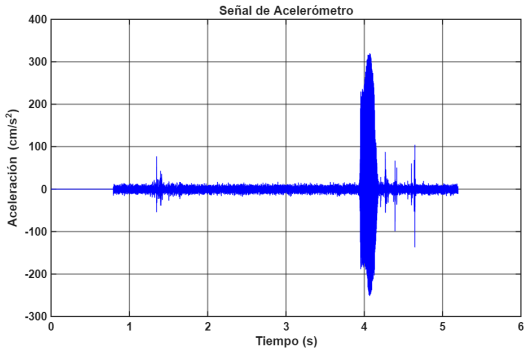
Primero, se realiza una comprobación básica de integridad (*sanity check*) que verifica que las señales existan, no contengan valores no finitos (NaN o Inf) y tengan una duración mínima de un segundo (20000 muestras), asegurando que haya suficiente información para un análisis robusto.

A continuación, las señales que se espera que correspondan a vocales sostenidas son sometidas a dos pruebas específicas:

1. **Verificación de vocal sostenida:** Se utiliza la función *detectSpeech* de Matlab para identificar las regiones con actividad de voz. Para que una grabación sea considerada una vocal sostenida válida, debe cumplir dos condiciones: primero, debe detectarse una única región de voz continua. Segundo, esta región debe ocupar al menos el 85 % de la duración total del archivo. Este criterio permite descartar grabaciones con fonaciones interrumpidas, como se ilustra en la Figura 15a, o aquellas donde la actividad vocal no fue bien registrada, como en el caso de la Figura 15b.



(a) Señal de micrófono con fonación interrumpida.



(b) Señal de acelerómetro registrada erróneamente.

Figura 15: Ejemplos de señales descartadas por el criterio de *vocal sostenida*.

2. **Verificación de periodicidad:** La naturaleza de una vocal sostenida implica una alta periodicidad. Esta característica se evalúa mediante la autocorrelación de la señal en la región de voz detectada. La autocorrelación mide la similitud de una señal consigo misma en diferentes desfases temporales. En una señal periódica, esta función presenta picos altos en los múltiplos del período fundamental. Se calcula un "puntaje de periodicidad" como la altura del peak de autocorrelación más alto dentro del rango de frecuencias vocales (60 a 400 Hz), rango definido en la sección 3.1. Si este puntaje, normalizado a un máximo de 1, es inferior a un umbral de 0.7, se considera que la señal no es suficientemente periódica y es descartada. Este criterio permite descartar señales que no presentan una estructura armónica clara, como aquellas compuestas principalmente por ruido, tal como se muestra en la Figura 16.

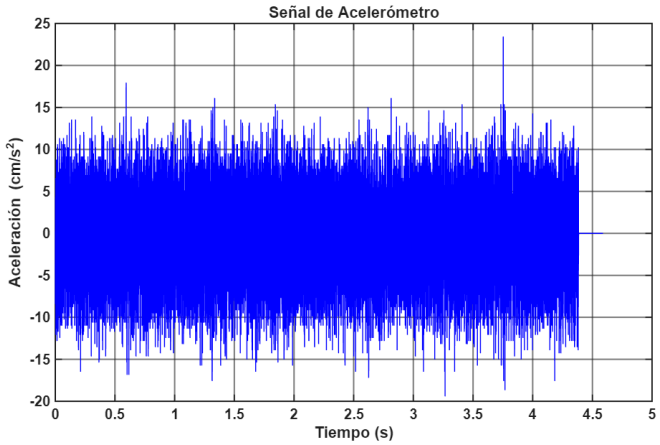


Figura 16: Ejemplo de señal de acelerómetro descartada por el criterio de *puntaje de periodicidad*. Se observa que la señal no presenta una estructura periódica definida.

Estas validaciones se aplican a las señales originales del micrófono, acelerómetro y flujo

glotal filtrado. Este último como medida de protección para que a la red de que pueda entrar una señal con artefactos.

5.1.3. Detección de GCIs desde la Señal de Micrófono

Para poder aplicar el método QCP, es necesario contar con una estimación de la ubicación de los GCIs. Este proceso se realiza a partir de la señal de micrófono ya filtrada, utilizando herramientas del toolbox COVAREP.

Primero, se aísla el segmento de interés de la señal que contiene la fonación mediante la función *detectSpeech*. Sobre esta región de voz se aplica el algoritmo *pitch_srh* para estimar la frecuencia fundamental (F_0). Como el cálculo se realiza por ventanas, el resultado de la función es una secuencia de valores de F_0. Para obtener un único valor se emplea la mediana de la secuencia, lo que reduce la influencia de posibles valores atípicos.

Posteriormente, esta F_0 se utiliza para guiar al algoritmo *gci_sedreams*, que opera sobre el mismo segmento de interés de la voz para proporcionar la localización de los GCIs. Sin embargo, es importante notar que esta estimación, aunque funcional, no es perfecta. Como se puede observar en la Figura 17, los marcadores de GCI pueden presentar un desfase de los puntos donde ocurre la máxima excitación del tracto vocal en la señal de micrófono. Este desplazamiento es, dentro de todo, tolerable, ya que la función de ponderación AME del método QCP define una ventana de análisis con rampas que se activan unas muestras después del GCI detectado y se desactiva unas muestras antes del siguiente GCI.

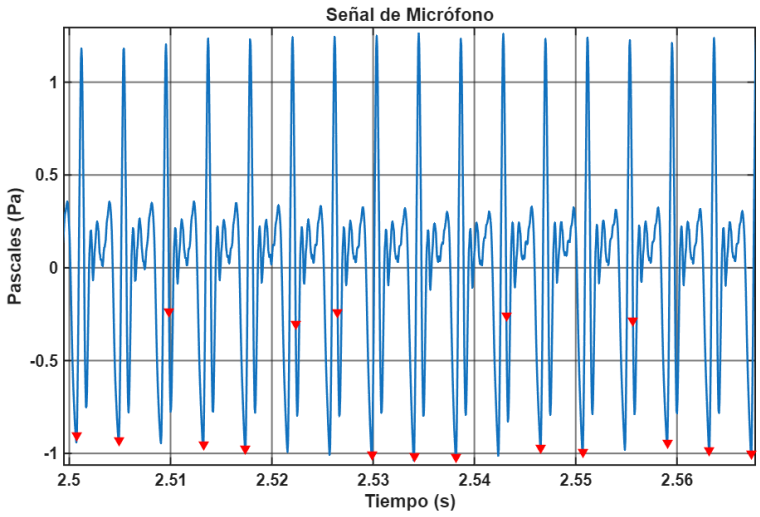


Figura 17: Ejemplo de la localización inicial de GCIs (marcadores rojos) sobre la señal de micrófono filtrada. Se observa algunos instantes desfasados de los peaks negativos de la señal.

Finalmente, estos GCIs iniciales son validados para asegurar su coherencia. La validación comprueba que: primero, haya un número mínimo de GCIs para considerar la fonación estable. Segundo, que la diferencia entre GCIs consecutivos sea estable, descartando señales donde haya saltos abruptos que sugieran fallos en la detección. Y tercero, que la F_0 media, calculada a partir de estos períodos, se encuentre dentro del rango de 60 a 400 Hz.

5.1.4. Estimación del Flujo Glotal mediante QCP

Con la estimación de GCIs, se procede a obtener el flujo glotal a partir de la señal de micrófono utilizando QCP. El código del método utilizado en este trabajo fue obtenido a partir de la implementación publicada por Freixes et al. [25]. Dicho código fue modificado para mejorar la legibilidad e incluir la función WLP, ausente en la versión publicada en el repositorio. La implementación de WLP se tomo del código fuente disponible públicamente en un repositorio de Github del usuario jopoh [48]. El código se encuentra en el apéndice A.2.

Primero, el método divide la señal en segmentos solapados. En cada segmento se aplica un pre-énfasis para realzar las frecuencias altas, y se enventana con una ventana de hanning para facilitar la estimación del tracto vocal cuando se haga el WLP. A partir de los GCIs se construye la función de ponderación AME, que reduce la influencia de las muestras cercanas al cierre glotal y mantiene el resto con peso unitario. Con esta ponderación se aplica WLP sobre el segmento modificado para estimar un modelo de solo polos del tracto vocal. La señal original del segmento se filtra inversamente con este modelo, obteniéndose la derivada del flujo glotal. Esta señal se integra con un coeficiente de integración $\rho = 0{,}99$, un valor común en el estado del arte, que compensa el efecto de la radiación labial y permite recuperar el flujo glotal estimado. Finalmente, cada segmento procesado se coloca en su intervalo correspondiente dentro de la señal de salida, y el estado de los filtros usados se transfiere entre segmentos, lo que permite reconstruir de manera continua la estimación del flujo glotal a lo largo de toda la señal de voz.

La construcción de la función AME requiere la definición de los parámetros descritos en la sección 3.2.1. En el trabajo que detalla el método QCP [22], sus autores propusieron valores óptimos para los parámetros del algoritmo, los cuales se basan en la frecuencia fundamental del hablante. Dado que se cuenta con una estimación de F_0, en este trabajo se adoptan los valores recomendados por los autores: $PQ = 0{,}05$, $N_{\text{ramp}} = 7$, y un valor

para DQ que depende de la F_0 estimada:

$$\begin{aligned} F_0 < 190 \text{ Hz} &\quad \Rightarrow DQ = 0{,}95 \\ 190 \leq F_0 < 280 \text{ Hz} &\quad \Rightarrow DQ = 0{,}55 \\ F_0 \geq 280 \text{ Hz} &\quad \Rightarrow DQ = 0{,}7 \end{aligned}$$

Para determinar el orden del filtro vocal, el tamaño de la ventana de análisis y el salto entre ventanas (hop size), se seguirá la configuración utilizada por Chien et al. [15], ya que en este estudio también emplea una frecuencia de muestreo de 20 kHz. En dicho trabajo, los autores emplean un orden de predicción lineal de p=20 para modelar hasta 10 formantes por debajo de la frecuencia de Nyquist (10 kHz), lo cual es crítico para capturar con precisión las resonancias del tracto vocal. Asimismo, se utiliza un tamaño de ventana de análisis de 32 ms y un salto entre ventanas de 16 ms. De esta manera, y como se indicó en la sección 3.2.1, se cuenta con ventanas donde puede asumirse estacionariedad, mientras que el salto permite suavizar las transiciones entre ellas.

El resultado de aplicar este proceso se puede observar en la Figura 18. Se aprecia que el instante de cierre glotal, marcado por el peak negativo más pronunciado en la derivada del flujo, coincide temporalmente con el inicio del ciclo de mayor excitación en la señal de micrófono. No obstante, también se observan artefactos, particularmente en la fase abierta de la derivada del flujo glotal. Estas distorsiones serán tratadas en la etapa posterior de procesamiento.

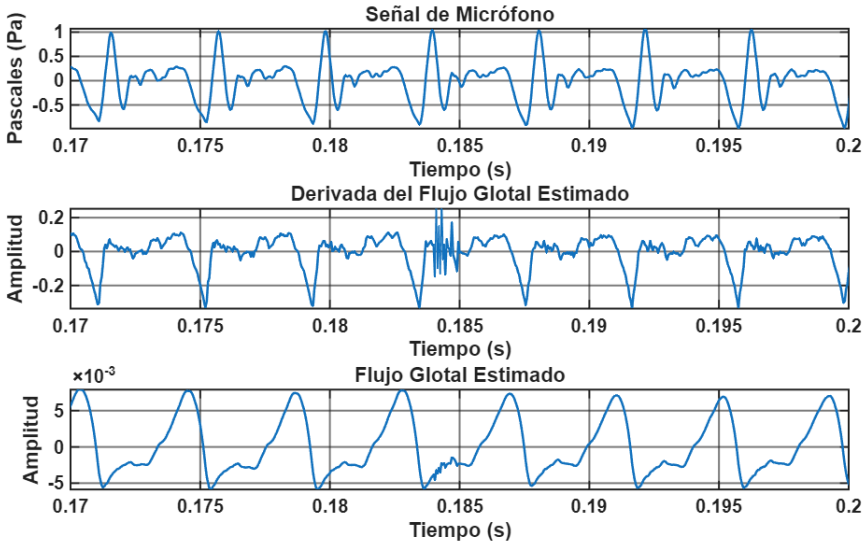


Figura 18: Ejemplo del resultado del filtrado inverso en una ventana temporal. Arriba: señal de micrófono. Centro: derivada del flujo glotal estimado. Abajo: flujo glotal estimado.

Cabe notar que las amplitudes no se colocan en unidades físicas de (L/s). Esto se debe a que se encontraron inconsistencias en las amplitudes resultantes del filtrado inverso. La Figura 19 muestra un ejemplo de este fenómeno. Mientras la envolvente de la señal de micrófono (arriba) permanece relativamente estable, la envolvente de la derivada del flujo glotal (centro) muestra un notorio incremento de amplitud cerca del segundo cuatro.

Para verificar si este aumento corresponde a un cambio real en la fonación, se extrajeron y alinearon dos segmentos de la señal de micrófono, uno de la zona de baja amplitud de la derivada y otro de la zona de alta amplitud. Como se muestra en la Figura 20, las formas de onda son similares, confirmando que la variación de amplitud es un artefacto del proceso de estimación. Se atribuye esta inestabilidad a sensibilidades numéricas en el algoritmo WLP. Para corroborar que no es un problema exclusivo de la implementación actual, se comparó el resultado con el método IAIF de COVAREP. La Figura 21 demuestra que IAIF presenta el mismo problema.

Esta inconsistencia en la amplitud es la razón fundamental por la que se debe realizar una normalización posterior. Entrenar una red neuronal con estas señales sería problemático, ya que el modelo aprendería que formas de onda de entrada idénticas pueden corresponder a salidas con amplitudes muy diferentes. La normalización de energía ciclo a ciclo, que se detallará más adelante, elimina esta ambigüedad y permite que el modelo se enfoque únicamente en la morfología de la señal.

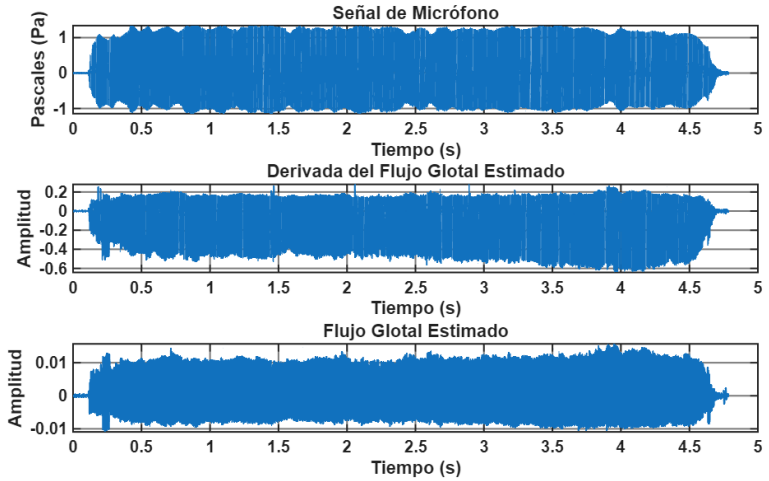


Figura 19: Comparación de las envolventes de las señales. Se observa un aumento de amplitud en la derivada del flujo glotal (centro) que no se corresponde con la señal de micrófono (arriba).

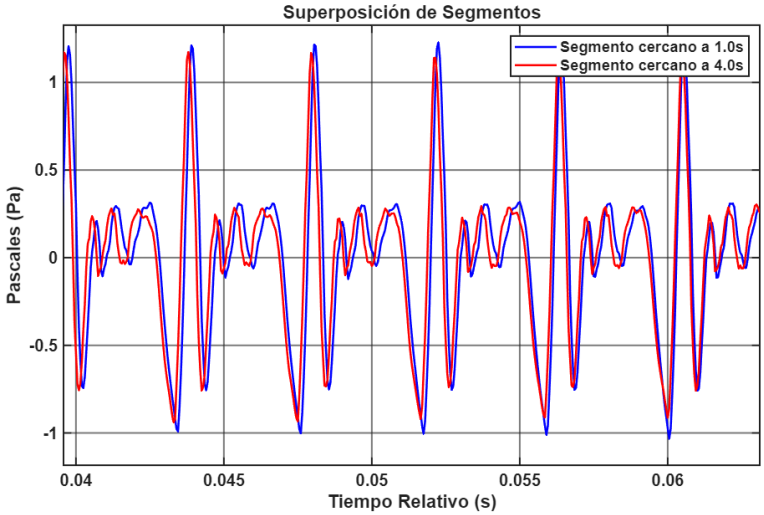


Figura 20: Superposición de dos segmentos de la señal de micrófono de la Fig. 19. La similitud en amplitud demuestra que la variación en la envolvente estimada es un artefacto.

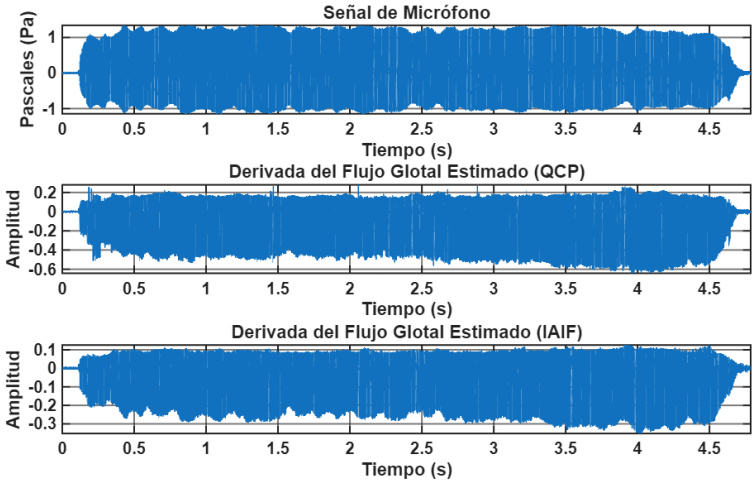
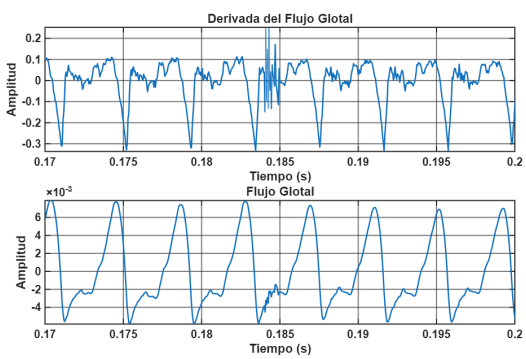


Figura 21: Comparación de la derivada del flujo glotal estimada con QCP (centro) e IAIF (abajo). Ambos métodos exhiben un artefacto de amplificación similar.

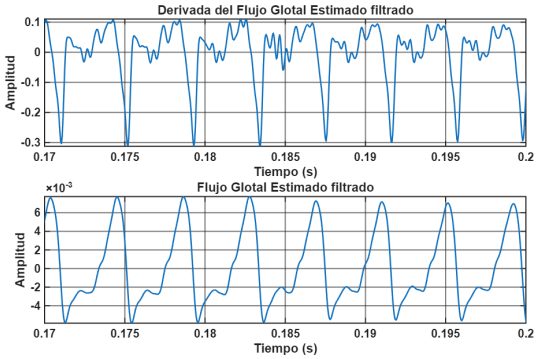
5.1.5. Filtrado del Flujo Glotal Estimado

Tras la estimación con QCP, tanto el flujo glotal (G) como su derivada (dG) son sometidos a un filtrado pasabanda para eliminar los artefactos residuales mencionados anteriormente y refinar su morfología. La frecuencia de corte inferior se fija en $0{,}8 \times F_0$ para eliminar cualquier componente de DC o deriva de baja frecuencia introducida durante la integración del QCP. La frecuencia de corte superior se establece en 3000 Hz para preservar la forma del peak negativo de la derivada y así, alinear mejor la señal de acelerómetro en el siguiente paso. Y 2000 Hz para el flujo, ya que la envolvente del espectro es principalmente de baja frecuencia [49]. De esta forma se atenúa el ruido de alta frecuencia y se

suaviza la forma de onda preservando la morfología, como se ilustra en la Figura 22.



(a) Señales antes del filtrado.



(b) Señales después del filtrado.

Figura 22: Efecto del filtrado pasabanda adaptativo sobre un segmento del flujo glotal estimado y su derivada. Se observa una reducción de los artefactos en la fase abierta y un suavizado general de las formas de onda.

5.1.6. Detección de GCIs a partir de la Derivada del Flujo Glotal

Si bien los GCIs iniciales obtenidos de la señal de micrófono son adecuados para guiar el algoritmo QCP, para la posterior normalización ciclo a ciclo se requiere una localización más precisa del instante de cierre glotal. La derivada del flujo glotal (dG) es ideal para este propósito, ya que el evento de cierre se es un peak negativo prominente y agudo, proporcionando un marcador temporal ideal.

El procedimiento para refinar la detección de GCIs se aplica sobre la región de voz de la señal dG ya filtrada. Primero, la señal se normaliza restando su media y dividiéndola por su desviación estándar (z-score). Esto permite establecer un umbral de detección de peaks que es independiente de la amplitud de la señal. A continuación, la señal normalizada se invierte, convirtiendo los peaks negativos en peaks positivos para la etapa siguiente.

Sobre esta señal invertida se utiliza la función *findpeaks* de Matlab para localizar los nuevos GCIs. La búsqueda se restringe con dos parámetros clave: un umbral de altura mínima (*MinPeakHeight*) de 0.5 para ignorar peaks espurios de baja amplitud, y una distancia mínima entre peaks (*MinPeakDistance*). Esta distancia se calcula a partir de la máxima frecuencia fundamental esperada (400 Hz), asegurando que no se detecte más de un GCI por ciclo glotal. El resultado es un conjunto de marcadores temporales de alta precisión, como se ilustra en la Figura 23.

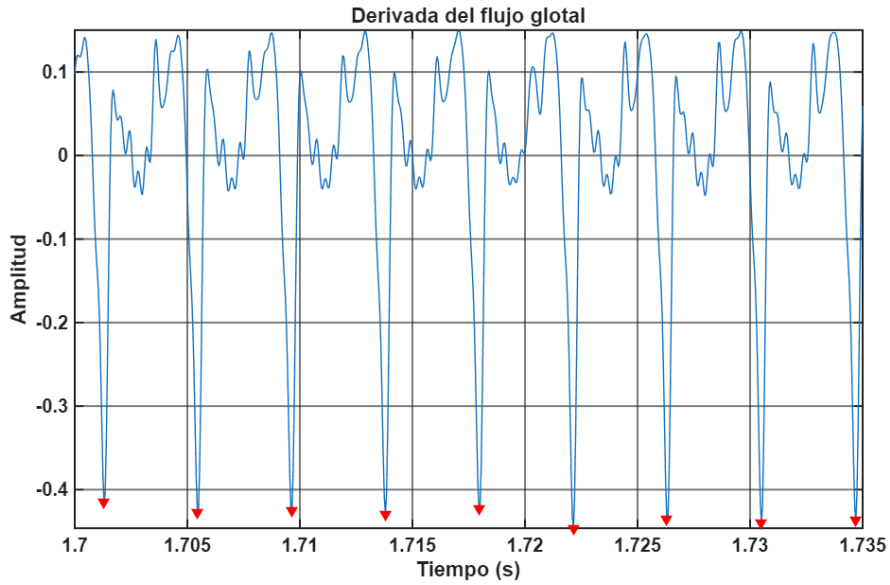


Figura 23: Ejemplo de la detección de GCIs refinados (marcadores rojos) sobre la derivada del flujo glotal (dG). Los marcadores se alinean con precisión en los peaks negativos más pronunciados de la señal.

Finalmente, al igual que con la estimación inicial, este nuevo conjunto de GCIs es sometido a un proceso de validación para confirmar su consistencia temporal y que la frecuencia fundamental media se encuentre dentro del rango plausible. Si la validación falla, el archivo es descartado.

5.1.7. Alineación Temporal

Una vez refinadas las señales glotales y sus GCIs, estas deben sincronizarse con la señal de acelerómetro.

Aunque las señales de micrófono y acelerómetro se graban de forma sincronizada, esto no garantiza que los eventos glotales estén alineados temporalmente en ambas. Esta falta de alineación se debe a los diferentes trayectos que recorre la señal. La señal acústica se propaga desde la glotis a través del tracto vocal y el aire hasta llegar al micrófono. De forma paralela, la vibración de las cuerdas vocales se transmite a través de los tejidos del cuello hasta el acelerómetro.

Dado que las distancias y los medios de propagación son distintos, se produce un desfase temporal entre la señal estimada del flujo glotal y la señal del acelerómetro. Este desfase no es constante entre grabaciones, ya que depende de factores como la colocación exacta de los sensores.

La presencia de un desfase representa un problema para el entrenamiento de una red neuronal. Si las señales tienen desfases distintas, la función de pérdida comparará muestras que no se corresponden en el tiempo. Como resultado, penalizaría fuertemente al modelo por producir una forma de onda correcta pero desplazada temporalmente, lo que impediría que aprenda la relación morfológica real entre la señal de entrada y la de salida. Por esta razón, es fundamental realizar un proceso de alineación

Para evitar esto, se alinea la señal del acelerómetro utilizando la derivada del flujo glotal (dG) como referencia, ya que sus picos agudos permiten una sincronización más precisa. El alineamiento se realiza con una función basada en correlación cruzada normalizada que encuentra el retardo que maximiza la similitud. Este retardo, usualmente de unas pocas muestras (equivalente a ≈ 1 ms a 20 kHz), se corrige mediante un desplazamiento circular. Este tipo de desplazamiento no afectan el contenido de la señal, ya que las grabaciones comienzan y terminan con segmentos no sonoros. La Figura 24 muestra un ejemplo de este proceso.

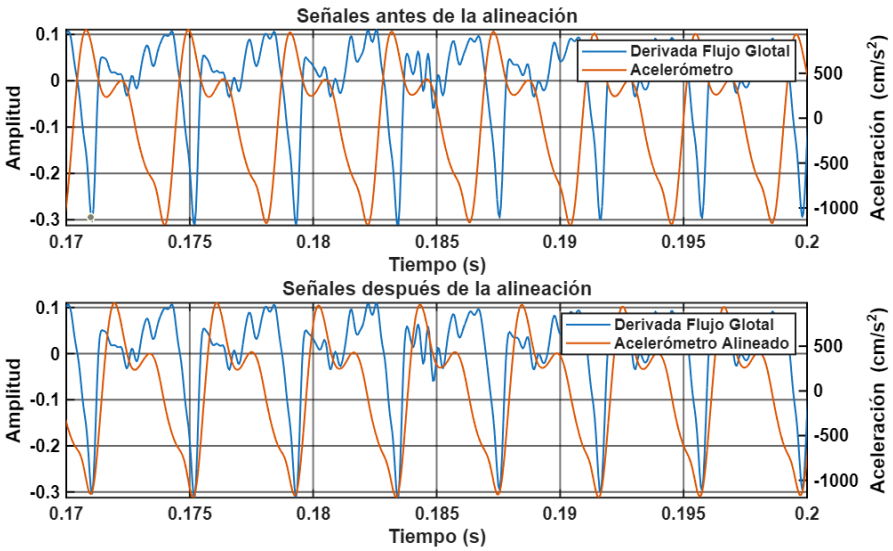


Figura 24: Ejemplo de alineación temporal entre la señal de acelerómetro (ACC) y la derivada del flujo glotal (dG). Arriba se muestran las señales con su desfase original y abajo sincronizadas tras la corrección.

5.1.8. Estandarización de las señales

Como se presentó en la sección 5.1.4, el flujo glotal estimado sufre de inconsistencias en la amplitud. A esto se le suma que la amplitud del acelerómetro puede presentar un comportamiento distinto a la del micrófono, tal como se ilustra en la Figura 25. En este ejemplo, mientras la envolvente de la señal del micrófono disminuye con suavidad, la del

acelerómetro muestra un decaimiento notorio.

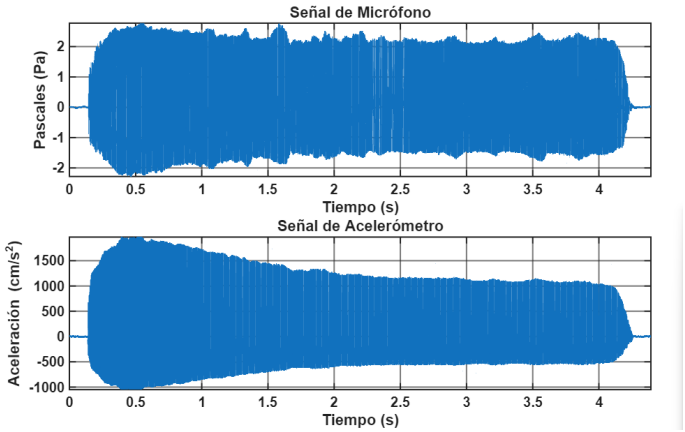


Figura 25: Comparación de las señales de micrófono y acelerómetro para una misma grabación. Se observa que la envolvente de la señal del acelerómetro (abajo) presenta un decaimiento de amplitud más pronunciado que el de la señal de micrófono (arriba).

Para facilitar que la red neuronal aprenda la relación morfológica, se estandariza la amplitud mediante una normalización de energía ciclo a ciclo. Para ello, se utilizan los GCIs obtenidos en la sección 5.1.6 para segmentar las señales G y ACC en ciclos glotales individuales. A continuación, se calcula la energía (norma L2) de cada ciclo y cada muestra se divide por la raíz cuadrada de este valor. Este procedimiento asegura que cada ciclo tenga energía unitaria, lo que preserva la forma de la onda y permite que la red se enfoque en ella en lugar de la amplitud absoluta. El resultado de este proceso se puede observar en la Figura 26.

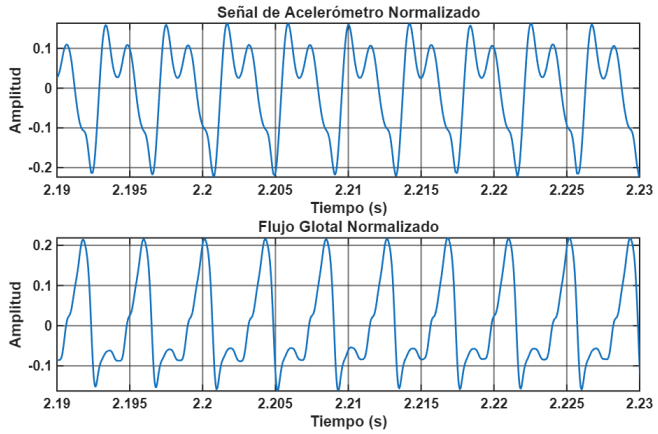


Figura 26: Resultado de la normalización de energía ciclo a ciclo aplicada a la señal de acelerómetro (ACC) y al flujo glotal estimado (G) de la grabación mostrada en la Figura 25.

5.1.9. Remuestreo y Almacenamiento

Como paso final, las señales G y ACC, ya procesadas, alineadas y normalizadas, se remuestrean de 20 kHz a 8 kHz. Esta reducción de la frecuencia de muestreo disminuye los costos computacionales del entrenamiento sin perder información relevante, ya que el contenido útil de estas señales se concentra en frecuencias inferiores a 4 kHz. Paralelamente, los índices de los GCIs se reescalan para que su posición temporal corresponda con la nueva frecuencia de muestreo. Luego, se aplica una última detección de voz para recortar cualquier silencio remanente en los extremos de las señales.

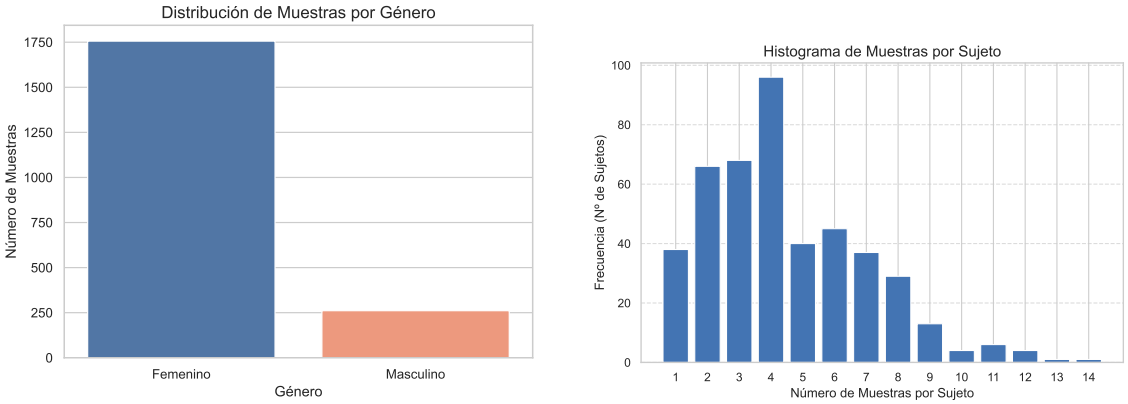
Finalmente, los pares de señales resultantes (G y ACC), junto con los GCIs finales, se guardan en archivos .mat individuales. Cada archivo incluye además metadatos relevantes como el identificador del sujeto, el género y el tipo de fonación, facilitando el análisis posterior del dataset generado.

5.2. Implementación de la Red Neuronal

Una vez generado el dataset con las señales de acelerómetro (ACC) y flujo glotal (G) procesadas, se procedió a la implementación, entrenamiento y evaluación del modelo TCN. Para este fin, se utilizó el lenguaje de programación Python junto con la librería de aprendizaje profundo PyTorch [50] y PyTorch Lightning para la estructuración del código de entrenamiento [51].

5.2.1. Dataset Resultante y Submuestreo

El pipeline de procesamiento descrito en la sección 5.1 generó un dataset final con una duración total de 2.68 horas (9661.5 segundos) de fonaciones sostenidas de la vocal /a/. La distribución del conjunto de datos, como se muestra en la Figura 27, evidencia una mayor cantidad de muestras de sujetos femeninos (1756 grabaciones) en comparación con los masculinos (261 grabaciones). Además, la cantidad de grabaciones por sujeto no es uniforme como era de esperar, ya que los sujetos que recibieron tratamiento pudieron haber tenido hasta 3 sesiones de grabación, con un promedio de 4.5 muestras por sujeto.



(a) Distribución de muestras por género.

(b) Histograma de muestras por sujeto.

Figura 27: Estadísticas del dataset generado tras la etapa de preprocesamiento.

5.2.2. Preparación de los Datos para el Entrenamiento

Debido a restricciones de tiempo, se seleccionó un subconjunto representativo de 50 minutos del dataset total. Para evitar el sesgo hacia sujetos con mayor número de muestras, se implementó un método de submuestreo probabilístico. Este método asigna una mayor probabilidad de eliminación a las muestras pertenecientes a sujetos sobrerrepresentados. De esta manera el modelo evita sobreajustarse a una anatomía en particular.

El subconjunto de 50 minutos se dividió en tres conjuntos: 70 % para entrenamiento, 15 % para validación y 15 % para prueba. Se adoptó una estrategia de división por sujeto, asegurando que todas las grabaciones de un mismo hablante pertenezcan a un único conjunto. Este enfoque garantiza que el modelo se evalúe sobre hablantes no vistos, de manera que se pueda observar su capacidad de generalización. Las señales de ACC y G se segmentaron en ventanas no solapadas de tamaño fijo, considerando únicamente el 90 % central de la señal total para minimizar la entrada de artefactos asociados a las transiciones. De esta manera, cada par de entrada-salida para la red consiste en un segmento de la señal de acelerómetro y su correspondiente segmento de la señal de flujo glotal. se aplicó un escalado Min-Max en el rango $[-1, 1]$. El escalador se ajustó de forma incremental sobre los segmentos de entrenamiento y, posteriormente, se utilizó para transformar todas las ventanas. El código se encuentra en el apéndice A.3.

5.2.3. Arquitectura del Modelo TCN

La arquitectura de la Red Convolucional Temporal (TCN) se implementó como una secuencia de bloques residuales con convoluciones dilatadas tomando como base la implementación de [52]. Cada bloque, como se detalla en el Apéndice A.4, se construye con varios componentes fundamentales. Incluye dos capas de convolución 1D, cuya función es aprender y detectar características locales en las secuencias de entrada. La no-linealidad es introducida por la función de activación ReLU. Para mejorar la generalización y mitigar el sobreajuste, se intercala una capa de Dropout entre las convoluciones. Finalmente, para asegurar una convergencia estable y eficiente, se aplicó *Group Normalization* después de cada capa convolucional. El uso de `padding='same'` en las capas convolucionales asegura que la longitud de la secuencia de salida sea idéntica a la de entrada.

5.2.4. Optimización de Hiperparámetros y Entrenamiento

Para determinar la configuración óptima del modelo TCN, se realizó un proceso de optimización que consistió en el entrenamiento completo de múltiples modelos, cada uno con una combinación de hiperparámetros distinta. El objetivo fue seleccionar el modelo que lograra minimizar el Error Cuadrático Medio (MSE) en el conjunto de validación.

Este proceso, gestionado con la librería Optuna [53], exploró el siguiente espacio de búsqueda:

- **Tamaño de ventana (k):** Selección entre $\{400, 800, 1600\}$. Se incluyó este parámetro en la búsqueda para experimentar si el modelo, al ser de naturaleza estadística, mejoraría sus predicciones al estar expuesto a las variaciones naturales de los ciclos glotales.
- **Número de bloques residuales:** Selección entre $\{12, 14, 16\}$.
- **Número de canales (hidden):** Selección entre $\{32, 64\}$.
- **Tamaño del kernel:** Selección entre $\{5, 7\}$.
- **Tasa de abandono (Dropout):** Un valor continuo en el intervalo $[0{,}0, 0{,}05]$.
- **Tasa de aprendizaje (Learning Rate):** Un valor continuo en una escala logarítmica dentro del intervalo $[1 \times 10^{-4}, 9 \times 10^{-4}]$.

Para cada combinación de hiperparámetros evaluada, se realizó un entrenamiento completo utilizando la siguiente configuración, gestionada con PyTorch Lightning [51]:

- **Función de pérdida:** Error Cuadrático Medio (MSE).
- **Optimizador:** Adam.
- **Número máximo de épocas:** 50.
- **Tamaño de lote (*batch size*):** 128.

Durante cada uno de estos entrenamientos, se implementó un mecanismo de parada temprana (*early stopping*) que detenía el proceso si la pérdida de validación no mejoraba durante 5 épocas consecutivas.

Al finalizar la búsqueda, el modelo que obtuvo el menor error en el conjunto de validación fue seleccionado como la versión final, conservando sus pesos ya entrenados para la etapa de evaluación. Los hiperparámetros de esta configuración óptima se detallan en la Tabla 2.

Tabla 2: Hiperparámetros óptimos de la TCN encontrados tras el proceso de optimización.

Hiperparámetro	**Valor Óptimo**
Tamaño de ventana (k)	1600 (200 ms)
Número de bloques residuales	14
Número de canales (hidden)	64
Tamaño del kernel	5
Tasa de abandono (Dropout)	0.0385
Tasa de aprendizaje	$2{,}96 \times 10^{-4}$

6. Resultados

En esta sección se presenta la evaluación del modelo TCN final. Se inicia con un análisis cuantitativo del rendimiento general y por subgrupos de sujetos, para luego proceder a una comparación visual cualitativa de las estimaciones.

6.1. Evaluación cuantitativa del modelo

El rendimiento del modelo se evaluó en el conjunto de prueba, compuesto por hablantes no utilizados durante el entrenamiento o la validación. Se emplearon dos métricas estándar para cuantificar el error entre la señal predicha ($\hat{y}$) y la señal de referencia (y): el Error Absoluto Medio (MAE) y la Raíz del Error Cuadrático Medio (RMSE).

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \quad , \quad \text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$$

La Tabla 3 muestra el rendimiento global del modelo, considerando todas las muestras del conjunto de prueba.

Tabla 3: Rendimiento general del modelo en el conjunto de prueba.

Métrica	**Valor**
RMSE	0.1772
MAE	0.1445

El rendimiento también fue analizado a través de los cuatro subgrupos de hablantes: Femenino Normal (NF), Masculino Normal (NM), Femenino con Patología (PF) y Masculino con Patología (PM). Este análisis permite evaluar si el modelo presenta sesgos o diferencias de rendimiento entre voces sanas y patológicas, o entre géneros. Los resultados se detallan en la Tabla 4.

Tabla 4: Rendimiento del modelo por subgrupo de hablantes.

Subgrupo	**RMSE**	**MAE**
Normal Femenina (NF)	0.1806	0.1473
Normal Masculino (NM)	0.1393	0.1139
Patológica Femenina (PF)	0.1934	0.1572
Patológico Masculino (PM)	0.1561	0.1278

Los resultados indican que el rendimiento del modelo depende tanto del género como de la condición vocal del hablante. Se observa un rendimiento superior en voces masculinas sanas, mientras que el mayor grado de error se encuentra en las voces femeninas con patología.

Este comportamiento puede atribuirse a dos factores. Por un lado, las voces femeninas poseen una frecuencia fundamental más alta. Por otro lado, las voces patológicas introducen irregularidades ciclo a ciclo y variaciones en la forma de la onda del flujo glotal. La combinación con una frecuencia elevada con estas inestabilidades genera señales más complejas, dificultando que el modelo aprenda un mapeo consistente y preciso desde la señal del acelerómetro.

6.2. Comparación visual de las estimaciones

La inspección visual de las señales estimadas complementa las métricas cuantitativas, ofreciendo una visión cualitativa de la precisión del modelo. Para este análisis, se examina un detalle de las 400 muestras centrales de cada ventana. Se presentan los tres mejores y peores casos de estimación, seleccionados mediante un muestreo espaciado.

6.2.1. Análisis de los Casos de Mayor Precisión

La Figura 28 muestra tres de las estimaciones más precisas del conjunto de prueba.

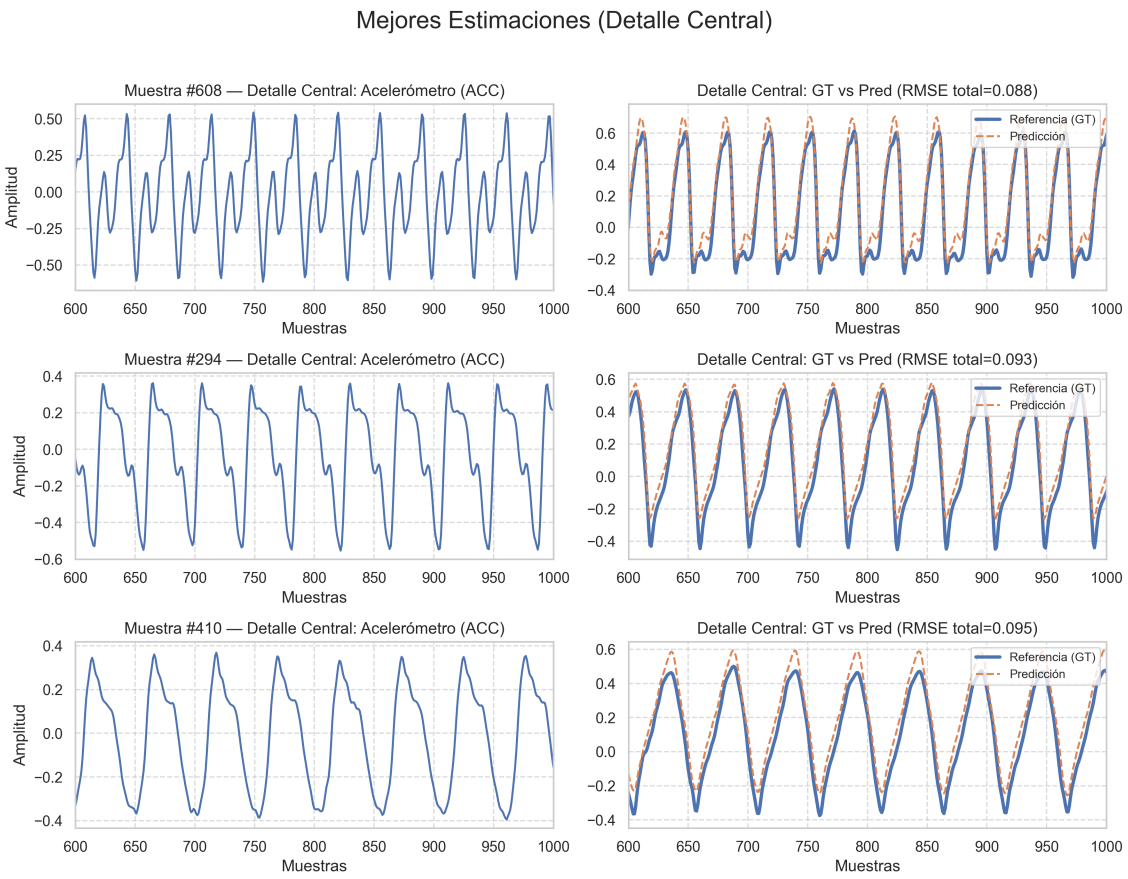


Figura 28: Ejemplos de las estimaciones más precisas (menor RMSE). Cada fila muestra un detalle de la señal de entrada (izquierda) y la comparación entre predicción y referencia para el mismo segmento central (derecha).

En estos casos, el modelo puede reproducir la morfología de la forma de onda de la referencia. Sin embargo, se observan limitaciones puntuales. Por ejemplo, en la muestra #608, el modelo presenta dificultades para seguir con exactitud la señal de referencia durante la fase cerrada del ciclo glotal de referencia. En contraste, la fase de apertura es estimada adecuadamente en todos los casos.

6.2.2. Análisis de los Casos de Menor Precisión

La Figura 29 muestra tres de las estimaciones con mayor error del conjunto de prueba.

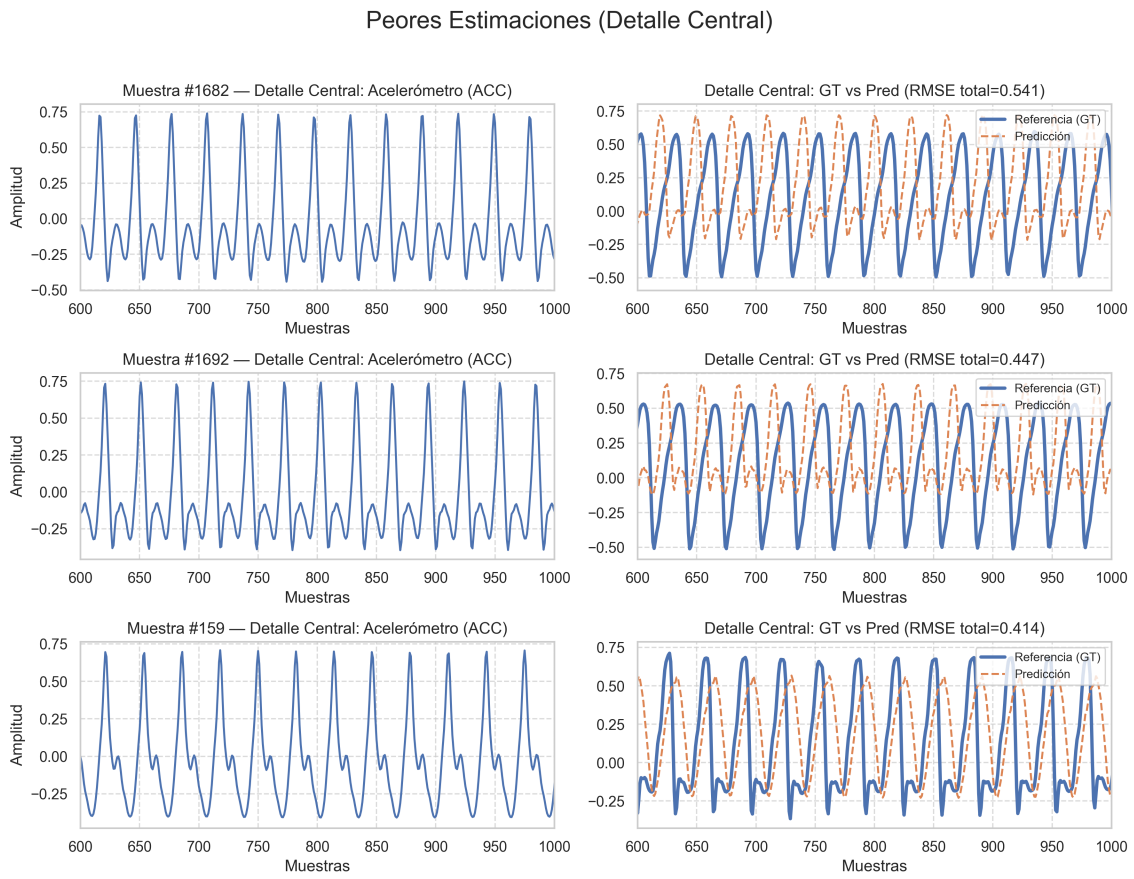


Figura 29: Ejemplos de las estimaciones con mayor error (RMSE). Cada fila muestra un detalle de la señal de entrada (izquierda) y la comparación entre predicción y referencia para el mismo segmento central (derecha).

En estas situaciones, aunque el modelo logra seguir la frecuencia fundamental de la señal, falla sistemáticamente en la estimación de la amplitud y la fase. Este comportamiento ocurre a pesar de que los datos fueron alineados y normalizados en energía durante el preprocesamiento. Esto sugiere que, pese a los filtros aplicados, pueden persistir en el conjunto de datos señales con características que afectan negativamente el aprendizaje del modelo.

7. Discusión

Los resultados cuantitativos y cualitativos demuestran la viabilidad de utilizar una red TCN para estimar la forma de onda del flujo glotal a partir de señales de acelerómetro. El modelo aprende exitosamente la relación general entre la vibración del cuello y el flujo glotal sin requerir una calibración específica por sujeto, lo cual supera una de las limitaciones de los métodos tradicionales como el IBIF. No obstante, el rendimiento del modelo queda limitado por la calidad de las referencias generadas con el método de filtrado inverso glotal que se utilice.

El mejor rendimiento del algoritmo se observa en voces masculinas y sin patologías. Esto es coherente, ya que estas voces suelen tener una frecuencia fundamental más baja y una mayor estabilidad, lo que facilita tanto la estimación del flujo glotal de referencia mediante QCP como el aprendizaje de la red neuronal. La señal en estos casos presenta una morfología clara y repetitiva, que el modelo TCN puede mapear de forma más consistente a la forma de onda del flujo glotal.

Por el contrario, el mayor error se presenta en voces femeninas con patología. Este resultado puede atribuirse a una confluencia de factores. Primero, las voces femeninas, al tener una frecuencia fundamental más alta, presentan ciclos glotales más cortos. Esto representa un desafío para los métodos de filtrado inverso, cuya precisión disminuye en frecuencias altas. Además, una estimación imprecisa de los GCIs puede empeorar aún más el rendimiento del filtrado en estos casos. En consecuencia, la calidad de las señales de referencia usadas para entrenar el modelo podría ser inferior para este subgrupo, introduciendo un "techo de rendimiento". Segundo, la presencia de patologías vocales introduce aperiocidad y variaciones en la dinámica de cierre glotal. Estas irregularidades se reflejan en la señal de acelerómetro y generan patrones más complejos y menos predecibles que dificultan el aprendizaje de la red. Esta dificultad se acentúa en las voces femeninas, ya que el modelo necesita capturar patrones que cambian rápidamente, volviendo más complejo el aprendizaje de dependencias temporales muy volátiles.

El análisis de los peores casos de estimación (29) sugiere que, a pesar del preprocesamiento, algunas señales de referencia podrían contener artefactos o inconsistencias que el modelo no puede replicar. La normalización de energía ciclo a ciclo fue una medida necesaria para mitigar las inestabilidades de amplitud del método QCP, pero fallos en

la segmentación de los ciclos o errores en la estimación de la fase cerrada por parte de QCP pueden generar formas de onda de referencia atípicas. El modelo, al estar entrenado para producir una morfología promedio, falla en estos casos extremos, produciendo una versión más "suavizada.º desfasada de la señal.

En cuanto a las condiciones de grabación, los datos fueron adquiridos en condiciones controladas de laboratorio. Aunque el acelerómetro es menos sensible a factores como el ruido ambiental, el rendimiento del algoritmo no fue evaluado en condiciones reales como las de un monitoreo ambulatorio. En dicho escenario, su efectividad podría verse afectada por factores como el movimiento de la persona o las variaciones en la posición del sensor si no se tratan adecuadamente.

8. Conclusión

En este trabajo se desarrolló e implementó un sistema completo para la estimación del flujo glotal a partir de señales de acelerómetro. Parte fundamental del estudio fue la creación de un método automatizado capaz de procesar y validar una base de datos para generar las señales de referencia necesarias para el entrenamiento.

Se exploró el uso de una Red Convolucional Temporal (TCN), una arquitectura no reportada previamente para la tarea de filtrado inverso glotal. Los resultados demuestran que este enfoque es viable y que el modelo es capaz de aprender la relación entre la señal del acelerómetro y el flujo glotal sin necesidad de una calibración por sujeto. Sin embargo, el análisis también revela que el principal cuello de botella del procedimiento está en la calidad de las referencias. El rendimiento del modelo depende directamente de la precisión de los métodos de filtrado inverso glotal tradicionales, como QCP, utilizados para generar el conjunto de datos.

Como trabajo futuro, se propone ampliar la evaluación del modelo. Sería útil realizar un análisis comparativo de parámetros glotales entre las predicciones y las referencias para cuantificar si el modelo captura adecuadamente las características dinámicas del flujo. También es relevante estudiar cómo los diferentes tipos de fonación afectan el rendimiento y probar el modelo fuera de su dominio de entrenamiento, por ejemplo, con vocales como la /i/ o la /u/, para evaluar su capacidad de generalización.

Finalmente, este estudio refuerza la idea de que los modelos estadísticos pueden ser un complemento valioso para los modelos físicos en la investigación de la fisiología de la voz. Debido a la dificultad para obtener referencias reales del flujo glotal, ambos paradigmas pueden complementarse. Una futura línea de investigación podría ser el uso de "Transfer Learning", entrenando un modelo con datos sintéticos para luego ajustarlo con datos clínicos como los de este trabajo. Otra opción es implementar redes neuronales informadas por la física para incorporar el conocimiento de modelos como el IBIF y así mejorar las predicciones.

A. Código

A.1. Generación de Dataset

El siguiente código tiene la lógica principal del bucle utilizado para analizar el conjunto total de datos:

```
1
2 N = 1000;
3 micbHighPass = fir1(N, 55/(fs/2), 'high');
4 accbHighPass = fir1(N, [20 2000]/(fs/2), 'bandpass', hamming(N+1));
5
6 data = load(matFilePath);
7
8 is_discarded = false;
9 discard_reason = '';
10
11 while true
12     % VERIFICACI N DE LA FRECUENCIA DE MUESTREO
13     if data.fs ~= fs
14         is_discarded = true;
15         discard_reason = 'Fs no es 20000';
16         break; % Salida temprana
17     end
18
19     % VERIFICACI N DE LA INTEGRIDAD DE SE ALES MIC Y ACC
20     if ~isfield(data, 'Mic') || ~isfield(data, 'ACC')
21         is_discarded = true;
22         discard_reason = 'Archivo no tiene MIC y/o ACC';
23         break;
24     end
25     if ~isValidSignal(data.Mic.Data)
26         is_discarded = true;
27         discard_reason = 'MIC: datos no válidos';
28         break;
29     end
30     if ~isValidSignal(data.ACC.Data)
31         is_discarded = true;
32         discard_reason = 'ACC: datos no válidos';
33         break;
```

```
    end

    % FILTRADO DE SE ALES
    mic = data.Mic.Data(:);
    acc = data.ACC.Data(:);
    filteredMIC = filtfilt(micbHighPass, 1, mic);

    % MIC puede venir con la polaridad cambiada,
    % utilizamos la función de COVAREP
    filteredMIC = filteredMIC * polarity_reskew(filteredMIC, fs);

    filteredACC = filtfilt(accbHighPass, 1, acc);
    total_duration_samples = length(filteredMIC);

    isVowelFile = ~strcmp(subDir, 'others');
    if isVowelFile
        [isValid, reason] = validateSustainedVowel(filteredMIC, fs,
    total_duration_samples, 'MIC');
        if ~isValid
            is_discarded = true;
            discard_reason = reason;
            break;
        end

        [isValid, reason] = validateSustainedVowel(filteredACC, fs,
    total_duration_samples, 'ACC');
        if ~isValid
            is_discarded = true;
            discard_reason = reason;
            break;
        end

        % --- PROCESAMIENTO SOLO PARA VOCALES ---

        % DETECCI N DE GCI (MIC)
        roiMIC = detectSpeech(filteredMIC(:), fs);
        voicedMIC = filteredMIC(roiMIC(1,1) : roiMIC(end,2));
        [F0s,~,~,~] = pitch_srh(voicedMIC,fs,60,400,32);
        F0 = median(F0s);
        [gcis, ~, ~] = gci_sedreams(voicedMIC, fs, F0, 1);
```

```
72        gcis = round(gcis * fs) + (roiMIC(1,1) - 1);
73
74        if ~isValidGCIs(gcis, fs)
75            is_discarded = true;
76            discard_reason = 'GCIs iniciales (sedreams) no válidos';
77            break;
78        end
79
80        % ESTIMACI N Y VALIDACI N DEL FLUJO GLOTAL (QCP)
81        [QCPGlottalFlow, dQCPGlottalFlow] = computeQCP(filteredMIC,
    gcis, fs);
82        Gb = fir1(N, [F0*0.8 2000]/(fs/2), 'bandpass', hamming(N+1));
83        QCPGlottalFlow = filtfilt(Gb, 1, QCPGlottalFlow);
84
85        [isValid, reason] = validateSustainedVowel(QCPGlottalFlow, fs,
    total_duration_samples, 'G');
86        if ~isValid
87            is_discarded = true;
88            discard_reason = reason;
89            break;
90        end
91
92        % DETECCI N DE GCI (DERIVADA DEL FLUJO)
93        dGb = fir1(N, [F0*0.8 3000]/(fs/2), 'bandpass', hamming(N+1));
94        dQCPGlottalFlow = filtfilt(dGb, 1, dQCPGlottalFlow);
95        roidG = detectSpeech(dQCPGlottalFlow(:), fs);
96        voiceddG = dQCPGlottalFlow(roidG(1,1) : roidG(end,2));
97        [dGgcis, ~] = getGCIs(voiceddG, fs);
98        dGgcis = dGgcis + (roidG(1,1) - 1);
99
100       if ~isValidGCIs(dGgcis, fs)
101           is_discarded = true;
102           discard_reason = 'GCIs(dQCP) no válidos';
103           break;
104       end
105
106       % ALINEACI N, NORMALIZACI N Y VALIDACI N FINAL
107       aligned_ACC = alignSignals(dQCPGlottalFlow, filteredACC, fs);
108       [G_norm, ACC_norm] = normalize_by_energy(QCPGlottalFlow,
    aligned_ACC, dGgcis);
```

```
109
110     end
111
112     break; % Si llegamos aquí, todas las validaciones pasaron. Salimos
        del bucle.
113
114 end % Fin del "while true"
```

El siguiente código contiene las funciones implementadas para validar las señales:

```
1  function valid = isValidSignal(x)
2      % --- Validaciones de Sanidad Básica de la Señal ---
3      % Comprueba tres condiciones fundamentales para que una señal sea
       procesable.
4
5      % 1. La señal no debe estar vacía.
6      % 2. Todos los valores deben ser finitos (no NaN o Inf).
7      % 3. La señal debe tener una longitud mínima (e.g., 20000 muestras
       ->1s)
8      %    para asegurar que hay suficiente información para un análisis
       robusto.
9      valid = ~isempty(x) && all(isfinite(x)) && length(x) > 20000;
10 end
11
12
13 function [isValid, reason] = validateSustainedVowel(signal, fs,
       total_samples, signalName)
14     isValid = false;
15
16     if ~isSustainedVowel(signal, fs, total_samples)
17         reason = sprintf('%s: no es una vocal sostenida', signalName);
18         return;
19     end
20
21     if ~checkPeriodicity(signal, fs)
22         reason = sprintf('%s: baja periodicidad', signalName);
23         return;
24     end
25
26     % Si llega aquí, pasó todas las pruebas
27     isValid = true;
```

```
    reason = '';
end

function valid = isSustainedVowel(x, fs, total_duration_samples)
    % --- Detección de Regiones de Voz ---
    % Se utiliza un detector de actividad de voz para encontrar segmentos hablados.
    % "MergeDistance" une segmentos de voz separados por silencios cortos (32 ms),
    % tratando la vocal como una única región continua.
    roi = detectSpeech(x(:), fs, "MergeDistance", 0.032*fs);

    % --- Validación de Unicidad de la Región ---
    % Una vocal sostenida debe ser detectada como una única región de interés (ROI).
    if size(roi, 1) ~= 1
        % Si se detectan cero o más de una región, no es una vocal sostenida.
        valid = false;
        return;
    end

    % --- Validación de Duración Relativa ---
    % Extraer las muestras de inicio y fin de la región de voz detectada
    start_sample = roi(1, 1);
    end_sample = roi(1, 2);

    % Calcular la duración en muestras de la región de voz
    roi_duration_samples = end_sample - start_sample;

    % Calcular qué porcentaje de la grabación total ocupa la región de voz
    percentage_of_signal = (roi_duration_samples / total_duration_samples) * 100;

    % Se requiere que la voz ocupe un porcentaje mínimo del total (e.g., 85%)
    % para asegurar que la grabación consiste principalmente en la vocal.
```

```
    required_percentage = 85;

    % Si la duración de la voz es muy corta en comparación con la grabación, se invalida.
    if percentage_of_signal < required_percentage
        valid = false;
        return;
    end

    valid = true;
end

function valid = checkPeriodicity(x, fs)
    % --- Aislamiento del Segmento de Interés ---
    % Detectar la región de voz para analizar solo la porción relevante de la señal
    roi = detectSpeech(x(:), fs);
    signal_start_sample = roi(1, 1);
    signal_end_sample = roi(1, 2);

    % Extraer el segmento de la señal que contiene la voz
    signal_segment = x(signal_start_sample:signal_end_sample);

    % --- Cálculo de Autocorrelación ---
    % Calcular la autocorrelación del segmento de voz.
    % 'coeff' normaliza los valores entre -1 y 1. Un valor alto indica alta similitud.
    [auto_corr, ~] = xcorr(signal_segment, 'coeff');
    % Encontrar el índice del peak central (lag = 0), que siempre es 1.
    [~, origin_idx] = max(auto_corr);

    % --- Búsqueda del peak de Periodicidad ---
    % Definir el rango de búsqueda para la F0 en muestras.
    % Esto corresponde a un rango de F0 vocal humana típica (ej. 60 Hz a 400 Hz).
    min_lag_samples = round(fs / 400); % Límite superior de F0
    max_lag_samples = round(fs / 60);  % Límite inferior de F0

    % Asegurar que el rango de búsqueda no exceda los límites del vector de autocorrelación
```

```
    end_of_search = origin_idx + max_lag_samples;
    if end_of_search > length(auto_corr)
        end_of_search = length(auto_corr);
    end

    % Definir el segmento de búsqueda después del peak de origen
    search_segment = auto_corr(origin_idx + min_lag_samples : end_of_search);

    % Si el segmento de búsqueda es vacío, la señal es muy corta para el análisis
    if isempty(search_segment)
        disp('El segmento de voz es demasiado corto para el análisis de periodicidad.');
        valid = false;
        return;
    end

    % --- Decisión de Validez ---
    % Encontrar la altura del peak más alto en el rango de búsqueda.
    % Esta altura es nuestro "puntaje de periodicidad".
    [main_peak_height, ~] = max(search_segment);

    % Definir un umbral de periodicidad. Un valor alto (ej. 0.7) indica una
    % señal fuertemente periódica, característica de una vocal sostenida.
    periodicity_threshold = 0.7;

    periodicity_score = main_peak_height;

    % Comparar el puntaje con el umbral para validar
    if periodicity_score < periodicity_threshold
        valid = false;
    else
        valid = true;
    end
end
```

```
129 function valid = isValidGCIs(gcis, fs)
130
131     % --- Parámetros de Validación ---
132     minGCIs = 70;           % Mínimo de GCIs para una señal estable
133     offset = 3;             % GCIs a ignorar en los bordes para evitar
    efectos de transición
134     thresholdFactor = 1.5;  % Factor para detectar periodos anómalos
    (1.5x la mediana)
135     minF0 = 70;             % F0 mínima esperada en Hz
136     maxF0 = 500;            % F0 máxima esperada en Hz
137
138     % --- Validaciones de Sanidad ---
139
140     % Se necesita un número mínimo de GCIs para procesar
141     if numel(gcis) < minGCIs
142         valid = false;
143         return;
144     end
145
146     % Chequeo para asegurar que se puede aplicar el offset
147     if numel(gcis) <= 2 * offset
148         valid = false;
149         return;
150     end
151
152     % --- Análisis de la Zona Estable ---
153
154     % Recortar gcis para analizar solo la porción estable de la voz
155     stableGCIs = gcis(1 + offset : end - offset);
156
157     % Deben quedar al menos 2 GCIs para poder calcular un periodo
158     if numel(stableGCIs) < 2
159         valid = false;
160         return;
161     end
162
163     % Calcular la distancia en muestras entre cada GCI
164     periods = diff(stableGCIs);
165
166     % --- Detección Robusta de GCIs Faltantes ---
```

```
167
168     % La mediana es más robusta a outliers que la media para el periodo
        de referencia
169     medianPeriod = median(periods);
170
171     % Un periodo no puede ser cero o negativo
172     if medianPeriod <= 0
173         valid = false;
174         return;
175     end
176
177     % Se define el umbral. Un GCI faltante genera un periodo ~2x la
       mediana
178     detectionThreshold = medianPeriod * thresholdFactor;
179
180     % Si CUALQUIER periodo supera este umbral, indica un GCI faltante
181     if any(periods > detectionThreshold)
182         valid = false;
183         return;
184     end
185
186     % --- Validación Final de F0 ---
187
188     % Se calcula la F0 a partir del periodo más estable (la mediana)
189     F0 = fs / medianPeriod;
190
191     % La F0 debe estar dentro de un rango vocal humano típeak
192     if F0 < minF0 || F0 > maxF0
193         valid = false;
194         return;
195     end
196
197     valid = true;
198 end
```

El siguiente código contiene la función para alinear las señales:

```
1 function aligned_y = alignSignals(x, y, fs)
2     % --- Parámetros de Alineación ---
3     % Máximo retardo permitido entre señales, en segundos (e.g., 20 ms)
      .
```

```
    % Se usa para limitar la búsqueda y evitar alineaciones erróneas.
    maxLag = 0.020;

    % --- Preprocesamiento de Señales ---
    % Asegurar que las señales de entrada sean vectores columna para consistencia
    xSeg = x(:);
    ySeg = y(:);

    % Normalizar ambas señales (media cero, desviación estándar uno).
    % Esto hace que la correlación cruzada sea insensible a la amplitud y al
    % nivel de DC de las señales, enfocándose solo en la similitud de la forma de onda.
    % 'eps' previene la división por cero si una señal es constante.
    xSeg = (xSeg - mean(xSeg)) ./ (std(xSeg) + eps);
    ySeg = (ySeg - mean(ySeg)) ./ (std(ySeg) + eps);

    % --- Cálculo de Correlación Cruzada ---
    % Convertir el retardo máximo de segundos a muestras
    maxLagSamp = round(maxLag * fs);

    % Calcular la correlación cruzada entre las dos señales.
    % 'coeff' normaliza el resultado para que el valor máximo sea 1.
    % Se busca el retardo ('lag') que maximiza la similitud entre ySeg y xSeg.
    [r, lags] = xcorr(ySeg, xSeg, maxLagSamp, 'coeff');

    % Encontrar el índice del máximo valor de correlación
    [~, iMax] = max(r);

    % Obtener el desfase en muestras correspondiente a la máxima correlación
    lagSamples = lags(iMax);

    % --- Aplicación del Desplazamiento ---
    % Aplicar un desplazamiento circular a la señal 'y' para alinearla con 'x'.
    % El signo negativo en 'lagSamples' corrige el desfase encontrado.
    aligned_y = circshift(y, -lagSamples);
```

```
end
```

El siguiente código contiene la función para obtener los GCIs de la derivada:

```
function [gcis, fdg_norm] = getGCIs(dG,fs)
    % --- Normalización de la Señal ---
    % Calcular la media de la señal de flujo glotal diferenciado (dG)
    media_fdg = mean(dG);
    % Calcular la desviación estándar de la señal
    std_fdg = std(dG);
    % Normalizar la señal (puntuación Z) para tener media 0 y desviación estándar 1
    fdg_norm = (dG - media_fdg) / std_fdg;
    % Invertir la señal para que los picos de cierre glotal (negativos) se vuelvan positivos
    fdg_invertida = -fdg_norm;

    % --- Detección de Picos (GCIs) ---
    % Definir la frecuencia fundamental máxima esperada para una voz humana (400 Hz)
    f0_max_esperada = 400;
    % Calcular la distancia mínima entre picos en muestras, basada en la F0 máxima
    distancia_minima_muestras = round(fs / f0_max_esperada);

    % Encontrar los picos en la señal invertida, que corresponden a los GCIs
    [~, gcis] = findpeaks(fdg_invertida, ...
        'MinPeakHeight', 0.5, ...                    % Establecer un umbral de altura para los picos
        'MinPeakDistance', distancia_minima_muestras); % Exigir una separación mínima entre ellos
end
```

El siguiente código contiene la función para estandarizar las señales para que cada ciclo tenga energía unitaria:

```
function [G_norm, acc_norm] = normalize_by_energy(G, acc, gci)
    % --- Preparación de Datos ---
    % Asegurar que ambas señales tengan la misma longitud para el procesamiento
    N = min(length(G), length(acc));
```

```
    glottalSignal_final = G(1:N);
    accSignal_final = acc(1:N);

    % Filtrar los GCIs para que solo se consideren los que están dentro de los límites de la señal
    gci_indices_final = gci(gci <= N);

    % Inicializar las señales normalizadas con ceros
    G_norm = zeros(size(glottalSignal_final));
    acc_norm = zeros(size(accSignal_final));

    % --- Normalización Ciclo a Ciclo ---
    % Iterar sobre cada ciclo glótico definido por los GCIs consecutivos
    for i = 1:(length(gci_indices_final) - 1)
        % Definir el inicio y fin de un ciclo glótico
        startIndex = gci_indices_final(i);
        endIndex = gci_indices_final(i+1) - 1;

        % Omitir el ciclo si los índices no son válidos (e.g., fin < inicio)
        if startIndex > endIndex
            continue;
        end

        % --- Normalización de la Señal Glótica ---
        % Extraer el segmento del ciclo actual de la señal glótica
        glottalCycle = glottalSignal_final(startIndex:endIndex);
        % Calcular la energía del ciclo (suma de cuadrados)
        cycleEnergy_g = sum(glottalCycle.^2);
        % El factor de normalización es la raíz cuadrada de la energía (norma L2)
        % 'eps' evita la división por cero si el ciclo tiene energía nula
        normFactor_g = sqrt(cycleEnergy_g) + eps;
        % Normalizar el ciclo y guardarlo en la señal de salida
        G_norm(startIndex:endIndex) = glottalCycle / normFactor_g;

        % --- Normalización de la Señal del Acelerómetro ---
        % Repetir el mismo proceso para la señal del acelerómetro
```

```
        accCycle = accSignal_final(startIndex:endIndex);
        cycleEnergy_a = sum(accCycle.^2);
        normFactor_a = sqrt(cycleEnergy_a) + eps;
        acc_norm(startIndex:endIndex) = accCycle / normFactor_a;
    end
end
```

A.2. Calculo de QCP

```
function [QCPGlottalFlow, dQCPGlottalFlow] = computeQCP(mic, gcis_cut, fs)
    % Parámetros fijos
    VTorder = 20;
    GSorder = 3;
    lipRad = 0.99;
    PQ = 0.05;
    RQ = 0.14; % Para construir la rampa, pero dentro de la función qcp_olaf se utiliza Nramp = 7 Y NO Se utiliza este factor
    STcompens = 0;
    causality = 'causal';

    % Calcular F0
    F0 = fs / median(diff(gcis_cut));

    % Ajustar DQ según F0
    if F0 < 190
        DQ = 0.95;
    elseif F0 < 280
        DQ = 0.55;
    else
        DQ = 0.7;
    end

    % Crear marcas temporales (en muestras)
    tmarks = make_tmarks(mic, fs, 32, 16);

    % Calcular QCP
    [QCPGlottalFlow, dQCPGlottalFlow, ~, ~] = qcp_olaf(mic, fs, ...
        VTorder, GSorder, lipRad, DQ, PQ, RQ, causality, STcompens, tmarks, gcis_cut);
```

```
end

function [G, Gd, H_VTA, H_G] = qcp_olaf(s, fs, VTorder, GSorder, lipRad
    , DQ, PQ, RQ, causality, STcompens, tmarks, gcis)
% Quasi Closed-Phase glottal inverse filtering with OLA processing
%
% Inputs:
%   s          - [1 N ] Speech signal (row vector)
%   fs         - Sampling frequency (Hz)
%   VTorder    - LPC order for vocal tract (e.g., 48)
%   GSorder    - LPC order for glottal source (e.g., 3)
%   lipRad     - Leaky integration coefficient (e.g., 0.99)
%   DQ         - Duration Quotient (0.4  1  .0)
%   PQ         - Position Quotient (0  0  .2)
%   RQ         - Ramp Quotient (e.g., 0.14)
%   causality - 'causal' or 'noncausal' for FIR filters
%   STcompens - Flag for spectral tilt compensation (true/false)
%   tmarks     - [2 Nf ] frame boundaries (samples)
%   gcis       - Glottal closure instants (sample indices)
%
% Outputs:
%   G     - [1 N ]   Reconstructed glottal flow signal
%   Gd    - [1 N ]   Glottal flow derivative signal
%   H_VTA - {1 Nf }  Cell array; each element is VT LPC coefficients [1
     VTorder ]
%   H_G   - {1 Nf }  Cell array; each element is GS LPC coefficients [1
     GSorder ]

% Ensure column vector becomes row
if iscolumn(s)
    s = s';
end

% Initialize outputs and filter memories
N = length(s);
G  = zeros(1, N);
Gd = zeros(1, N);
nf = size(tmarks, 2);
H_VTA = cell(1, nf);
H_G   = cell(1, nf);
```

```
66
67 % Frame processing time marks
68 tcenter = round((tmarks(1, :) + tmarks(2, :)) / 2);
69 tmc = round((tcenter(2:end) + tcenter(1:end-1)) / 2);
70 tmc = [tmarks(1, 1), tmc, tmarks(2, end)];
71
72 % Inverse filter memories
73 Z_vt   = zeros(1, VTorder);
74 Z_pre  = 0;
75 Z_int  = 0;
76 Z_st   = 0;
77 Z_der  = 0;
78
79 % Window function for LPC
80 winfunc = 'hanning';
81
82 wmin = 1e-5;
83
84 Lpf = VTorder + 1;
85
86 for n = 1:nf
87     % Extract frame for weighting
88     x = s(tmarks(1, n):tmarks(2, n));
89     pos_gci_frame = find((gcis >= tmarks(1, n)) & (gcis <= tmarks(2, n)
    ));
90     gci_ins = gcis(pos_gci_frame) - tmarks(1, n) + 1;
91     T0 = mean(diff(gci_ins));
92     Nramp = 7;
93     w = makeW(x, VTorder, DQ, PQ, wmin, Nramp, gci_ins, fs);
94
95     % Prepare signal segment for inverse filtering
96     if n == 1
97         segment = [linspace(-s(tmc(n)), s(tmc(n)), Lpf), s(tmc(n):(tmc(
    n+1)-1))];
98         idx = (Lpf+1):length(segment);
99     else
100         segment = s(tmc(n):(tmc(n+1)-1));
101         idx = 1:length(segment);
102     end
103
```

```
104     % Pre-emphasis
105     [s2, Z_pre] = filter([1 -1], 1, x, Z_pre);
106 
107     % Vocal tract LPC on windowed, weighted frame
108     sw = win(s2, winfunc);
109     [Hvt, ~] = wlp(sw,  VTorder, w);
110 
111     % Inverse filtering to get glottal source estimate
112     [dg, Z_vt] = filter(Hvt, 1, segment, Z_vt);
113     dg = dg(idx);
114     [sg, Z_int] = integrate(dg, lipRad, causality, Z_int,fs);
115 
116     % Glottal source LPC
117     Hg = lpc(win(sg, winfunc), GSorder);
118 
119     if STcompens
120         % Spectral tilt compensation
121         %[bvtt, avtt] = invfreqz(freqz(1, Hvt), 1024, 0, 1);
122         [bvtt, avtt] = invfreqz(freqz(1, Hvt), linspace(0, pi, 512), 0,
    1);
123         [sg, Z_st] = filter(bvtt, avtt, sg, Z_st);
124         [dg, Z_der] = filter([1 -lipRad], 1, sg, Z_der);
125         Hg = lpc(win(sg, winfunc), GSorder);
126     end
127 
128     H_VTA{n} = Hvt(:)';
129     H_G{n}   = Hg(:)';
130     G(tmc(n):(tmc(n+1)-1))  = sg;
131     Gd(tmc(n):(tmc(n+1)-1)) = dg;
132 end
133 end
134 
135 function y = win(x, winfunc)
136     y = feval(winfunc, length(x))' .* x;
137 end
138 
139 function [y, Z] = integrate(x, rho, causality, Z, fs)
140     if strcmp(causality,'causal')
141         % toda la señal de una vez; no por tramas
142         [y, Z] = filter(1, [1 -rho], x, Z);
```

```
143         y = y / ((1 - rho) * fs);
144     else
145         % offline, cero fase, sin estado
146         y = filtfilt(1, [1 -rho], x);
147         y = y / ((1 - rho)^2 * fs);
148         Z = [];    % no se usa
149     end
150 end
151 
152 function p = get_if_order(fs)
153     % Get suitable inverse filter order for given sampling frequency
154     p = round(fs / 1000) + 2;
155 end
156 
157 function w = makeW(x, p, DQ, PQ, d, Nramp, gci_ins, fs)
158     % Create AME weight function for frame x
159     N = length(x);
160     if Nramp > 0
161         UPramp = linspace(d, 1, 2 + Nramp);
162         UPramp = UPramp(2:end-1);
163         DOWNramp = UPramp(end:-1:1);
164     end
165 
166     if DQ + PQ > 1
167         DQ = 1 - PQ;
168     end
169 
170     Tfallback = median(diff(gci_ins));
171 
172     w = d * ones(1, N + p);
173     if isempty(gci_ins)
174         T2 = 0;
175         T1 = 1;
176     else
177         for i = 1:(length(gci_ins) - 1)
178             T = gci_ins(i+1) - gci_ins(i);
179             T1 = round(DQ * T);
180             T2 = round(PQ * T);
181             while T1 + T2 > T
182                 T1 = T1 - 1;
```

```
183             end
184             w(gci_ins(i) + T2 : gci_ins(i) + T2 + T1 - 1) = 1;
185             if Nramp > 0
186                 w(gci_ins(i) + T2 : gci_ins(i) + T2 + Nramp - 1) =
    UPramp;
187                 if gci_ins(i) + T2 + T1 - Nramp > 0
188                     w(gci_ins(i) + T2 + T1 - Nramp : gci_ins(i) + T2 +
    T1 - 1) = DOWNramp;
189                 end
190             end
191             Tlast = T;
192         end
193
194         if numel(gci_ins) >= 2
195             T = gci_ins(end) - gci_ins(end - 1);
196         elseif exist('Tlast','var')
197             T = Tlast;
198         else
199             T = Tfallback;
200         end
201
202         T1 = round(DQ * T);
203         T2 = round(PQ * T);
204         Nend = N - (T2 + gci_ins(end));
205         if T2 + gci_ins(end) < N
206             if T1 + T2 < Nend
207                  w(gci_ins(end) + T2 : gci_ins(end) + T2 + T1 - 1) = 1;
208                     if Nramp > 0
209                         w(gci_ins(end) + T2 : gci_ins(end) + T2 + Nramp
     - 1) = UPramp;
210                         if gci_ins(end) + T2 + T1 - Nramp > 0
211                             w(gci_ins(end) + T2 + T1 - Nramp : gci_ins(
    end) + T2 + T1 - 1) = DOWNramp;
212                         end
213                     end
214             else
215                 T1 = Nend - T2;
216                 w(gci_ins(end) + T2 : gci_ins(end) + T2 + T1 - 1) = 1;
217                 if Nramp > 0
218                     w(gci_ins(end) + T2 : gci_ins(end) + T2 + Nramp -
```

```
    1) = UPramp;
219                 end
220             end
221         end
222     end
223 end
224
225 function [A,w] = wlp(s,p,M)
226 % A = wlp(s,p,m,k)
227 % weighted linear prediction
228 % A - filter coefficients
229 % s - signal
230 % p - prediction order
231 % m - STE window length (scalar) or weighting function (vector)
232 % k - STE window lag (default 1)
233
234   % J.P. 120509, 180909, 100910
235
236 s = s(:);
237
238 s = s + eps;
239
240 N = length(s);
241
242 w = M(:);
243 if length(w)<N+p
244     error('Weighting function must have at least N+p values.');
245 end
246
247 wsr = sqrt(w(1:(N+p)));
248 Y = zeros(N+p,p+1);
249 for i1=0:p
250     Y(:,i1+1) = [zeros(i1,1);s;zeros(p-i1,1)].*wsr;
251 end
252
253 R = (Y'*Y)/N;
254
255 A = R(2:end,2:end)\R(2:end,1);
256 A = [1;-A]';
257 end
```

```

function tmarks = make_tmarks(s, fs, frame_len_ms, frame_shift_ms)
    frame_len = round(frame_len_ms / 1000 * fs);
    frame_shift = round(frame_shift_ms / 1000 * fs);

    N = length(s);
    starts = 1:frame_shift:(N - frame_len + 1);
    ends = starts + frame_len - 1;

    tmarks = [starts; ends];
end
```

A.3. Preparación de datos

El siguiente código contiene las funciones utilizadas para preparar los datos del entrenamiento:

```
def compute_delete_probabilities(subject_counts, power=2, min_clip=1e-4):
    """
    Calcula probabilidades de eliminación no lineales para cada sujeto.
    - power > 1: penaliza más a sujetos con más muestras.
    - min_clip: probabilidad mínima para que nadie quede con cero.

    Returns: dict subject -> delete probability
    """
    # Normalizamos primero
    normalized = subject_counts / subject_counts.sum()

    # Elevamos al power para dar más peso a los grandes
    weighted = normalized ** power

    # Re-normalizamos para que sumen 1
    delete_probs = weighted / weighted.sum()

    # Clip mínimo para evitar probabilidad cero
    delete_probs = delete_probs.clip(lower=min_clip)

    return delete_probs.to_dict()
```

```
22
23 def make_dataset(cfg, split_by_subject: bool = True) -> tuple[list,
    list, list]:
24     """Aplica los filtros, hace split y devuelve tres listas de tuplas
    (absolute_path, source_path)."""
25     df = pd.read_csv(cfg.index_csv)
26
27     # Crear las mascaras para luego filtrar df
28     m = pd.Series(True, index=df.index)
29     if cfg.subject_filter:
30         m &= df.subject.isin(pd.Series(cfg.subject_filter).to_numpy())
31     if cfg.utterance_filter:
32         m &= df.utterance.isin(cfg.utterance_filter)
33     if cfg.gender_filter:
34         m &= df.gender.isin(cfg.gender_filter)
35     if cfg.condition_filter:
36         m &= df.condition.isin(cfg.condition_filter)
37     if cfg.phonation_filter:
38         m &= df.phonationType.isin(cfg.phonation_filter)
39
40     df_filtrado = df[m].reset_index(drop=True)
41     df_filtrado = df_filtrado[(df_filtrado['useQCP'] == 1)].copy()
42
43     # Asignar probabilidades a los sujetos para quitar samples hasta
    llegar a los minutos requeridos
44     subject_counts = df_filtrado['subject'].value_counts()
45     subject_weights = compute_delete_probabilities(subject_counts,
    power=1.4)
46     df_filtrado['delete_prob'] = df_filtrado['subject'].map(
    subject_weights)
47
48     seconds_wanted = cfg.minutes_wanted * 60
49     while df_filtrado['duration_sec'].sum() > seconds_wanted:
50         # Normalizar probabilidades para las filas restantes
51         probs = df_filtrado['delete_prob'].values
52         probs = probs / probs.sum()
53
54         idx_to_remove = np.random.choice(df_filtrado.index, p=probs)
55         df_filtrado = df_filtrado.drop(idx_to_remove)
56
```

```
    if cfg.split_by_subject:
        train_df, val_df, test_df = split_dataset_by_subject(
            df_filtrado, split_ratios=cfg.split_ratios, seed=cfg.seed
        )
    else:
        raise NotImplementedError("La lógica para split_dataset sin sujeto debe ser actualizada.")

    base = Path(cfg.base_path)

    def create_path_tuples(df_split):
        path_tuples = []
        for _, row in df_split.iterrows():
            abs_path = str(base / row['filePath'])
            if Path(abs_path).exists():
                path_tuples.append((abs_path, row['sourcePath']))
        return path_tuples

    train_data = create_path_tuples(train_df)
    val_data   = create_path_tuples(val_df)
    test_data  = create_path_tuples(test_df)

    return train_data, val_data, test_data

def split_dataset_by_subject(
    df: pd.DataFrame,
    split_ratios=(0.7, 0.15, 0.15),
    seed=2025
) -> tuple[pd.DataFrame, pd.DataFrame, pd.DataFrame]:
    """
    Recibe un DataFrame y devuelve tres DataFrames (train, val, test),
    asegurando que un sujeto completo quede solo en uno de los splits.
    """
    train_ratio, val_ratio, test_ratio = split_ratios

    # 1ra división: test vs resto
    gss = GroupShuffleSplit(n_splits=1, test_size=test_ratio, random_state=seed)
    train_val_idx, test_idx = next(gss.split(df, groups=df.subject))
    df_train_val = df.iloc[train_val_idx]
```

```
95      df_test      = df.iloc[test_idx]
96
97      # 2da división: train vs val sobre el resto
98      val_size = val_ratio / (train_ratio + val_ratio)
99      gss2 = GroupShuffleSplit(n_splits=1, test_size=val_size,
        random_state=seed + 1)
100     train_idx, val_idx = next(gss2.split(df_train_val, groups=
        df_train_val.subject))
101
102     # CAMBIO: Devolver los DataFrames
103     df_train = df_train_val.iloc[train_idx]
104     df_val   = df_train_val.iloc[val_idx]
105
106     return df_train, df_val, df_test
107
108 def build_indexes_with_filter(file_data, k, n_cycles=10, mad_k=3.0, rng
        =None):
109     accepted = []
110
111     for abs_path, source_path in file_data:
112
113         mat = loadmat(abs_path, simplify_cells=True)
114         x_arr = mat['acc'].flatten().astype('float32')
115
116         L = len(x_arr)
117         start = int(0.05 * L)
118         end   = int(0.95 * L)
119
120         for i in range(start, end - k + 1, k):
121             accepted.append((abs_path, i, source_path))
122
123     return accepted
124
125
126 def compute_stats_from_indexes(indexes, k,
127                                scaler_kind_x='zscore',
128                                scaler_kind_y=None,
129                                feature_range=(-1, 1)):
130     def _make_scaler(kind, feature_range):
131         if kind == 'zscore':
```

```
            return StandardScaler()
        elif kind == 'minmax':
            return MinMaxScaler(feature_range=feature_range)
        else:
            raise ValueError(f"Unsupported scaler kind: {kind}")

    scaler_x = _make_scaler(scaler_kind_x, feature_range)
    scaler_y = _make_scaler(scaler_kind_y, feature_range) if
   scaler_kind_y else None

    from collections import defaultdict
    by_fp = defaultdict(list)
    for fp, i, _ in indexes:
        by_fp[fp].append(i)

    for fp, indices in by_fp.items():
        mat = loadmat(fp, simplify_cells=True)
        x_arr = mat['acc'].flatten().astype('float32')
        y_arr = mat['G'].flatten().astype('float32')

        for i in indices:
            seg_x = x_arr[i : i + k]
            scaler_x.partial_fit(seg_x.reshape(-1, 1))

            if scaler_y:
                seg_y = y_arr[i : i + k]
                scaler_y.partial_fit(seg_y.reshape(-1, 1))

    return scaler_x, scaler_y
```

A.4. Implementación de la TCN

El siguiente código contiene la implementación de la red TCN utilizada:

```
class TCNBlock(nn.Module):
    def __init__(self, channels, dilation, p_drop, kernel_size):
        super().__init__()
        self.net = nn.Sequential(
            # First dilated convolution
```

```
            nn.Conv1d(channels, channels, kernel_size=kernel_size,
    padding='same', dilation=dilation),
            nn.GroupNorm(num_groups=1, num_channels=channels),
            nn.ReLU(),
            nn.Dropout(p_drop),

            # Second dilated convolution
            nn.Conv1d(channels, channels, kernel_size=kernel_size,
    padding='same', dilation=dilation),
            nn.GroupNorm(num_groups=1, num_channels=channels),
            nn.ReLU(),
            nn.Dropout(p_drop),
        )

    def forward(self, x):
        return self.net(x)


class TCN(nn.Module):
    def __init__(self, name, n_blocks=12, hidden=64, p_drop=0.005,
    kernel_size=5):
        super().__init__()
        self.skip = nn.Conv1d(1, hidden, 1)
        dilations = [2**i for i in range(n_blocks)]
        self.blocks = nn.ModuleList([TCNBlock(hidden, d, p_drop,
    kernel_size) for d in dilations])

        self.fc = nn.Conv1d(hidden, 1, 1)

    def forward(self, x):                    # x: (B,1,K)
        h = self.skip(x)
        for blk in self.blocks:
            h = blk(h) + h                   # residual
        return self.fc(h)                    # (B,1,K)
```

Referencias

[1] Paavo Alku. Glottal inverse filtering analysis of human voice production - a review of estimation and parameterization methods of the glottal excitation and their applications. *Sadhana*, 36(5):623–650, October 2011.

[2] Gunnar Fant. *Acoustic theory of speech production: with calculations based on X-ray studies of Russian articulations*. Number 2. Walter de Gruyter, 1971.

[3] Sudarsana Reddy Kadiri and Paavo Alku. Analysis and detection of pathological voice using glottal source features. *IEEE Journal of Selected Topics in Signal Processing*, 14(2):367–379, 2020.

[4] Daryush D. Mehta, Jarrad H. Van Stan, Matías Zañartu, Mohammad Ghassemi, John V. Guttag, Victor M. Espinoza, Juan P. Cortés, II Cheyne, Harley A., and Robert E. Hillman. Using ambulatory voice monitoring to investigate common voice disorders: Research update. *Frontiers in Bioengineering and Biotechnology*, 3:155, 2015.

[5] Thomas Drugman, Paavo Alku, Abeer Alwan, and Bayya Yegnanarayana. Glottal source processing: From analysis to applications. *Computer Speech Language*, 28(5):1117–1138, 2014.

[6] Matías Zañartu, Julio C. Ho, Daryush D. Mehta, Robert E. Hillman, and George R. Wodicka. Subglottal impedance-based inverse filtering of voiced sounds using neck surface acceleration. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(9):1929–1939, 2013.

[7] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[8] J.R. Deller, J.G. Proakis, and J.H.L. Hansen. *Discrete-time Processing of Speech Signals*. AnIEEE press classic reissue. Macmillan Publishing Company, 1993.

[9] Zhaoyan Zhang. Mechanics of human voice production and control. *The Journal of the Acoustical Society of America*, 140(4):2614, 2016.

[10] Laura L. Koenig, Jorge C. Lucero, and Elizabeth Perlman. Speech production variability in fricatives of children and adults: results of functional data analysis. *The Journal of the Acoustical Society of America*, 124(5):3158–3170, 2008.

[11] Andrew Ma, Kenneth K Lau, and Dominic Thyagarajan. Voice changes in parkinson's disease: What are they telling us? *Journal of Clinical Neuroscience*, 72:1–7, 2020.

[12] Sudarsana Reddy Kadiri and Alku. Glottal features for classification of phonation type from speech and neck surface accelerometer signals. *Computer Speech & Language*, 70:101232, 2021.

[13] Jan Gauffin and Johan Sundberg. Spectral correlates of glottal voice source waveform characteristics. *Journal of Speech, Language, and Hearing Research*, 32(3):556–565, 1989.

[14] Sudarsana Reddy Kadiri, RaviShankar Prasad, and B. Yegnanarayana. Detection of glottal closure instant and glottal open region from speech signals using spectral flatness measure. *Speech Communication*, 116:30–43, 2020.

[15] Yu-Ren Chien, Daryush D. Mehta, Jón Guðnason, Matías Zañartu, and Thomas F. Quatieri. Evaluation of glottal inverse filtering algorithms using a physiologically based articulatory speech synthesizer. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(8):1718–1730, 2017.

[16] Kevin Russell. Acoustic phonetics. https://home.cc.umanitoba.ca/~krussll/phonetics/acoustic/formants.html, 2024. Accessed: 2025-06-03.

[17] Hyung-Suk Kim. Linear predictive coding is all-pole resonance modeling. https://ccrma.stanford.edu/~hskim08/lpc/, 2020. Accessed: 2025-06-03.

[18] Tom Bäckström, Okko Räsänen, Abraham Zewoudie, Pablo Pérez Zarazaga, Liisa Koivusalo, Sneha Das, Esteban Gómez Mellado, Marieum Bouafif Mansali, Daniel Ramos, Sudarsana Kadiri, Paavo Alku, and Mohammad Hassan Vali. *Introduction to Speech Processing*. 2 edition, 2022.

[19] Paavo Alku. Parameterisation methods of the glottal flow estimated by inverse filtering. In Christina d'Alessandro and Klaus R. Scherer, editors, *Voice Quality: Functions, Analysis and Synthesis (VOQUAL'03)*. ISCA Archive, August 2003.

[20] D. Wong, J. Markel, and A. Gray. Least squares glottal inverse filtering from the acoustic speech waveform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 27(4):350–355, 1979.

[21] Paavo Alku. Glottal wave analysis with pitch synchronous iterative adaptive inverse filtering. *Speech Communication*, 11(2):109–118, 1992. Eurospeech '91.

[22] Manu Airaksinen, Tuomo Raitio, Brad Story, and Paavo Alku. Quasi closed phase glottal inverse filtering analysis with weighted linear prediction. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(3):596–607, 2014.

[23] Thomas Drugman, Baris Bozkurt, and Thierry Dutoit. Causal–anticausal decomposition of speech using complex cepstrum for glottal source estimation. *Speech Communication*, 53(6):855–866, 2011.

[24] Peter Birkholz. Modeling consonant-vowel coarticulation for articulatory speech synthesis. *PLoS ONE*, 8(4):e60603, 2013.

[25] Marc Freixes, Luis Joglar-Ongay, Joan Claudi Socoró, and Francesc Alías-Pujol. Evaluation of glottal inverse filtering techniques on openglot synthetic male and female vowels. *Applied Sciences*, 13(15), 2023.

[26] Paavo Alku, Tiina Murtola, Jarmo Malinen, Juha Kuortti, Brad Story, Manu Airaksinen, Mika Salmi, Erkki Vilkman, and Ahmed Geneid. Openglot – an open environment for the evaluation of glottal inverse filtering. *Speech Communication*, 107:38–47, 2019.

[27] Thomas Drugman, Baris Bozkurt, and Thierry Dutoit. A comparative study of glottal source estimation techniques. *Computer Speech & Language*, 26(1):20–34, 2012.

[28] Ingo Langheinrich, Simon Stone, Xinyu Zhang, and Peter Birkholz. Glottal inverse filtering based on articulatory synthesis and deep learning. pages 1327–1331, 09 2022.

[29] Harold Cheyne. Estimating glottal voicing source characteristics by measuring and modeling the acceleration of the skin on the neck. *The Journal of the Acoustical Society of America*, 112:2445–2446, 11 2002.

[30] Steven M. Lulich, Jennifer R. Morton, Harish Arsikere, Michael S. Sommers, Gina K. Leung, and Abeer Alwan. Subglottal resonances of adult male and female native speakers of american english. *J Acoust Soc Am*, 132(4):2592–2602, Oct 2012.

[31] Joaquín Sepúlveda, Jesús A. Parra, Emiro J. Ibarra, Mauricio Araya, Patricio De La Cuadra, and Matías Zañartu. Estimation of physiological vocal features from neck surface acceleration signals using probabilistic bayesian neural networks. *IEEE Transactions on Audio, Speech and Language Processing*, 33:1576–1589, 2025.

[32] IBM. What are neural networks? https://www.ibm.com/think/topics/neural-networks, 2025. Accessed: 2025-08-31.

[33] Amazon Web Services (AWS). What is a neural network? https://aws.amazon.com/what-is/neural-network/, 2023. Accessed on 2 de septiembre de 2025.

[34] IBM. Deep learning. https://www.ibm.com/think/topics/deep-learning, 2025. Accessed: 2025-08-31.

[35] Anna Zou and Zhiyuan Li. Interpretability of neural network with physiological mechanisms, 2022.

[36] N.P. Narendra, Manu Airaksinen, and Paavo Alku. Glottal source estimation from coded telephone speech using a deep neural network. In *Interspeech 2017*, pages 3931–3935, 2017.

[37] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[38] MathWorks. Long short-term memory (lstm). https://la.mathworks.com/discovery/lstm.html, 2025. Accessed: 2025-09-05.

[39] Francesco Lässig. Temporal convolutional networks and forecasting. https://unit8.com/resources/temporal-convolutional-networks-and-forecasting/, 2021. Accessed: 2025-09-05.

[40] Josep Ferrer. How transformers work: A detailed exploration of transformer architecture. https://www.datacamp.com/tutorial/how-transformers-work, 2024. Accessed: 2025-09-05.

[41] Thomas Drugman, Mark Thomas, Jon Gudnason, Patrick Naylor, and Thierry Dutoit. Detection of glottal closure instants from speech signals: A quantitative review. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(3):994–1006, 2012.

[42] Thomas Drugman and Thierry Dutoit. Glottal Closure and Opening Instant Detection from Speech Signals. In *Proc. Interspeech 2009*, pages 2891–2894, 2009.

[43] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *CoRR*, abs/1803.01271, 2018.

[44] Yuzhen Zhu, Shaojie Luo, Di Huang, Weiyan Zheng, Fang Su, and Beiping Hou. Drcnn: decomposing residual convolutional neural networks for time series forecasting. *Scientific Reports*, 13(15901), 2023.

[45] Ryan Turner, David Eriksson, Michael McCourt, Juha Kiili, Eero Laaksonen, Zhen Xu, and Isabelle Guyon. Bayesian optimization is superior to random search for machine learning hyperparameter tuning: Analysis of the black-box optimization challenge 2020. *CoRR*, abs/2104.10201, 2021.

[46] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

[47] G. Degottex, J. Kane, T. Drugman, T. Raitio, and S. Scherer. Covarep - a collaborative voice analysis repository for speech technologies. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Florence, Italy, 2014.

[48] jopoh. acoustic-feature-extraction: Python scripts for acoustic feature extraction. https://github.com/jopoh/acoustic-feature-extraction, 2020. GitHub repository.

[49] Boris Doval, Christophe d'Alessandro, and Nathalie Henrich Bernardoni. The spectrum of glottal flow models. *Acta Acustica united with Acustica*, 92:1026–1046, 11 2006.

[50] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. *CoRR*, abs/1912.01703, 2019.

[51] William Falcon and The PyTorch Lightning team. PyTorch Lightning. `https://github.com/Lightning-AI/lightning`, March 2019.

[52] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv:1803.01271*, 2018.

[53] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. *CoRR*, abs/1907.10902, 2019.