

Introducción a las Tecnologías del Habla

Dr. Iván López-Espejo

DÍA 1: INTRODUCCIÓN, APLICACIONES, Y ANÁLISIS DE LA SEÑAL DE VOZ

iloes@ugr.es

Monday 2nd June, 2025



PONTIFICIA UNIVERSIDAD
CATOLICA
DE VALPARAISO



UNIVERSIDAD
DE GRANADA

Tabla de Contenidos

- 1 Introducción y Breve Recorrido Histórico
- 2 Aplicaciones de las Tecnologías del Habla
- 3 Análisis de la Señal de Voz

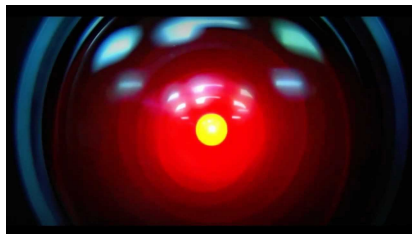
Tabla de Contenidos

- 1 Introducción y Breve Recorrido Histórico
- 2 Aplicaciones de las Tecnologías del Habla
- 3 Análisis de la Señal de Voz

Introducción y Breve Recorrido Histórico

- Imitar el comportamiento humano en una máquina, *sobre todo la capacidad de hablar y responder al lenguaje oral*, ha intrigado a ingenieros y científicos durante siglos
- El progreso de las tecnologías de la voz ha estado fuertemente ligado al del **reconocimiento automático del habla (RAH)** así como al de la **síntesis de voz**

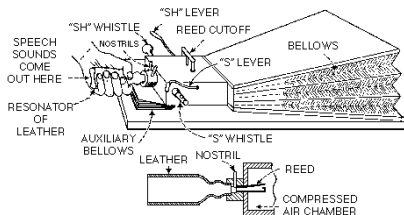
2001: Una odisea del espacio
(Stanley Kubrick, 1968)



HAL 9000

Introducción y Breve Recorrido Histórico

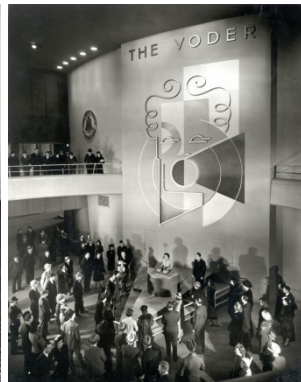
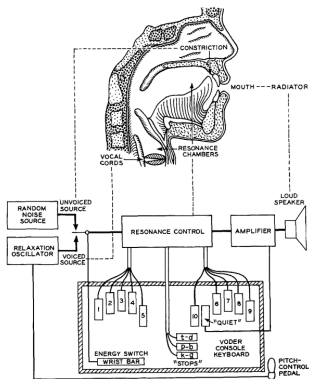
- El desarrollo de máquinas para imitar la comunicación verbal humana parece haber empezado en la segunda mitad del siglo XVIII:
 - 1773: El científico ruso Christian Kratzenstein produjo sonidos vocales usando tubos de resonancia conectados a tubos de órgano
 - 1791: Wolfgang von Kempelen construyó una máquina de habla mecanicoacústica
 - ~1850: Charles Wheatstone fabricó una versión de la máquina de von Kempelen usando resonadores hechos de cuero



[Flanagan72] J. L. Flanagan, "Speech Analysis, Synthesis and Perception". Springer-Verlag, 1972

Introducción y Breve Recorrido Histórico

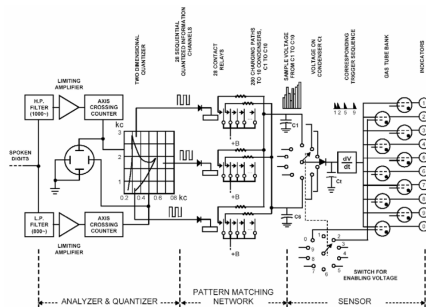
- **1922:** En los laboratorios Bell (Fletcher) se documenta la **relación entre el espectro de voz y las características del habla** (incluyendo inteligibilidad)
- **1930:** Homer Dudley desarrolla un **synthesizador de voz** llamado **VODER** → *Equivalente eléctrico de la máquina de Wheatstone*



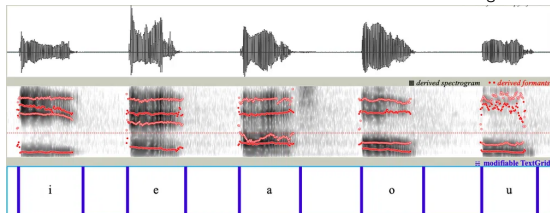
[Dudley39] H. Dudley *et al.*, "A synthetic speaker". Journal of the Franklin Institute, 1939

Introducción y Breve Recorrido Histórico

- **1952**: Se desarrolla en los laboratorios Bell **Audrey**, un sistema monolutor de reconocimiento de dígitos aislados a partir de frecuencias formantes de vocales (*reconocimiento de patrones*)



[Davis52] K. H. Davis *et al.*, "Automatic Recognition of Spoken Digits". The Journal of the Acoustical Society of America, 1952



Introducción y Breve Recorrido Histórico

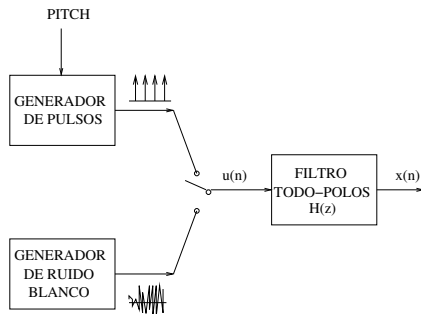
- **1962:** Sakai y Doshita desarrollan un reconocedor de fonemas^[Sakai62]
- Implementación del *primer segmentador de habla* para el análisis y reconocimiento del habla en distintas porciones del enunciado de entrada
- **¡Precursor de un sistema de reconocimiento continuo de voz!**



[Sakai62] J. Sakai and S. Doshita, "The Phonetic Typewriter". In Proc. of IFIP 1962

Introducción y Breve Recorrido Histórico

- **Finales de 1960:** Atal e Itakura formularon independientemente los fundamentos de la **codificación predictiva lineal (LPC, Linear Predictive Coding)**
 - Simplificación de la estimación de la respuesta del tracto vocal
- **Mediados de 1970:** Itakura, Rabiner y Levinson, y otros, propusieron combinar **reconocimiento de patrones y LPC**



Introducción y Breve Recorrido Histórico



- **Tangora:** Sistema de reconocimiento **dependiente del locutor** (*dictado*)
- Foco en conseguir un gran tamaño de vocabulario
- **Modelo de lenguaje:** Conjunto de reglas sintácticas o gramáticas estadísticas (1980: *N*-grama)
- Foco en el diseño de un sistema **independiente del locutor** (*marcación por voz*)
- Precursor del uso de **modelos ocultos de Márkov (HMMs)** para la representación de unidades lingüísticas (fonemas, sílabas, palabras...)
- Mayor énfasis en **modelado acústico** y **detección de palabras clave**

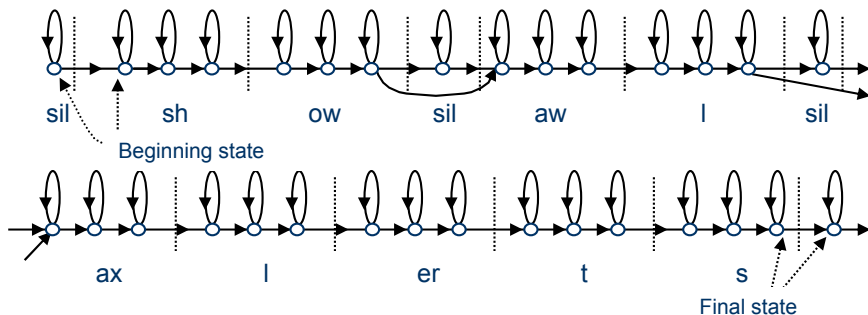


"Every time I fire a linguist, the performance of the speech recognizer goes up"

Frederick Jelinek (1932 – 2010)

Introducción y Breve Recorrido Histórico

- A principios de 1980, en **Bell Labs**, la teoría de los HMMs se extendió al uso de densidades de mezcla

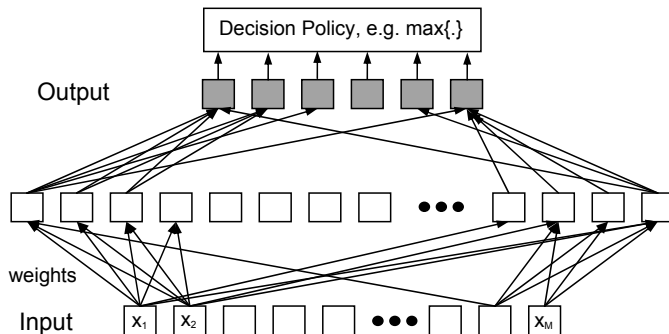


"Show all alerts" modeled as phones: ϕ -sh-ow, ϕ -ax-l, ax-l-er, l-er-t

[Juang04] B. H. Juang and L. R. Rabiner, "Automatic Speech Recognition – A Brief History of the Technology Development"

Introducción y Breve Recorrido Histórico

- *Reintroducción* a **finales de 1980** de las **redes neuronales artificiales**
 - **Procesamiento distribuido en paralelo**
 - Invención del algoritmo de **retropropagación** (*Geoffrey Hinton*)

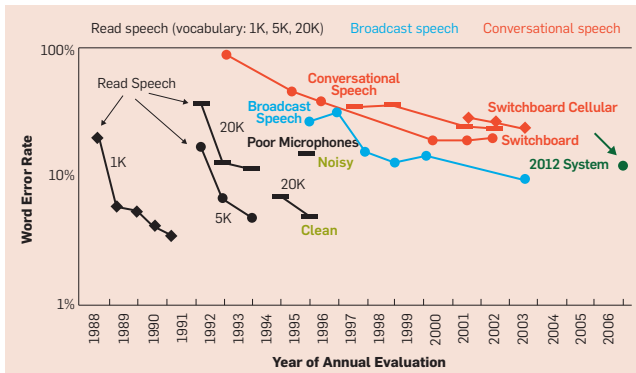


[Juang04] B. H. Juang and L. R. Rabiner, "Automatic Speech Recognition – A Brief History of the Technology Development"

¡Un perceptrón multicapa es un aproximador universal!

Introducción y Breve Recorrido Histórico

- Con el apoyo de **DARPA** (*Defense Advanced Research Projects Agency*), se impulsó el **testeo de tecnología de RAH** para diversas tareas y vocabularios

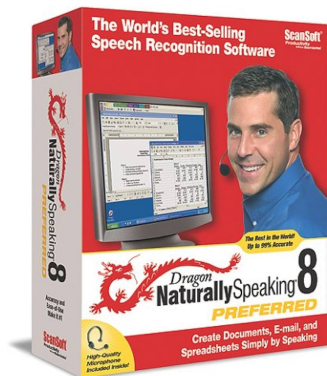


[Huang14] X. Huang *et al.*, "A Historical Perspective of Speech Recognition". Communications of the ACM, 2014

- 1 Dificultad de reconocer habla conversacional
- 2 ¡Cuanto más datos de entrenamiento, mejor!

Introducción y Breve Recorrido Histórico

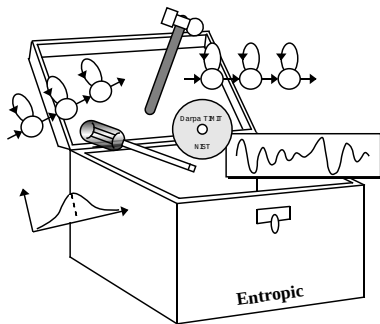
- 1990: Dragon lanza **Dragon Dictate**, el primer producto de reconocimiento de voz para el público general



Introducción y Breve Recorrido Histórico



**Hidden Markov Model Toolkit
(HTK)**



[Young02] S. Young *et al.*, "The HTK Book"



**UNIVERSITY OF
CAMBRIDGE**
Department of Engineering

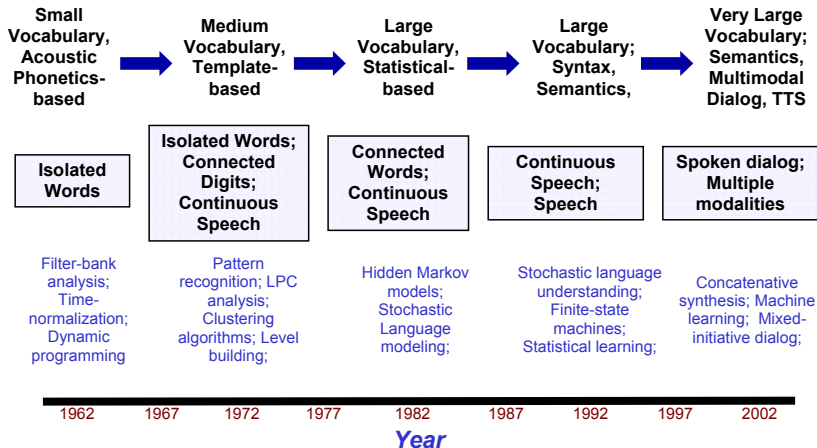


Steve Young

Gunnar Evermann
Dan Kershaw
Gareth Moore
Julian Odell
Dave Ollason
Dan Povey
Valtcho Valtchev
Phil Woodland

Introducción y Breve Recorrido Histórico

Milestones in Speech and Multimodal Technology Research



[Juang04] B. H. Juang and L. R. Rabiner, "Automatic Speech Recognition – A Brief History of the Technology Development"

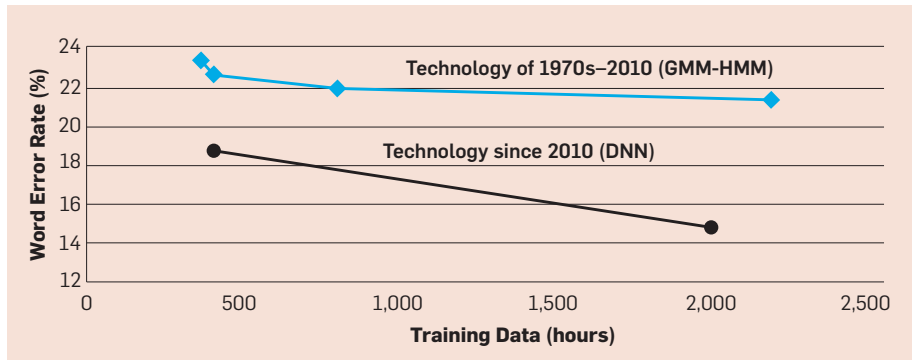
Introducción y Breve Recorrido Histórico

- *Cambio de paradigma a partir de 2010:* **aprendizaje profundo**



- **Datos, datos y más datos**
- **Recursos computacionales**

Introducción y Breve Recorrido Histórico



[Huang14] X. Huang *et al.*, "A Historical Perspective of Speech Recognition". Communications of the ACM, 2014

Introducción y Breve Recorrido Histórico



- **2008:** Google lanza la app Voice Search para iPhone



Hey Siri

- **2011:** Apple anuncia **Siri** (*no sólo reconoce voz, sino que la entiende y actúa en consecuencia*)



Hey Cortana

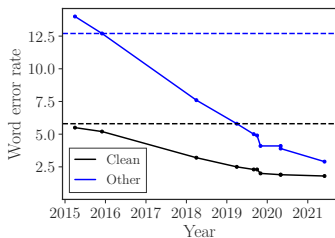
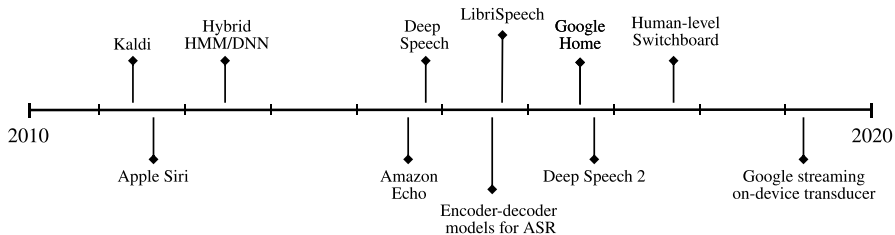
- **2014:** Microsoft anuncia **Cortana**, un asistente personal similar a Siri



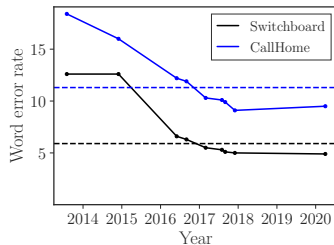
amazon alexa

- **2014:** Amazon anuncia Echo, un altavoz controlado por voz donde corre el asistente **Alexa**

Introducción y Breve Recorrido Histórico



(a) LibriSpeech

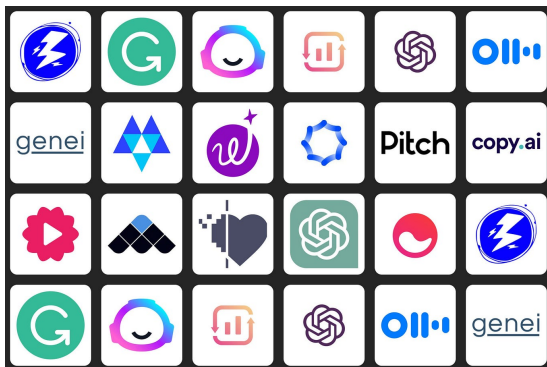


(b) Switchboard Hub5'00

[Hannun21] A. Hannun, "The History of Speech Recognition to the Year 2030". arXiv, 2021

Introducción y Breve Recorrido Histórico

- *Hacia una inteligencia artificial general...*



- 1 **Arquitectura Transformer**
(mecanismo de atención)
- 2 **Pre-entrenamiento predictivo (LLM)**
- 3 **Ajuste fino**
- 4 **RLHF** (*Reinforcement Learning from Human Feedback*)

- **Procesamiento del lenguaje natural (LLMs) + tecnologías del habla**

Introducción y Breve Recorrido Histórico

*“¡Es una locura! Estamos hablando de un enorme aumento de la **productividad**, lo que debería significar más bienes y servicios para todos, y por tanto, que todos estemos mejor. Pero, en realidad, ¡va a ser todo lo contrario! Y esto es porque **vivimos en una sociedad capitalista**. Lo que va a ocurrir es que este inmenso aumento de la productividad generará mucho más dinero para las grandes empresas y los ricos, y **aumentará la brecha entre los ricos y los que pierden sus empleos**.*

*Y tan pronto como esa brecha se agranda, se crea un terreno fértil para el fascismo. Así que... es realmente aterrador que estemos llegando a un punto en el que, en lugar de mejorar las cosas, las estamos empeorando cada vez más. Es una locura, porque **estamos haciendo algo que debería beneficiar a todos**. Obviamente, ayudará en la atención sanitaria y en la educación, pero si los beneficios sólo van a parar a los ricos, eso va a empeorar la sociedad.”*



Geoffrey Hinton
Premio Nobel de Física 2024

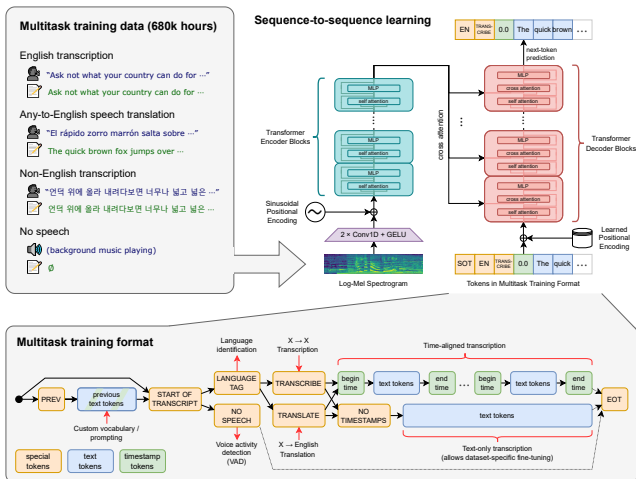
Tabla de Contenidos

1 Introducción y Breve Recorrido Histórico

2 Aplicaciones de las Tecnologías del Habla

3 Análisis de la Señal de Voz

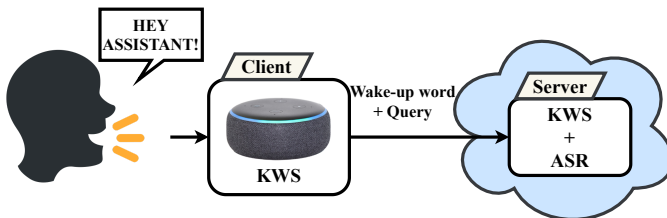
Reconocimiento Automático del Habla (RAH)



[Radford23] A. Radford *et al.*, “Robust Speech Recognition via Large-Scale Weak Supervision”. In Proc. of ICML 2023

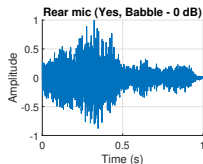
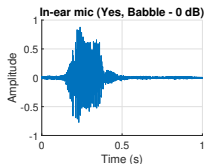
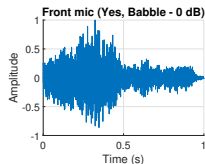
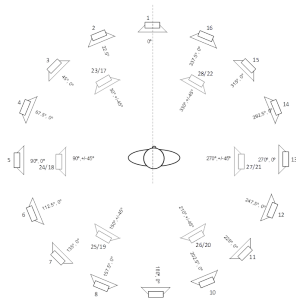
Detección de Palabras Clave

- La **detección de palabras clave (KWS)** se puede definir como la tarea de identificar palabras clave en flujos de audio que contienen VOZ
- Aplicaciones de KWS:** Minería de datos de voz, indexación de audio, direccionamiento de llamadas telefónicas, etc.



[Espejo21] I. López-Espejo *et al.*, "Deep Spoken Keyword Spotting: An Overview". IEEE Access, 2021

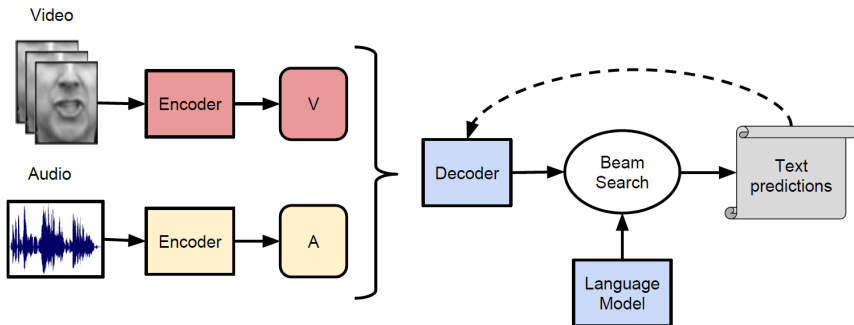
Control por Voz de Audífonos Robusto al Ruido



[Espeja24] I. López-Espejo *et al.*, "Noise-Robust Hearing Aid Voice Control". IEEE Signal Processing Letters, 2024

RAH Audiovisual

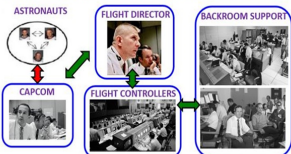
- **Aplicaciones de la lectura labial:** Hablar a un teléfono inteligente en entornos altamente ruidosos, transcribir y doblar cintas mudas de archivo, *mejorar el rendimiento del RAH en general...*
- **Cadena de procesamiento del RAH audiovisual:**



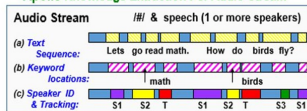
[Afouras18] T. Afouras *et al.*, "Deep Audio-Visual Speech Recognition". IEEE Transactions on Pattern Analysis and Machine Intelligence, 2018

Identificación de Temas en Grabaciones de la NASA

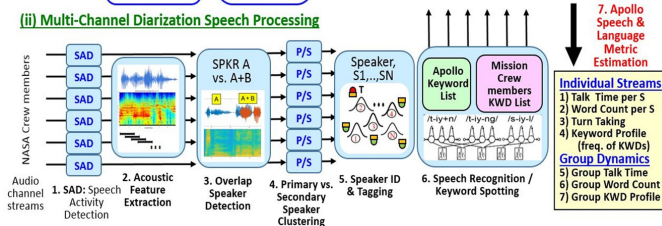
(i) Communication Overview for Apollo-MCC



(iii) CRSS-UTD DIARIZATION OUTPUT: Apollo Knowledge Extraction Per Audio Stream



(ii) Multi-Channel Diarization Speech Processing



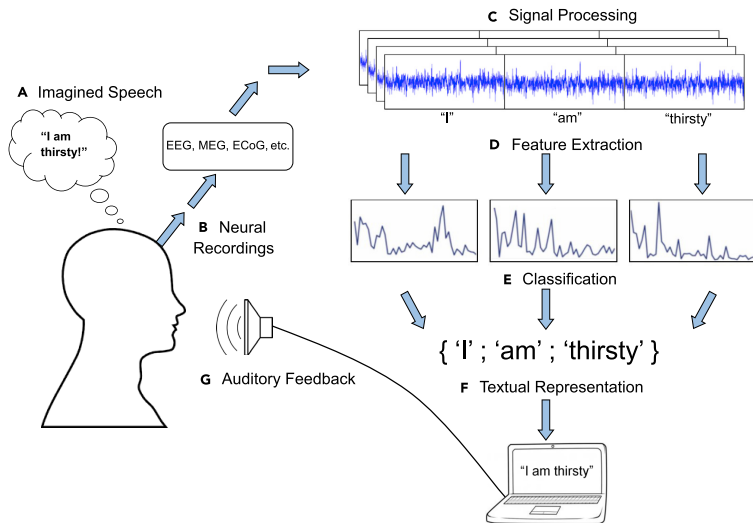
[Joglekar23] A. Joglekar *et al.*, "Fearless Steps APOLLO: Challenges in keyword spotting and topic detection for naturalistic audio streams". The Journal of the Acoustical Society of America, 2023

AGILE-KWS: A Giant Leap for Keyword Spotting

Comisión Europea, Becas Globales Marie Curie (HORIZON-MSCA-2021-PF-01)



Decodificación del Habla Imaginada

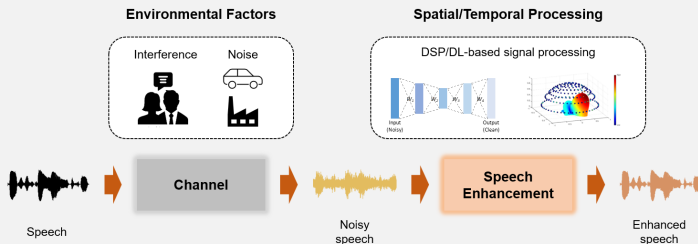


[Cooney18] C. Cooney *et al.*, "Neurolinguistics Research Advancing Development of a Direct-Speech Brain-Computer Interface".
Elsevier iScience, 2018

Realce de la Voz

- El **realce de la voz** persigue optimizar la **calidad** e **inteligibilidad** del habla eliminando ruido e interferencias no deseadas
- Importante procedimiento para mejorar el rendimiento de la comunicación hablada y la calidad de los contenidos multimedia

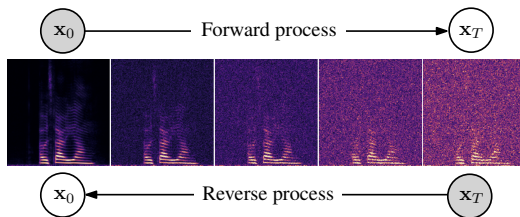
Enhancement Process



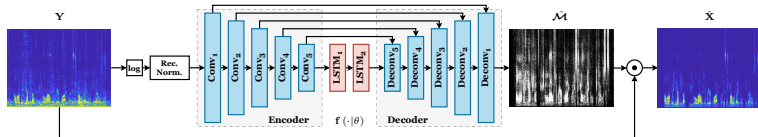
[DSPAILab] DSP & AI Lab, <http://dsp.yonsei.ac.kr/research/speech-enhancement/>

Realce de la Voz

- Aproximaciones **discriminativas** (p. ej., **FullSubNet+**) vs. **generativas** (p. ej., **SGMSE+**)

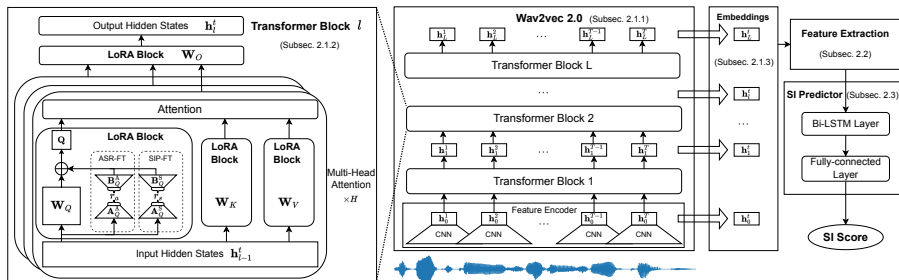


[Richter23] J. Richter *et al.*, "Speech Enhancement and Dereverberation With Diffusion-Based Generative Models". IEEE/ACM Transactions on Audio, Speech, and Language Processing, 2023



[Espejo24b] I. López-Espejo *et al.*, "On Speech Pre-emphasis as a Simple and Inexpensive Method to Boost Speech Enhancement". In Proc. of IberSPEECH 2024

Predicción de la Inteligibilidad de la Voz

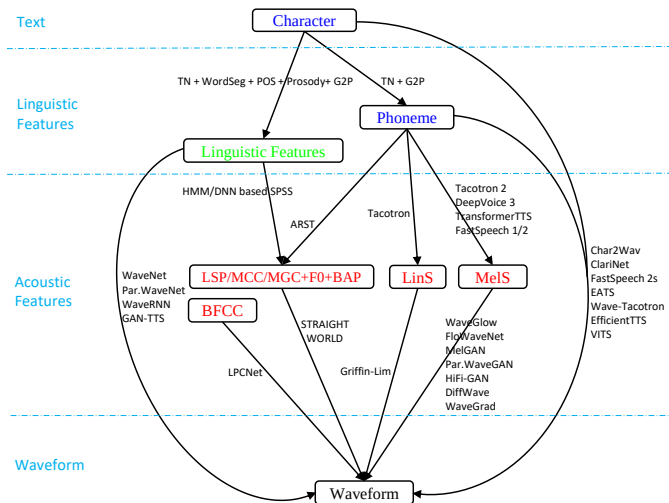


[Wang24] H. Wang *et al.*, "No-Reference Speech Intelligibility Prediction Leveraging a Noisy-Speech ASR Pre-Trained Model". In Proc. of Interspeech 2024

● Aprendizaje auto-supervisado de representaciones del habla

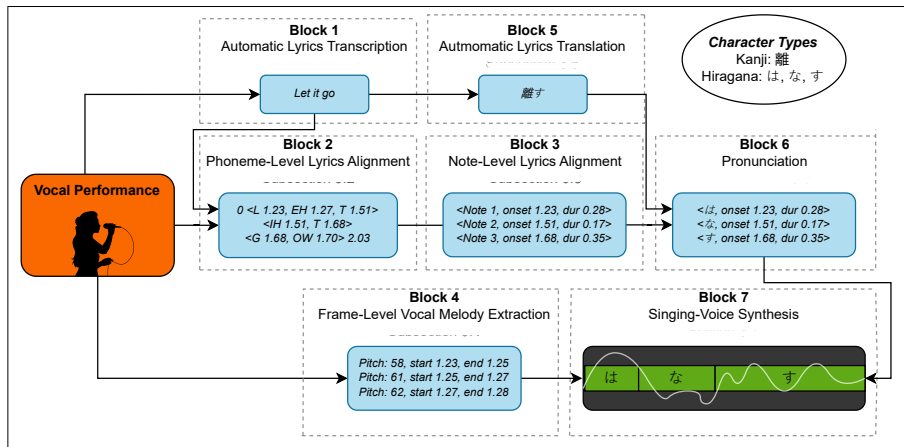
- 1 Wav2vec 2.0 (🧐 https://huggingface.co/docs/transformers/model_doc/wav2vec2)
- 2 HuBERT (🧐 https://huggingface.co/docs/transformers/model_doc/hubert)
- 3 WavLM (🧐 https://huggingface.co/docs/transformers/model_doc/wavlm)

Síntesis de Voz



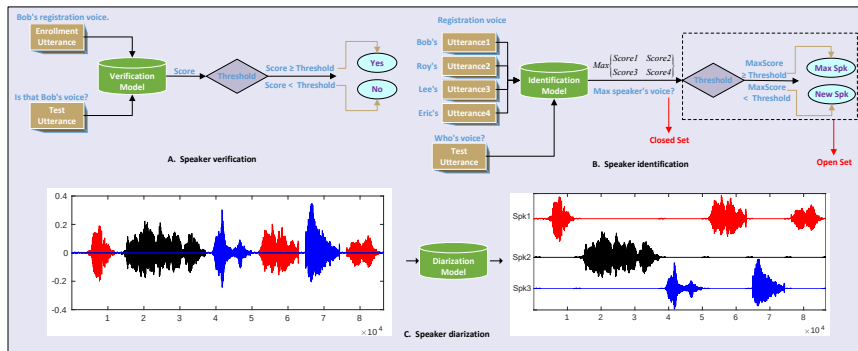
[Tan21] X. Tan et al., "A Survey on Neural Speech Synthesis". arXiv, 2021

Traducción Automática de Voz Cantada a Voz Cantada



[Antonisen24] S. Antonisen and I. López-Espejo, "PolySinger: Singing-Voice to Singing-Voice Translation from English to Japanese". In Proc. of ISMIR 2024

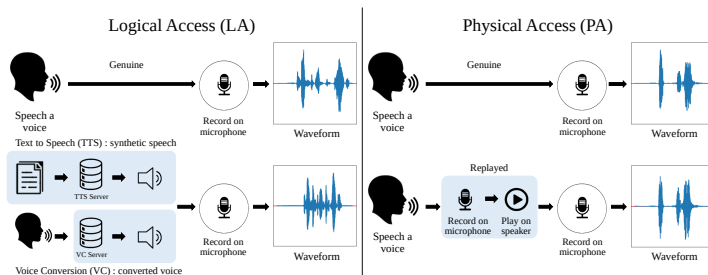
Verificación/Reconocimiento de Hablantes



[Bai21] Z. Bai and X.-L. Zhang, "Speaker recognition based on deep learning: An overview". Elsevier Neural Networks, 2021

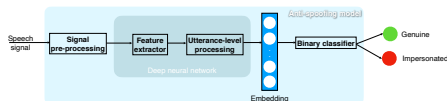
- **Verificación (1:1)** vs. **identificación/reconocimiento (1:N)**
- **Diarización:** ¿En qué momento habló quién?

Detección de Suplantación por Voz



[Lee23] Y. Lee *et al.*, "Experimental Case Study of Self-Supervised Learning for Voice Spoofing Detection". IEEE Access, 2023

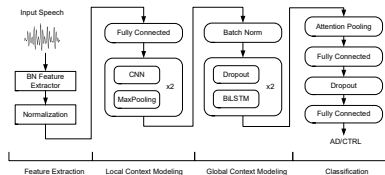
- **Ataques de acceso lógico:** Síntesis de voz, conversión de voz (*deepfakes*)
- **Ataques de acceso físico:** Imitación, repetición/reproducción



[Gomez24] A. M. Gomez *et al.*, "Signal and Neural Processing against Spoofing Attacks and Deepfakes for Secure Voice Interaction (ASASVI)". In Proc. of IberSPEECH 2024

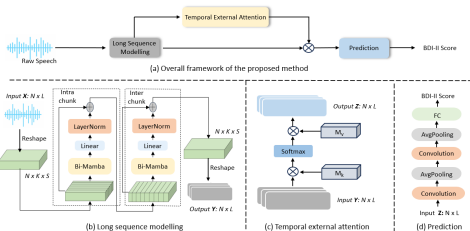
Diagnóstico de Enfermedades

- Detección de la enfermedad de Alzheimer



[Liu21] Z. Liu *et al.*, "Detecting Alzheimer's Disease from Speech Using NNs with Bottleneck Features and DA". In Proc. of ICASSP 2021

- Detección de la depresión

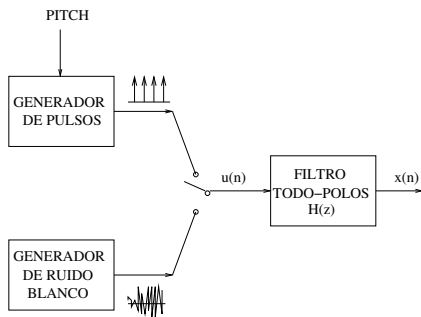


[Li25] S. Li *et al.*, "Efficient Long Speech Sequence Modelling for Time-Domain Depression Level Estimation". In Proc. of ICASSP 2025

Tabla de Contenidos

- 1 Introducción y Breve Recorrido Histórico
- 2 Aplicaciones de las Tecnologías del Habla
- 3 Análisis de la Señal de Voz**

Modelo LPC de Producción de Voz

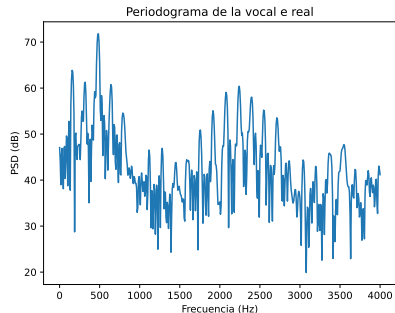
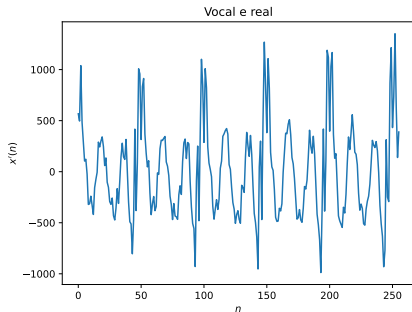


- **Sonidos sordos (/s/, /f/, ...):** Excitación tipo ruido blanco (flujo de aire procedente de los pulmones) – $AR(p)$
- **Sonidos sonoros (/a/, /m/, /b/, ...):** Excitación tipo tren de impulsos (vibración de las cuerdas vocales) – Todo-polos

- **Frecuencia de pitch:** Frecuencia de vibración de las cuerdas vocales

Modelo LPC de Producción de Voz

● Ejemplo de vocal española 'e':

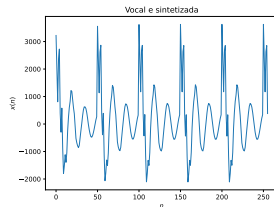
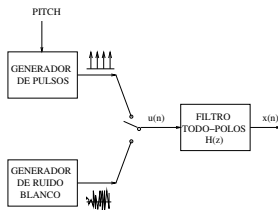
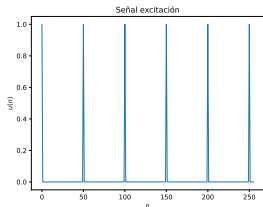


● Frecuencia de muestreo de 8.000 Hz

● Dos componentes diferenciadas:

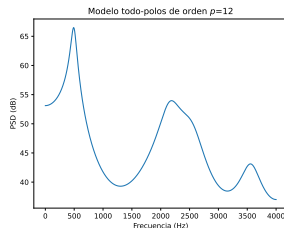
- 1 **Formantes:** Debidos al tracto vocal (correlaciones cortas)
- 2 **Pitch:** Debido a la vibración de las cuerdas vocales (correlaciones largas)

Modelo LPC de Producción de Voz



- Se estiman los parámetros $\{a_1, \dots, a_p, b_0\}$ mediante las ecuaciones de Yule-Walker

$$P_{LPC}(\omega) = \frac{1}{N} |H(e^{j\omega})|^2 = \frac{|b_0|^2 / N}{\left| 1 + \sum_{k=1}^p a_k e^{-j\omega k} \right|^2}$$



$p = 12$

Análisis Homomórfico

- **Análisis homomórfico:** Nos permite *separar* la información relativa al tracto vocal de la excitación
- Pasos:
 - 1 Se aplica el logaritmo a una representación espectral de la señal de voz
 - 2 Se aplica la transformada inversa de Fourier al resultado del paso anterior
- **Resultado:** Nueva secuencia temporal llamada **cepstrum** que es función de la **frecuencia**

$$P_x(\omega) = |\mathcal{F}\{x(n) = h(n) * u(n)\}|^2 = |H(\omega)|^2 P_u(\omega)$$

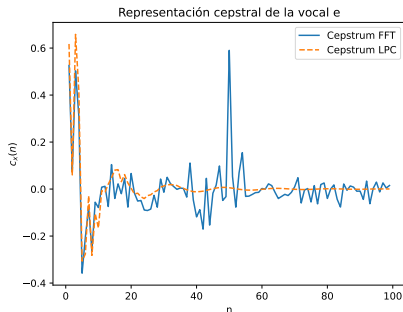
$$\log(P_x(\omega)) = \log(|H(\omega)|^2) + \log(P_u(\omega))$$

$$c_x(n) = \mathcal{F}^{-1}\{\log(P_x(\omega))\} = \mathcal{F}^{-1}\{\log(|H(\omega)|^2)\} + \mathcal{F}^{-1}\{\log(P_u(\omega))\}$$

$$c_x(n) = c_h(n) + c_u(n)$$

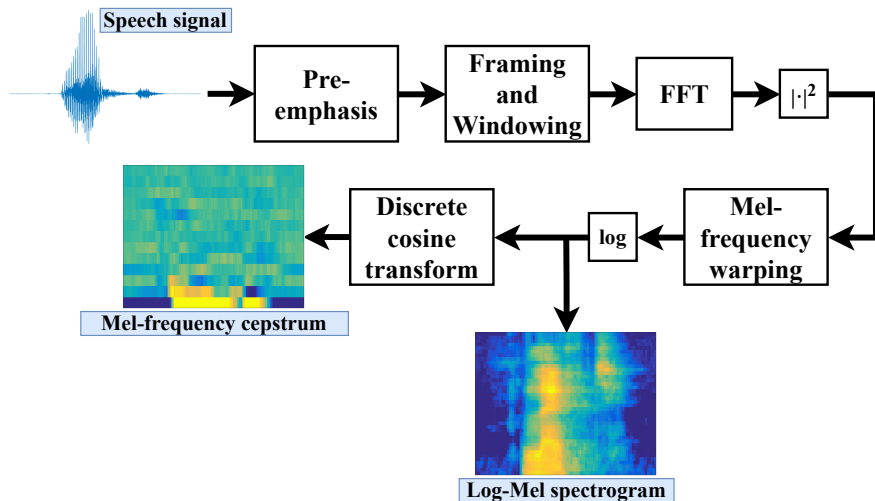
Análisis Homomórfico

- $c_x(n) = c_h(n) + c_u(n)$:
 - ① $c_h(n)$: Componente cepstral de **baja cuelfrencia** debida al **tracto vocal/filtro**
 - ② $c_u(n)$: Componente cepstral de **alta cuelfrencia** debida al **pitch**



- $c_h(n)$ es discriminativo de la información fonética
- $c_x(0)$ suele descartarse (*relacionado con la energía de la señal*)

Características Relacionadas con la Escala Mel



[Espejo21] I. López-Espejo *et al.*, "Deep Spoken Keyword Spotting: An Overview". IEEE Access, 2021

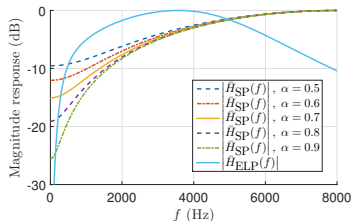
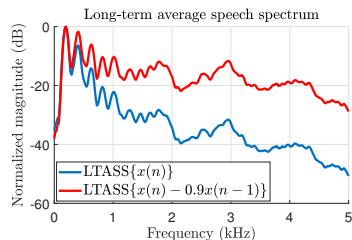
Características Relacionadas con la Escala Mel

Pre-énfasis

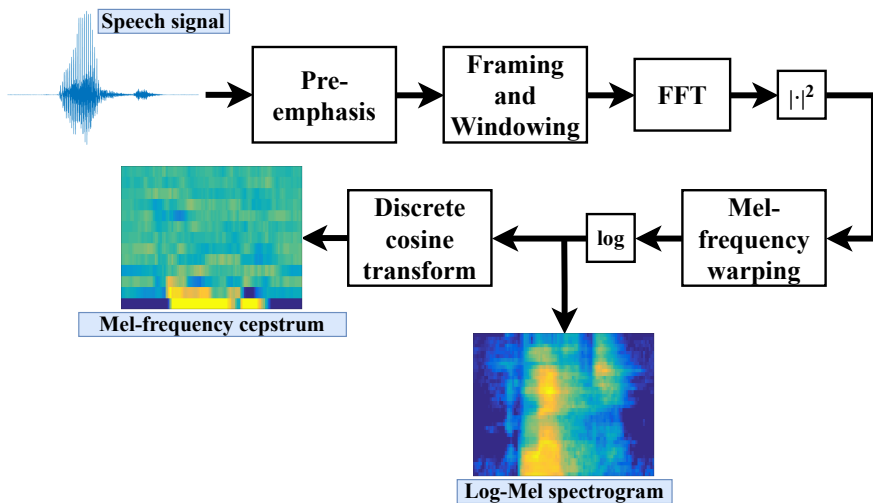
- El habla se caracteriza por una **inclinación espectral** originada en la excitación glotal debida a la vibración de las cuerdas vocales
- Inclinación espectral** $\rightarrow$ Los sistemas de procesamiento del habla podrían “pasar por alto” las frecuencias más altas
- El **filtrado de pre-énfasis** es un pre-procesamiento simple pero efectivo que compensa las componentes de alta frecuencia *aplanando el espectro del habla*

$$|H(f)| = |1 - \alpha e^{-j2\pi f/f_s}| = \sqrt{\alpha^2 - 2\alpha \cos(2\pi f/f_s) + 1}$$

Filtro FIR paso-alto de primer orden



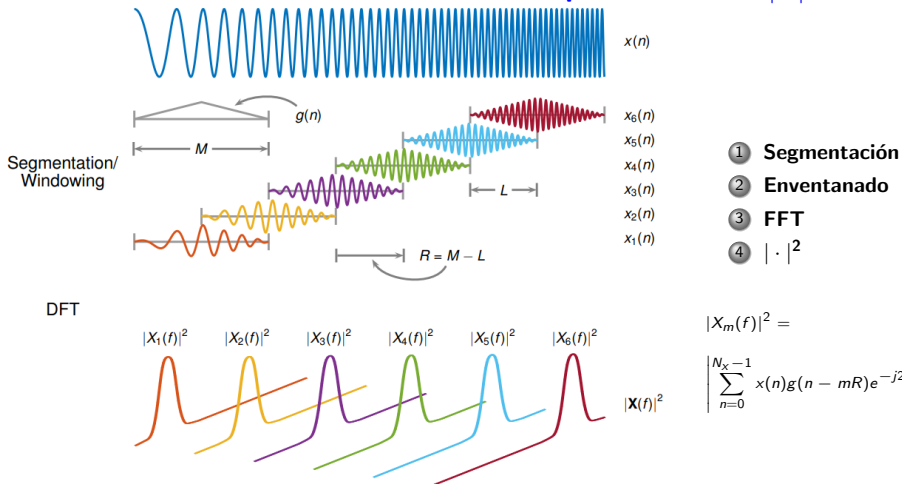
Características Relacionadas con la Escala Mel



[Espejo21] I. López-Espejo *et al.*, "Deep Spoken Keyword Spotting: An Overview". IEEE Access, 2021

Características Relacionadas con la Escala Mel

Transformada discreta de Fourier de tiempo reducido + $|\cdot|^2$

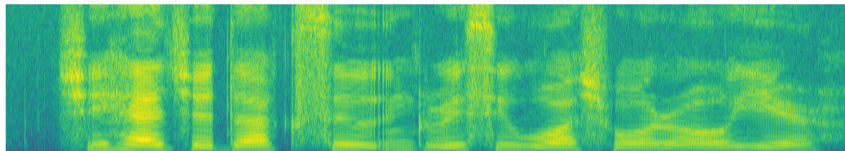


[MATLABstft] MATLAB Help Center, "stft".

<https://www.mathworks.com/help/signal/ref/stft.html>

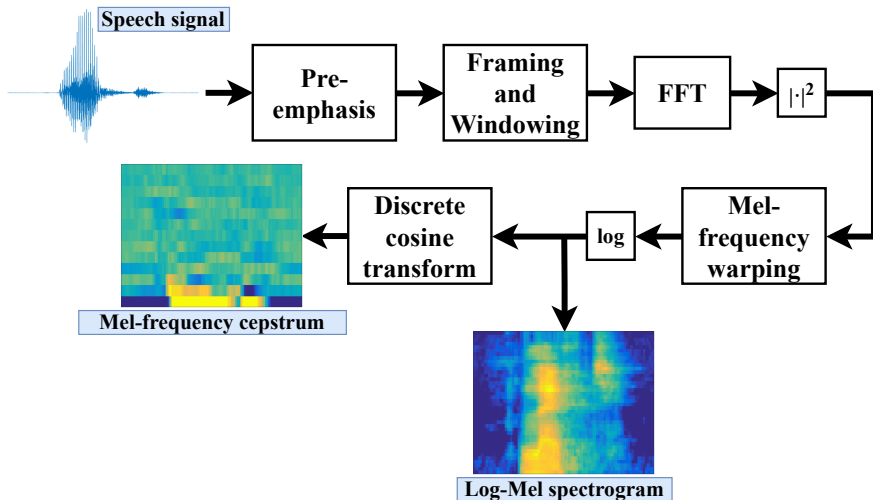
Características Relacionadas con la Escala Mel

- Ejemplo de cálculo de **espectrograma**:



[Tang24] H. Tang, "Speech Signal Analysis 2". Automatic Speech Recognition—ASR Lecture 3, 2024

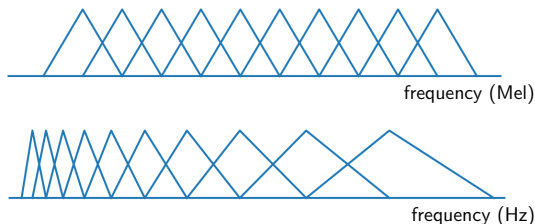
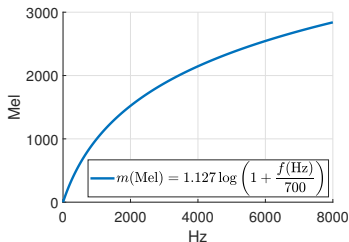
Características Relacionadas con la Escala Mel



[Espejo21] I. López-Espejo *et al.*, "Deep Spoken Keyword Spotting: An Overview". IEEE Access, 2021

Características Relacionadas con la Escala Mel

Aplicación de banco de filtros Mel



[Tang24] H. Tang, "Speech Signal Analysis 2". Automatic Speech Recognition—ASR Lecture 3, 2024

$$\begin{pmatrix} Y_m(k_1) \\ Y_m(k_2) \\ \vdots \\ Y_m(k_K) \end{pmatrix} = \begin{pmatrix} H_1(f_1) & H_1(f_2) & \cdots & H_1(f_F) \\ H_2(f_1) & H_2(f_2) & \cdots & H_2(f_F) \\ \vdots & \vdots & \ddots & \vdots \\ H_K(f_1) & H_K(f_2) & \cdots & H_K(f_F) \end{pmatrix} \begin{pmatrix} |X_m(f_1)|^2 \\ |X_m(f_2)|^2 \\ \vdots \\ |X_m(f_F)|^2 \end{pmatrix}, \quad \forall m \in [1, T]$$

T tramas temporales, F bins de frecuencia lineal, K filtros

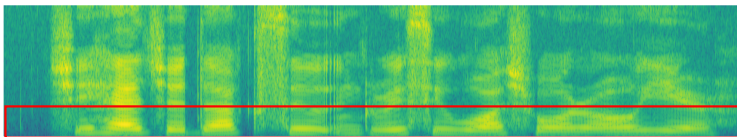
$H_k(f)$: k -ésimo filtro del banco de filtros Mel ($k \in [1, K]$)

$Y_m(k)$: m -ésima trama temporal del espectrograma Mel

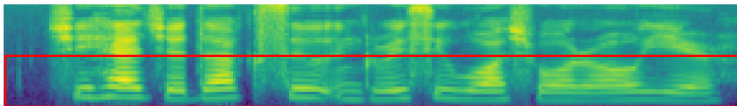
Características Relacionadas con la Escala Mel

- Ejemplo de cálculo de **espectrograma Mel** a partir de espectrograma lineal:

linear spectrogram

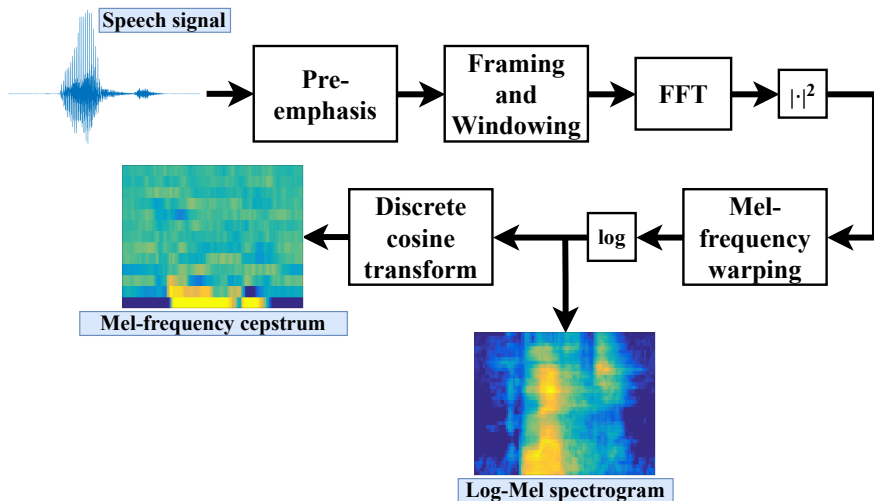


Mel spectrogram



[Tang24] H. Tang, "Speech Signal Analysis 2". Automatic Speech Recognition—ASR Lecture 3, 2024

Características Relacionadas con la Escala Mel



[Espejo21] I. López-Espejo *et al.*, "Deep Spoken Keyword Spotting: An Overview". IEEE Access, 2021

Características Relacionadas con la Escala Mel

Cálculo de los coeficientes cepstrales de frecuencia Mel (MFCCs)

- Se trata de aplicar la **transformada discreta del coseno (DCT)** a nuestro espectrograma log-Mel ($\forall m \in [1, T]$):

$$\begin{pmatrix} Z_m(q_1) \\ Z_m(q_2) \\ \vdots \\ Z_m(q_Q) \end{pmatrix} = \text{Espectrograma log-Mel} \cdot \begin{pmatrix} Y_m(k_1) \\ Y_m(k_2) \\ \vdots \\ Y_m(k_K) \end{pmatrix}$$

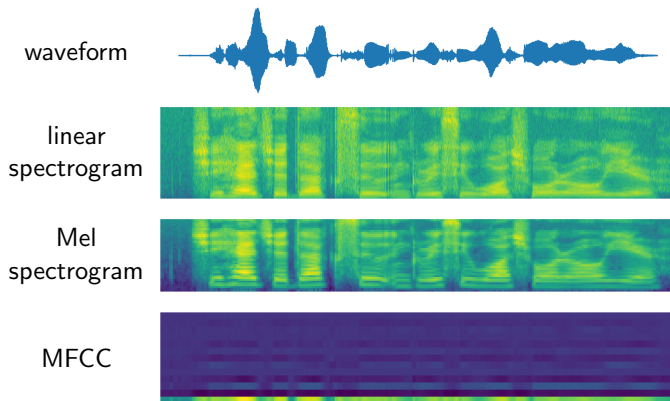
$$C(\nu, \ell) = \kappa_\nu \cos \left(\frac{(2\ell + 1)\nu\pi}{2K} \right), \quad \nu = 1, \dots, Q, \quad \ell = 1, \dots, K$$

$$\kappa_\nu = \begin{cases} \sqrt{1/K} & \text{si } \nu = 0 \\ \sqrt{2/K} & \text{en otro caso} \end{cases}$$

Q bins de frecuencia, K filtros, $Q < K$

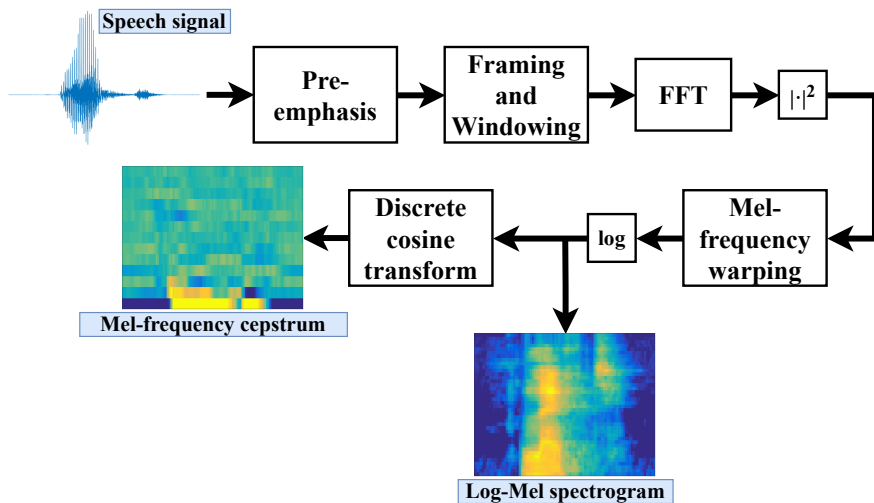
Características Relacionadas con la Escala Mel

- Ejemplo de cálculo de **MFCCs**:



[Tang24] H. Tang, "Speech Signal Analysis 2". Automatic Speech Recognition—ASR Lecture 3, 2024

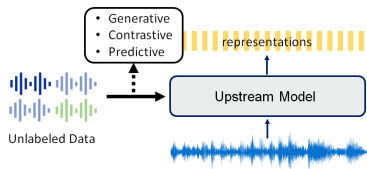
Características Relacionadas con la Escala Mel



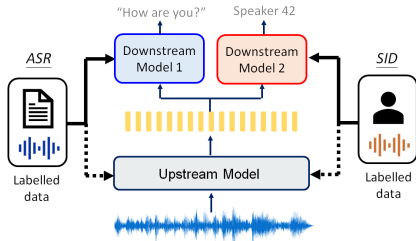
[Espejo21] I. López-Espejo *et al.*, "Deep Spoken Keyword Spotting: An Overview". IEEE Access, 2021

Aprendizaje Auto-Supervisado de Representaciones de Voz

Phase 1: Pre-train



Phase 2: Downstream

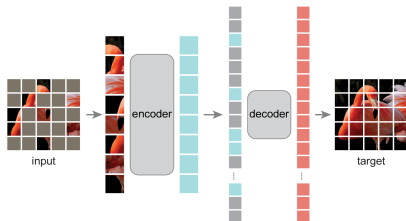


[Mohamed22] A. Mohamed *et al.*, "Self-Supervised Speech Representation Learning: A Review". IEEE Journal of Selected Topics in Signal Processing, 2022

- Existe una **ingente cantidad de datos de voz sin etiquetar** ←
¿Cómo aprovecharlos?
- Aprendizaje auto-supervisado de representaciones del habla:
 - 1 **Generativo**
 - 2 **Contrastivo**
 - 3 **Predictivo**
- Tareas posteriores: **Ajuste fino o no**
- En general, superior a las características relacionadas con la escala Mel

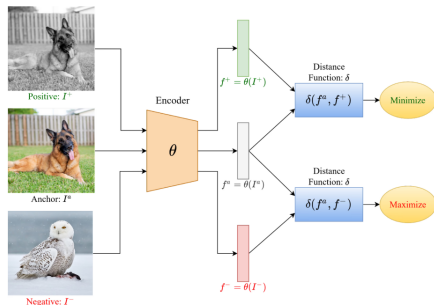
Aprendizaje Auto-Supervisado de Representaciones de Voz

- Ejemplo de **aprendizaje generativo**: **Reconstrucción de datos enmascarados**



[Yao24] Y. Yao *et al.*, "Masked Contrastive Representation Learning for Self-Supervised Visual Pre-Training". In Proc. of DSAA 2024

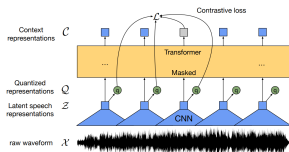
- Ejemplo de **aprendizaje contrastivo**



[Kundu22] R. Kundu, "The Beginner's Guide to Contrastive Learning". <https://www.v7labs.com/blog/contrastive-learning-guide>

Aprendizaje Auto-Supervisado de Representaciones de Voz

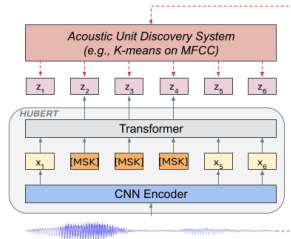
wav2vec 2.0



[Baevski20] A. Baevski *et al.*, "wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations". In Proc. of NeurIPS 2020

- Aprendizaje contrastivo con enmascaramiento

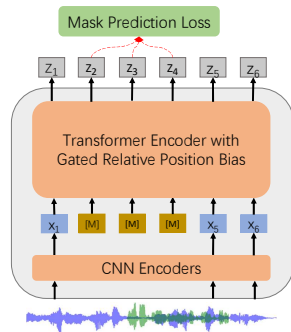
HuBERT



[Hsu21] W.-N. Hsu *et al.*, "HuBERT: Self-Supervised Speech Representation Learning by Masked Prediction of Hidden Units". IEEE/ACM Transactions on Audio, Speech, and Language Processing, 2021

- Entropía cruzada sólo sobre segmentos enmascarados

WavLM



[Chen22] S. Chen *et al.*, "WavLM: Large-Scale Self-Supervised Pre-Training for Full Stack Speech Processing". IEEE Journal of Selected Topics in Signal Processing, 2022

- Aprendizaje predictivo similar a HuBERT

Introducción a las Tecnologías del Habla

Dr. Iván López-Espejo

DÍA 1: INTRODUCCIÓN, APLICACIONES, Y ANÁLISIS DE LA SEÑAL DE VOZ

iloes@ugr.es

Monday 2nd June, 2025



PONTIFICIA UNIVERSIDAD
CATOLICA
DE VALPARAISO



**UNIVERSIDAD
DE GRANADA**

Introducción a las Tecnologías del Habla

Dr. Iván López-Espejo

DÍA 2: INTRODUCCIÓN A LA VERIFICACIÓN DE HABLANTE

iloes@ugr.es

Tuesday 3rd June, 2025



PONTIFICIA UNIVERSIDAD
CATOLICA
DE VALPARAISO



UNIVERSIDAD
DE GRANADA

Tabla de Contenidos

- 1 Introducción a la Verificación de Hablante
- 2 Implementación de un Sistema de Verificación de Hablante

Tabla de Contenidos

1 Introducción a la Verificación de Hablante

2 Implementación de un Sistema de Verificación de Hablante

Introducción a la Verificación de Hablante

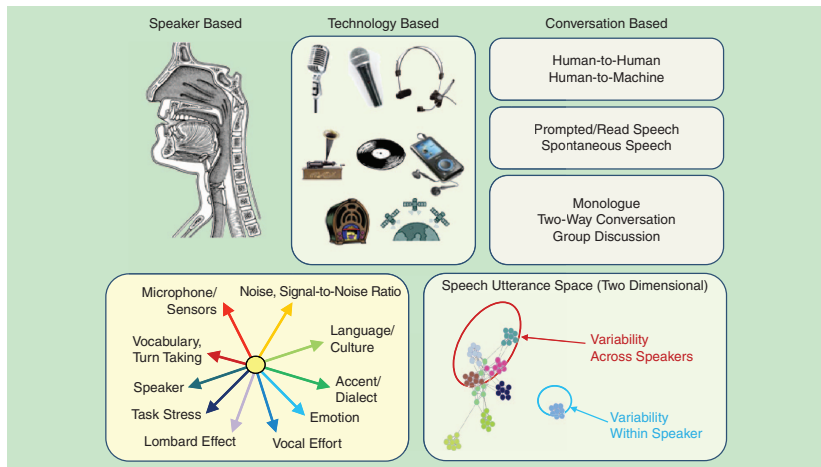
- Los sistemas de **reconocimiento de hablantes** se han convertido en un medio importante para verificar la identidad en aplicaciones de comercio electrónico, ciencia forense, aplicación de la ley, etc.
- Dos aproximaciones principales en **biometría de voz**:
 - 1 **Identificación/reconocimiento** ($1 : N$): Escenario **cerrado** vs. escenario **abierto**
 - 2 **Verificación** ($1 : 1$)
- **Independencia** vs. **dependencia del texto**



[MediaMedic] MediaMedic, "Voice Biometrics: Advancements and Applications in Forensic Investigations,"
<https://www.mediamedic.studio/voice-biometrics-advancements-and-applications/>

Introducción a la Verificación de Hablante

- Fuentes de **variabilidad** en el contexto del reconocimiento del hablante:



[Hansen15] J. H. L. Hansen and T. Hasan, "Speaker Recognition by Machines and Humans: A tutorial review," IEEE Signal Processing Magazine, 2015

Introducción a la Verificación de Hablante

- **Enrolamiento recursivo:** Actualizamos la muestra de referencia de un hablante tras una verificación exitosa para robustecer el sistema frente a la *variabilidad intra-hablante*

- ① *Edad (paso del tiempo)*
- ② *Enfermedad*
- ③ *Estado anímico*
- ④ ...



- **¡Si y sólo si la verificación ha sido exitosa!** (i.e., $s' = s$):

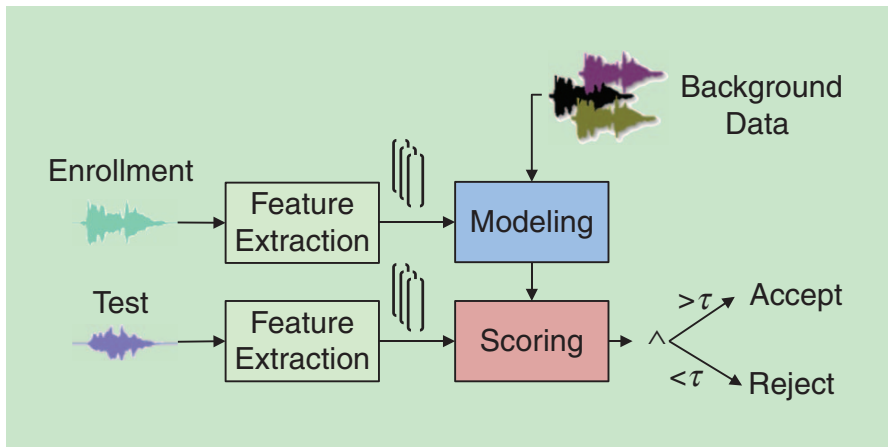
$$\mathbf{e}_t^{(s)} = \lambda(\sigma(x), \Psi, \gamma) \mathbf{e}_{t-1}^{(s)} + (1 - \lambda(\sigma(x), \Psi, \gamma)) \mathbf{v}_t^{(s')}$$

- $\mathbf{v}_t^{(s')}$: Nueva muestra de verificación procedente del hablante s'
- $\mathbf{e}_{t-1}^{(s)}$: Muestra de referencia anterior del hablante s
- $\mathbf{e}_t^{(s)}$: Muestra de referencia actualizada del hablante s
- $\lambda(\sigma(x), \Psi, \gamma)$: Factor de recuerdo dependiente de la puntuación calibrada $\sigma(x)$

[Espejo24] I. López-Espejo et al., “Authenticating a User,” US Patent, 2024

Introducción a la Verificación de Hablante

- Diagrama de bloques de un **sistema básico de verificación de hablante**:



[Hansen15] J. H. L. Hansen and T. Hasan, "Speaker Recognition by Machines and Humans: A tutorial review," IEEE Signal Processing Magazine, 2015

Bases de Datos

- **VoxCeleb1**: Más de 100k grabaciones de 1.251 famosos extraídas de YouTube
- **VoxCeleb2**: Más de 1M de grabaciones de 6.112 famosos extraídas de YouTube



[Nagrani17] A. Nagrani *et al.*, "VoxCeleb: a large-scale speaker identification dataset," in Proc. of Interspeech 2017

[Chung18] J. S. Chung *et al.*, "VoxCeleb2: Deep Speaker Recognition," in Proc. of Interspeech 2018



- **NIST SRE (Speaker Recognition Evaluation)**: Desde el año 1996 hasta la actualidad
- **NIST SRE 2018**

<https://sre.nist.gov/>

Curvas ROC y DET

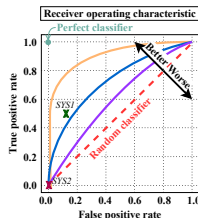
- Probabilidad de que una muestra positiva sea correctamente detectada como tal:

$$\text{True Positive Rate (TPR)} = \text{Recall} \equiv \frac{TP}{TP + FN}$$

- Probabilidad de que una muestra negativa sea incorrectamente clasificada como positiva:

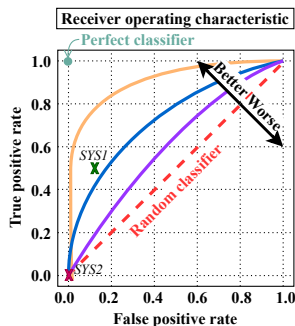
$$\text{False Positive Rate (FPR)} \equiv \frac{FP}{FP + TN}$$

- La curva **característica operativa del receptor (ROC)** se obtiene barriendo el umbral de sensibilidad/decisión:



Curvas ROC y DET

- **Área bajo la curva ROC ($AUC_{ROC} \in [0, 1]$):** Probabilidad de que un clasificador clasifique una muestra positiva elegida aleatoriamente en un rango superior a una muestra negativa elegida aleatoriamente



Ground truth NK NK KW NK NK KW NK NK NK NK

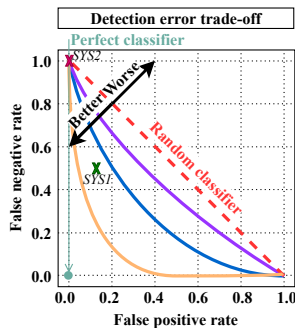
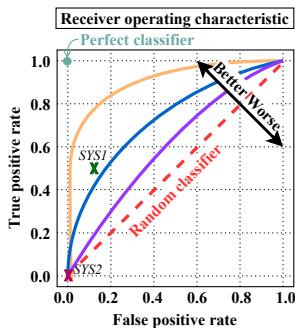
SYS1	NK	NK	KW	NK	NK	NK	KW	NK	NK	NK
SYS2	NK	NK	NK	NK	NK	NK	NK	NK	NK	NK

Curvas ROC y DET

- Dado que

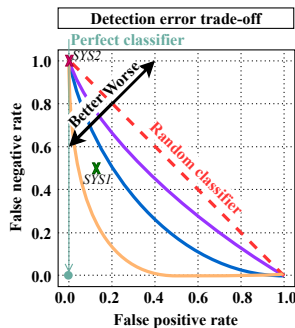
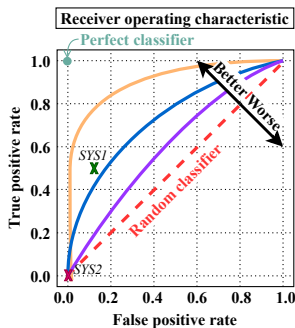
$$\text{False Negative Rate (FNR)} \equiv \frac{FN}{FN + TP} = 1 - \text{TPR},$$

la curva de **compensación de errores de detección (DET)** no es otra cosa que una versión volteada verticalmente de la curva ROC:

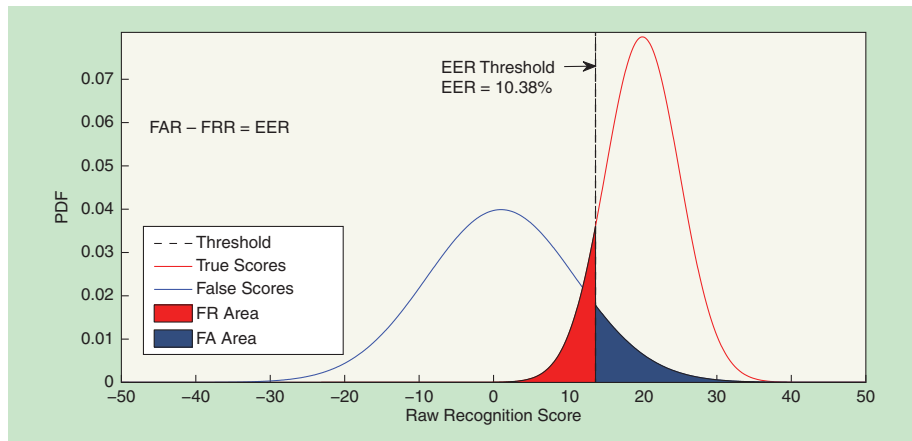


Curvas ROC y DET

- **Área bajo la curva DET ($AUC_{DET} \in [0, 1]$):** Cuanto menor, mejor
- **Tasa de error igual (EER):** Punto de intersección entre la función identidad y la curva DET (i.e., punto en el cual $FNR = FPR$)



Estableciendo el Umbral de Decisión



[Hansen15] J. H. L. Hansen and T. Hasan, "Speaker Recognition by Machines and Humans: A tutorial review," IEEE Signal Processing Magazine, 2015

Función de Coste de Detección

- La **función de coste de detección (DCF)** fue propuesta por **NIST (National Institute of Standards and Technology)**



$$DCF(\tau) = C_{miss}P_{miss}(\tau)P_{target} + C_{FA}P_{FA}(\tau)(1 - P_{target})$$

- 1 C_{miss} : Coste de un falso negativo (p. ej., 10)
 - 2 C_{FA} : Coste de un falso positivo (p. ej., 1)
 - 3 P_{target} : Probabilidad *a priori* de un hablante objetivo (p. ej., 0,01)
 - 4 $P_{miss}(\tau)$: Probabilidad de un falso negativo dado un umbral τ
 - 5 $P_{FA}(\tau)$: Probabilidad de un falso positivo dado un umbral τ
- El objetivo sería encontrar el **valor de τ que minimiza $DCF(\tau)$**

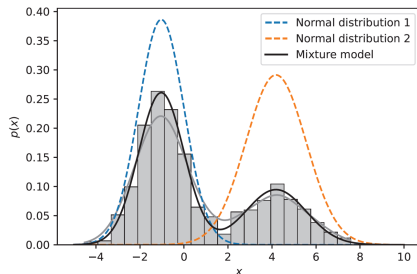
Introducción a la Verificación de Hablante

CINCO GENERACIONES PRINCIPALES

- ① **GMM-UBM (Gaussian Mixture Models-Universal Background Model)**: Tecnología basada en modelos de mezcla de gaussianas
- ② **GMM-SVM (Gaussian Mixture Models-Support Vector Machines)**: Supervectores GMM clasificados con SVMs
- ③ **JFA (Joint Factor Analysis)**: Descomposición de la variabilidad total en componentes de hablante y canal/sesión
- ④ **i-vectors (identity vectors)**: Modelado de la variabilidad total
- ⑤ **Redes neuronales**: Tecnología basada en aprendizaje profundo

1ª Generación: GMM-UBM (2000)

$\mu_1 = -1, \sigma_1 = 1$ — $\mu_2 = 4, \sigma_2 = 1.5$
 $w_1 = 0.7$ — $w_2 = 0.3$



[Yehoshua23] R. Yehoshua, "Gaussian Mixture Models (GMMs): from Theory to Implementation," <https://towardsdatascience.com/gaussian-mixture-models-gmms-from-theory-to-implementation/>

$$\theta = \{w_k, \mu_k, \Sigma_k; 1 \leq k \leq \mathcal{K}\}$$

Dado un conjunto de datos de entrenamiento $\mathbf{X} = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(N)}\}$,

$$\theta^* = \arg \max_{\theta} p(\mathbf{X}|\theta) = \arg \max_{\theta} \prod_{i=1}^N p(\mathbf{x}^{(i)}|\theta) \text{ (¡Expectación-maximización!)}$$

Modelos de mezcla de gaussianas:

$$\mathcal{N}(\mathbf{x}|\mu, \Sigma) = \frac{e^{-\frac{1}{2}(\mathbf{x}-\mu)^{\top} \Sigma^{-1}(\mathbf{x}-\mu)}}{\sqrt{(2\pi)^d |\Sigma|}}$$

$$p(\mathbf{x}) = \sum_{k=1}^{\mathcal{K}} w_k \mathcal{N}_k(\mathbf{x}|\mu_k, \Sigma_k)$$

$$\sum_{k=1}^{\mathcal{K}} w_k = 1$$

$$0 \leq w_k \leq 1 \quad \forall k = 1, \dots, \mathcal{K}$$

1ª Generación: GMM-UBM (2000)

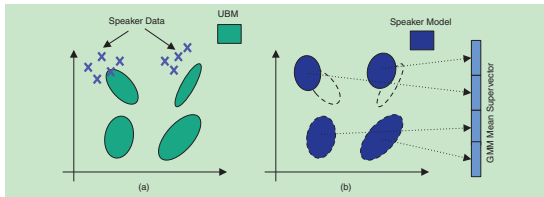
- λ_s : GMM dependiente del hablante s (**adaptación MAP de un UBM**)
- λ_0 : UBM
- $\mathbf{X} = \{\mathbf{x}_n; n = 1, \dots, T\}$ son vectores de características procedentes de una observación O

Contraste de hipótesis:

- 1 H_0 : O proviene del hablante s
- 2 H_1 : O no procede del hablante s

Se calcula $\Lambda(\mathbf{X}) = \log \left(\frac{p(\mathbf{X}|\lambda_s)}{p(\mathbf{X}|\lambda_0)} \right) = \log p(\mathbf{X}|\lambda_s) - \log p(\mathbf{X}|\lambda_0)$

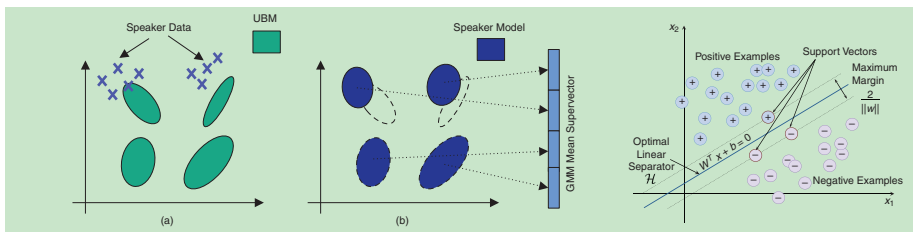
Si $\Lambda(\mathbf{X}) \geq \tau$, entonces se acepta la hipótesis H_0



[Hansen15] J. H. L. Hansen and T. Hasan, "Speaker Recognition by Machines and Humans: A tutorial review," IEEE Signal Processing Magazine, 2015

2ª Generación: GMM-SVM (2006)

- Los **supervectores GMM** proporcionan representaciones del hablante de **dimensionalidad fija**
- GMM-SVM**: Clasificación de supervectores GMM usando SVMs (*Support Vector Machines*)



[Hansen15] J. H. L. Hansen and T. Hasan, "Speaker Recognition by Machines and Humans: A tutorial review," IEEE Signal Processing Magazine, 2015

3ª Generación: JFA (2004)

- **FA (Factor Analysis)**: Modo de explicar la variabilidad debida al hablante y al canal en el espacio de los **supervectores**

$\mathbf{m}_{s,h}$: Supervector GMM dependiente del hablante s (y de la sesión h):

$$\mathbf{m}_{s,h} = \mathbf{m}_0 + \mathbf{m}_{spk} + \mathbf{m}_{chn} + \mathbf{m}_{res}$$

- 1 $\mathbf{m}_0$: Componente independiente del entorno, canal y hablante (cte., procedente del UBM)
- 2 $\mathbf{m}_{spk}$: Componente dependiente del hablante
- 3 $\mathbf{m}_{chn}$: Componente dependiente del entorno y canal
- 4 $\mathbf{m}_{res}$: Residuo

3ª Generación: JFA (2004)

- **Joint Factor Analysis – JFA** (*en el dominio de los supervectores GMM*):

$$\mathbf{m}_{s,h} = \mathbf{m}_0 + \underbrace{\mathbf{U}\mathbf{x}_h}_{\mathbf{m}_{chn}} + \underbrace{\mathbf{V}\mathbf{y}_s}_{\mathbf{m}_{spk}} + \underbrace{\mathbf{D}\mathbf{z}_{s,h}}_{\mathbf{m}_{res}}$$

- **U** y **V** son **matrices de bajo rango** estimadas durante una fase de **entrenamiento** mediante un algoritmo de reducción de dimensionalidad tipo PCA
- **D** es una **matriz diagonal** estimada junto a **U** y **V** mediante un algoritmo tipo EM
- Dado un supervector $\mathbf{m}_{s,h}$, $\mathbf{x}_h$, $\mathbf{y}_s$ y $\mathbf{z}_{s,h}$ (los **factores de canal, de hablante y residuales**) se obtienen mediante estimación MAP o bayesiana considerando que $(\mathbf{x}_h, \mathbf{y}_s, \mathbf{z}_{s,h}) \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
- $\mathbf{y}_s$ es el **vector de características** del hablante usado en la **comparación/verificación**

4ª Generación: i-vectors (2009)

- **Dr. Najim Dehak** investigó el uso de **JFA** como **extractor de características** ($\mathbf{y}_s$) y **SVMs** para **clasificación**
- Se percató de que $\mathbf{x}_h$ *también contenía información relativa al hablante*
- **Espacio de variabilidad total**: Decidió combinar en un único espacio los factores de hablante y canal:

$$\mathbf{m}_{s,h} = \mathbf{m}_0 + \mathbf{T}\mathbf{w}_{s,h}$$

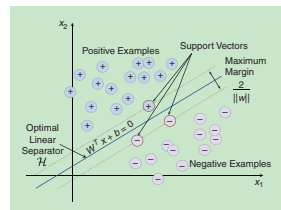
$\mathbf{T}$ es la matriz de variabilidad total de bajo rango y $\mathbf{w}_{s,h} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

i-vector: $\mathbf{w}_{s,h}^* = E[\mathbf{w}_{s,h} | \mathbf{F}]$

$$\mathbf{F} = \sum_n \gamma_k(n) (\mathbf{x}_n - \boldsymbol{\mu}_k^{UBM})$$

$$\gamma_k(n) = P(k | \mathbf{x}_n) = \frac{p(\mathbf{x}_n | k) P(k)}{p(\mathbf{x}_n)} = \frac{\mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) w_k}{\sum_k w_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}$$

Najim Dehak, PhD

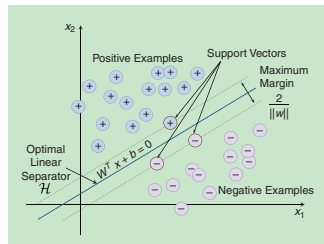


[Hansen15] J. H. L. Hansen and T. Hasan, "Speaker Recognition by Machines and Humans: A tutorial review," IEEE Signal Processing Magazine, 2015

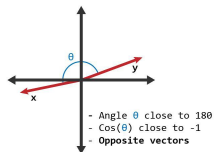
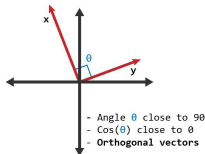
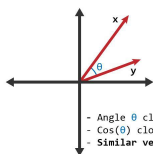
Pero... ¿Cómo Usamos y_s y $w_{s,h}$ para Verificación?

- Extraemos w_{test} de una muestra para verificar y reclamamos la identidad asociada a w_{target}

● SVMs (Support Vector Machines)



[Hansen15] J. H. L. Hansen and T. Hasan, "Speaker Recognition by Machines and Humans: A tutorial review," IEEE Signal Processing Magazine, 2015



[Karabiber24] F. Karabiber, "Cosine Similarity,"

<https://www.learn datasci.com/glossary/cosine-similarity/>

● Similitud coseno:

$$S_c(w_{test}, w_{target}) = \cos(\theta) =$$

$$\frac{w_{test} \cdot w_{target}}{\|w_{test}\| \|w_{target}\|} \in [-1, 1]$$

PLDA

- **PLDA (Probabilistic Linear Discriminant Analysis)**: Sigue supuestos de modelado similares a los de JFA

Un **i-vector** $\mathbf{w}_{s,h}$ se puede descomponer como:

$$\mathbf{w}_{s,h} = \mathbf{w}_0 + \Phi\beta_s + \Gamma\alpha_h + \epsilon_{s,h}$$

- ① $\mathbf{w}_0$ es un i-vector promedio *independiente* del hablante
 - ② Φ y Γ son **matrices de bajo rango** que caracterizan los subespacios de hablante y canal
 - ③ $(\beta_s, \alpha_h) \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ son factores de hablante y canal
 - ④ $\epsilon_{s,h}$ es un vector de residuo
- Si $\mathbf{w}_{s,h} \leftarrow \mathbf{w}_{s,h} / \|\mathbf{w}_{s,h}\|_2$, $\mathbf{w}_{s,h} \sim \mathcal{N}$ (**modelo PLDA gaussiano**)
 - Usar un modelo de covarianza completa para $\epsilon_{s,h} \sim \mathcal{N}(\mathbf{0}, \Sigma_\epsilon)$ nos permite simplificar el modelo PLDA:

$$\mathbf{w}_{s,h} = \mathbf{w}_0 + \Phi\beta_s + \epsilon_{s,h}$$

PLDA

- Extraemos $\mathbf{w}_{test}$ de una muestra para verificar y reclamamos la identidad asociada a $\mathbf{w}_{target}$

Contraste de hipótesis:

- 1 H_0 : $\mathbf{w}_{test}$ y $\mathbf{w}_{target}$ provienen del mismo hablante
- 2 H_1 : $\mathbf{w}_{test}$ y $\mathbf{w}_{target}$ proceden de diferentes hablantes

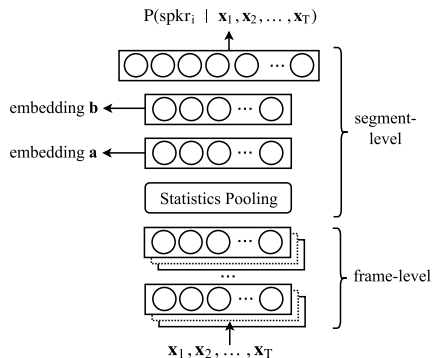
$$\text{LLR}(\mathbf{w}_{test}, \mathbf{w}_{target}) = \log \left(\frac{P(\mathbf{w}_{test}, \mathbf{w}_{target} | H_0)}{P(\mathbf{w}_{test} | H_1) P(\mathbf{w}_{target} | H_1)} \right)$$

- $\text{LLR}(\mathbf{w}_{test}, \mathbf{w}_{target})$ se puede aproximar como función de Φ y Σ_ϵ
- Si $\text{LLR}(\mathbf{w}_{test}, \mathbf{w}_{target}) \geq \tau$, se acepta que $\mathbf{w}_{test}$ y $\mathbf{w}_{target}$ provienen del mismo hablante

5º Generación: Redes Neuronales (2018)

- Múltiples propuestas basadas en red neuronal, si bien fueron los **x-vectors** los que rompieron la baraja

TDNN (Time-Delay Neural Network)



Layer	Layer context	Total context	Input x output
frame1	$[t - 2, t + 2]$	5	120x512
frame2	$\{t - 2, t, t + 2\}$	9	1536x512
frame3	$\{t - 3, t, t + 3\}$	15	1536x512
frame4	$\{t\}$	15	512x512
frame5	$\{t\}$	15	512x1500
stats pooling	$[0, T)$	T	1500Tx3000
segment6	$\{0\}$	T	3000x512
segment7	$\{0\}$	T	512x512
softmax	$\{0\}$	T	512xN

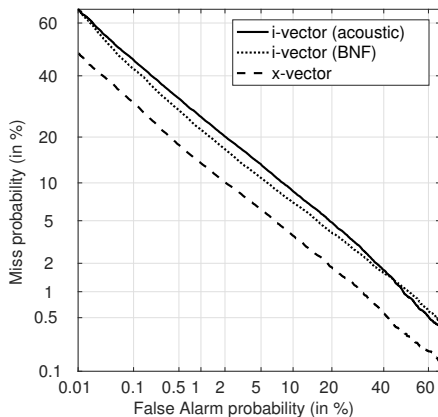
[Snyder18] D. Snyder *et al.*, "X-Vectors: Robust DNN Embeddings for Speaker Recognition," in Proc. of ICASSP 2018

- x-vectors**: Salida de la capa *segment6* (a partir de espectrograma Mel)
- La comparación de x-vectors se realiza mediante **PLDA**
- ¡El **aumento de datos** de entrenamiento es clave!

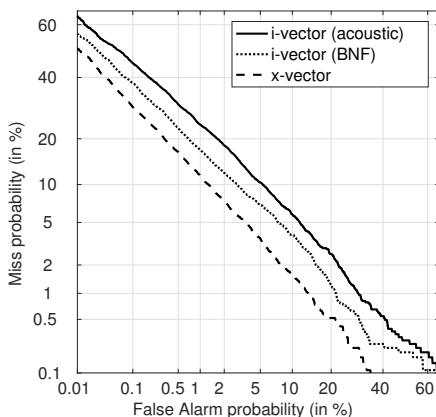
[Snyder17] D. Snyder *et al.*, "Deep Neural Network Embeddings for Text-Independent Speaker Verification," in Proc. of Interspeech 2017

5º Generación: Redes Neuronales (2018)

Curva DET para el set cantonés de NIST SRE16



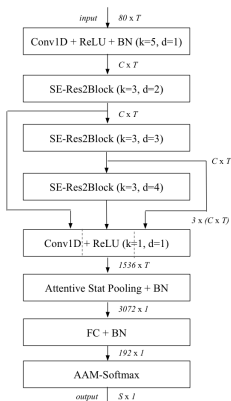
Curva DET para "Speakers In The Wild"



[Snyder18] D. Snyder *et al.*, "X-Vectors: Robust DNN Embeddings for Speaker Recognition," in Proc. of ICASSP 2018

5º Generación: Redes Neuronales (2018)

- **ECAPA-TDNN**: TDNN mejorada para extracción de *speaker embeddings*



[Desplanques20] B. Desplanques *et al.*, "ECAPA-TDNN: Emphasized Channel Attention, Propagation and Aggregation in TDNN Based Speaker Verification," in Proc. of Interspeech 2020

I. López-Espejo (UGR)

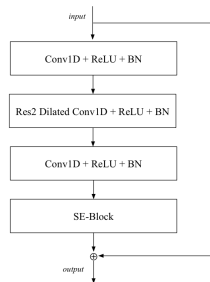
Attentive Stat Pooling

$$e_{t,c} = \mathbf{v}_c^\top f(\mathbf{W}\mathbf{h}_t + \mathbf{b}) + k_c$$

$$\alpha_{t,c} = \frac{\exp(e_{t,c})}{\sum_{\tau} \exp(e_{\tau,c})}$$

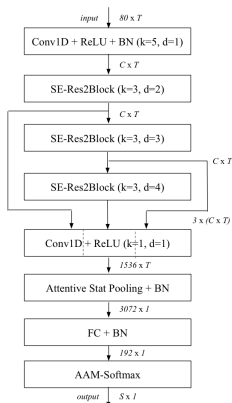
$$\tilde{\mu}_c = \sum_t \alpha_{t,c} h_{t,c} \quad \tilde{\sigma}_c = \sqrt{\sum_t \alpha_{t,c} h_{t,c}^2 - \tilde{\mu}_c^2}$$

$$\tilde{\boldsymbol{\mu}} = (\tilde{\mu}_1, \dots, \tilde{\mu}_c, \dots, \tilde{\mu}_C) \quad \tilde{\boldsymbol{\sigma}} = (\tilde{\sigma}_1, \dots, \tilde{\sigma}_c, \dots, \tilde{\sigma}_C)$$



5º Generación: Redes Neuronales (2018)

- **ECAPA-TDNN**: TDNN mejorada para extracción de *speaker embeddings*



[Desplanques20] B. Desplanques *et al.*, "ECAPA-TDNN: Emphasized Channel Attention, Propagation and Aggregation in TDNN Based Speaker Verification," in Proc. of Interspeech 2020

I. López-Espejo (UGR)

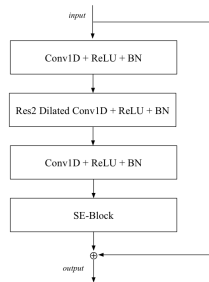
SE (Squeeze-and-Excitation)-Block

$$\text{Squeeze: } \mathbf{z} = \frac{1}{T} \sum_t \mathbf{h}_t$$

Excitation:

$$\mathbf{s} = \sigma(\mathbf{W}_2 f(\mathbf{W}_1 \mathbf{z} + \mathbf{b}_1) + \mathbf{b}_2)$$

$$\tilde{\mathbf{h}}_c = s_c \mathbf{h}_c \quad (s_c \in [0, 1])$$



- Agregación de características multicapa

5º Generación: Redes Neuronales (2018)

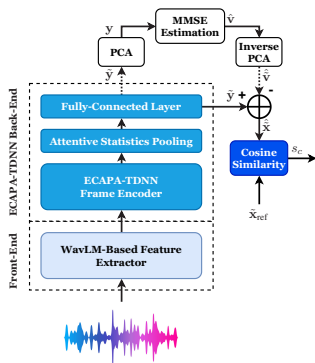
- La comparación de *speaker embeddings* se lleva a cabo mediante **similitud coseno**:

Architecture	# Params	VoxCeleb1		VoxCeleb1-E		VoxCeleb1-H		VoxSRC19
		EER(%)	MinDCF	EER(%)	MinDCF	EER(%)	MinDCF	EER(%)
E-TDNN	6.8M	1.49	0.1604	1.61	0.1712	2.69	0.2419	1.81
E-TDNN (large)	20.4M	1.26	0.1399	1.37	0.1487	2.35	0.2153	1.61
ResNet18	13.8M	1.47	0.1772	1.60	0.1789	2.88	0.2672	1.97
ResNet34	23.9M	1.19	0.1592	1.33	0.1560	2.46	0.2288	1.57
ECAPA-TDNN (C=512)	6.2M	1.01	0.1274	1.24	0.1418	2.32	0.2181	1.32
ECAPA-TDNN (C=1024)	14.7M	0.87	0.1066	1.12	0.1318	2.12	0.2101	1.22

[Desplanques20] B. Desplanques *et al.*, "ECAPA-TDNN: Emphasized Channel Attention, Propagation and Aggregation in TDNN Based Speaker Verification," in Proc. of Interspeech 2020

Compensación de *Embeddings*

- Necesidad de compensación de *embeddings* procedentes de **habla con fonación no neutra** (p. ej., **gritada** y **susurrada**)



$\tilde{x} \in \mathbb{R}^D$: *Embedding* normal

$\tilde{y} \in \mathbb{R}^D$: *Embedding* de fonación no neutra

$\tilde{v} \in \mathbb{R}^D$: **Vector de transferencia de esfuerzo vocal**

$$\tilde{y} = \tilde{x} + \tilde{v} \Rightarrow \hat{x} = \tilde{y} - \hat{v}$$

- $\mathbf{v} = \mathbf{W}_L^\top \tilde{\mathbf{v}}$ e $\mathbf{y} = \mathbf{W}_L^\top \tilde{\mathbf{y}}$, donde $\mathbf{W}_L \in \mathbb{R}^{D \times L}$, $L \ll D$, es una matriz de transformación PCA
- $p(\mathbf{z} = (\mathbf{v}, \mathbf{y}) \in \mathbb{R}^{2L})$ se modela mediante un GMM

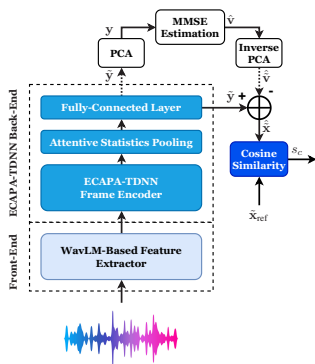
$$\hat{\mathbf{x}} = \tilde{\mathbf{y}} - \underbrace{\mathbf{W}_L \hat{\mathbf{v}}}_{\hat{\mathbf{v}}}$$

$$\hat{\mathbf{v}} = \mathbb{E}[\mathbf{v}|\mathbf{y}] \text{ (estimación MMSE)}$$

[Espejo23] I. López-Espejo et al., "Improved Vocal Effort Transfer Vector Estimation for Vocal Effort-Robust Speaker Verification," in Proc. of MLSP 2023

Compensación de *Embeddings*

- Necesidad de compensación de *embeddings* procedentes de **habla con fonación no neutra** (p. ej., **gritada** y **susurrada**)



[Espejo23] I. López-Espejo *et al.*, "Improved Vocal Effort Transfer Vector Estimation for Vocal Effort-Robust Speaker Verification," in Proc. of MLSP 2023

- La ECAPA-TDNN se entrena sobre una versión aumentada de **VoxCeleb2**
- La métrica de evaluación es **EER (%)** (*cuanto menor, mejor*)

Shouted and normal speech:

Condition	E-T+MFCC	E-T+WavLM	MEMLIN	MEMLIN+PCA	MMSE _x	MMSE _v
A _S -A _S	19.96	17.11	15.62	31.50	28.72	15.22
N _S -N _S	9.73	7.25	7.25	7.25	7.25	7.25
S-S	11.58	9.94	10.44	27.46	25.53	5.91
N _S -S	25.28	21.76	20.74	41.00	35.56	17.74

Whispered and normal speech:

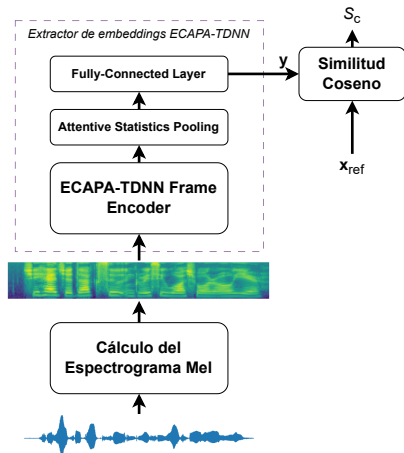
Condition	E-T+MFCC	E-T+WavLM	MEMLIN	MEMLIN+PCA	MMSE _x	MMSE _v
A _W -A _W	16.54	11.24	8.25	31.87	23.95	8.27
N _W -N _W	1.21	0.62	0.62	0.62	0.62	0.62
W-W	4.38	5.26	4.00	19.31	19.77	2.87
N _W -W	12.81	9.81	11.47	44.38	30.59	8.86

Tabla de Contenidos

1 Introducción a la Verificación de Hablante

2 Implementación de un Sistema de Verificación de Hablante

Implementación de un Sistema de Verificación de Hablante



Implementación de un sistema de verificación de hablante:

- Cálculo del **espectrograma Mel**
- Extractor de *embeddings* $\mathbf{y}, \mathbf{x}_{ref} \in \mathbb{R}^{192}$ basado en **ECAPA-TDNN**
- Comparación de *embeddings* mediante **similitud coseno** (si $S_c \geq \tau$, $\mathbf{y}$ y $\mathbf{x}_{ref}$ proceden del mismo hablante)



<https://huggingface.co/yangwang825/ecapa-tdnn-vox2>

Implementación de un Sistema de Verificación de Hablante

Instalación de Anaconda y Spyder:



python™



- ➊ Ir a <https://www.anaconda.com/download> y descargar e instalar Anaconda Distribution
- ➋ Iniciar anaconda-navigator
- ➌ En *Environments*, crear el entorno de trabajo cursoVoz
- ➍ Seleccionar el nuevo entorno, ver los paquetes *Not Installed* y marcar e instalar Spyder
- ➎ En *Home*, seleccionar el entorno cursoVoz y lanzar Spyder
- ➏ En Spyder, ir a *Tools*→*Preferences*→*iPython Console*→*Graphics*→*Backend* y elegir *Automatic* como el modo en que se muestran los gráficos

Implementación de un Sistema de Verificación de Hablante

 PyTorch SpeechBrain NumPy matplotlib

- **Necesitamos instalar algunas librerías de trabajo:**

- **PyTorch:** Construcción de modelos de aprendizaje profundo
- **SpeechBrain:** Desarrollo de tecnología de voz, audio, etc.
- **Numpy:** Cálculo científico
- **Sounddevice:** Grabación y reproducción de sonido
- **Matplotlib:** Creación de visualizaciones

- ① Abrir Anaconda Prompt y activar el entorno de trabajo mediante el comando `conda activate cursoVoz`
- ② Instalar los anteriores módulos haciendo uso de `pip`:
`pip install torch speechbrain numpy sounddevice matplotlib`

Implementación de un Sistema de Verificación de Hablante

- Creamos un fichero de código Python llamado `spkverif_lib.py`
- **Importamos los módulos** que vamos a necesitar:

```
import torch
from speechbrain.pretrained.interfaces import Pretrained
import numpy as np
```

Implementación de un Sistema de Verificación de Hablante

- Creamos la **clase que define nuestro extractor de embeddings basado en el espectrograma Mel**:

```
class EmbeddingExtractor(Pretrained):

    # Módulos necesarios.
    MODULES_NEEDED = [
        "compute_features",
        "mean_var_norm",
        "embedding_model"
    ]

    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)

    # Método que realiza la extracción de embeddings en sí.
    # wav: Muestras de audio.
    def embedding_extractor(self, wav):

        wav = torch.tensor(wav) # Convertimos el array de numpy en un tensor.

        wav = wav.unsqueeze(0) # Añadimos una dimensión por la izquierda al segmento so-
        noro.

        # Extracción de características Mel.
        feats = self.mods.compute_features(wav)
        feats = self.mods.mean_var_norm(feats, torch.ones(1)) # Normalización de media y va-
        rianza.

        # Extracción del embedding a partir de las características Mel.
        embedding = self.mods.embedding_model(feats, torch.ones(1))

        # Convertimos el embedding en un array de numpy.
        embedding = embedding.numpy()
        embedding = embedding[0,0,: ]

        return embedding
```

Implementación de un Sistema de Verificación de Hablante

- Incluimos un método para **calcular la similitud coseno** de cara a determinar si dos *embeddings* dados proceden o no del mismo hablante:

```
def cosineDist(x, y):  
    # Calculamos las normas de los vectores.  
    nx = np.sqrt(np.sum(x**2))  
    ny = np.sqrt(np.sum(y**2))  
  
    # Normalizamos los vectores.  
    x /= nx  
    y /= ny  
  
    # Calculamos la similitud coseno.  
    Sc = np.dot(x,y)  
  
    return Sc
```

Implementación de un Sistema de Verificación de Hablante

- Implementamos un código muy básico con el que **podremos grabar dos muestras de voz y determinar si proceden o no del mismo hablante**
- Creamos un nuevo fichero de código Python llamado `Demo_Live.py` e importamos los módulos que vamos a requerir:

```
import spkverif_lib
import sounddevice as sd
import matplotlib.pyplot as plt
import numpy as np
```

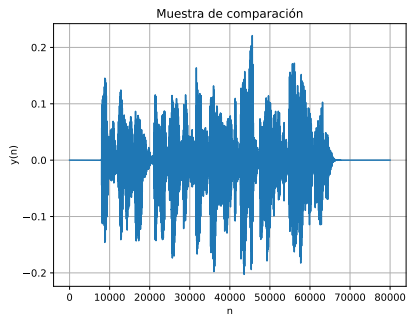
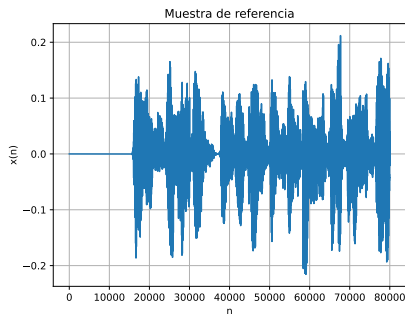

Implementación de un Sistema de Verificación de Hablante

- **Grabamos dos muestras de voz** de 5 segundos de duración cada una a 16 kHz de frecuencia de muestreo **y las representamos**:

```
fs = 16000 # Frecuencia de muestreo de trabajo en Hz: 16 kHz.
seconds = 5 # Duración en segundos de las muestras sonoras.
# ----- #
input('Presiona una tecla para comenzar a grabar la muestra de referencia...')
print('Grabando...')
samp1 = sd.rec(int(seconds * fs), samplerate=fs, channels=1)
sd.wait()
print('¡Grabación de la muestra de referencia concluida!')
# ----- #
input('Presiona una tecla para comenzar a grabar la muestra de comparación...')
print('Grabando...')
samp2 = sd.rec(int(seconds * fs), samplerate=fs, channels=1)
sd.wait()
print('¡Grabación de la muestra de comparación concluida!')
# ----- #
plt.figure()
plt.plot(samp1)
plt.grid(True)
plt.xlabel('n')
plt.ylabel('x(n)')
plt.title('Muestra de referencia')
plt.figure()
plt.plot(samp2)
plt.grid(True)
plt.xlabel('n')
plt.ylabel('y(n)')
plt.title('Muestra de comparación')
```

Implementación de un Sistema de Verificación de Hablante

- Ejemplo de dos muestras de voz grabadas por mí usando la presente implementación
- De la muestra de la izquierda se extraerá x_{ref} , y, de la de la derecha, y :



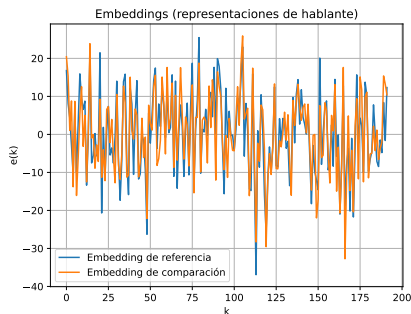
Implementación de un Sistema de Verificación de Hablante

- **Instanciamos**, desde un repositorio, **nuestro extractor de embeddings** basado en un modelo ECAPA-TDNN pre-entrenado usando [SpeechBrain](#)
- Extraemos los *embeddings* $\mathbf{x}_{ref}$ e $\mathbf{y}$ y los representamos:

```
# Instanciamos el extractor de embeddings.
emb_extractor = spkverif_lib.EmbeddingExtractor.from_hparams(
    source='yangwang825/ecapa-tdnn-vox2'
)
embedding_1 = emb_extractor.embedding_extractor(samp1[:,0]) # Embedding de referencia.
embedding_2 = emb_extractor.embedding_extractor(samp2[:,0]) # Embedding de comparación.
plt.figure()
plt.plot(embedding_1)
plt.plot(embedding_2)
plt.legend(['Embedding de referencia', 'Embedding de comparación'])
plt.xlabel('k')
plt.ylabel('e(k)')
plt.grid(True)
plt.title('Embeddings (representaciones de hablante)')
```

Implementación de un Sistema de Verificación de Hablante

- A continuación vemos los *embeddings* de **referencia**, $\mathbf{x}_{ref}$, y **comparación**, $\mathbf{y}$, correspondientes a nuestro ejemplo anterior:



- Su **similitud** salta a la vista, **consistente con** el hecho de **que proceden** de muestras de voz **de una misma persona**

Implementación de un Sistema de Verificación de Hablante

- Calculamos la similitud coseno $\mathcal{S}_c$ entre $\mathbf{x}_{ref}$ e $\mathbf{y}$, y, si $\mathcal{S}_c \geq \tau = 0,5$, decimos que las dos muestras de voz proceden del mismo hablante
- Representamos en el espacio de coordenadas polares los dos *embeddings*:

```

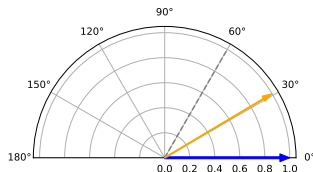
sc = spkverif.lib.cosineDist(embedding_1, embedding_2) # Similitud coseno.
thr = 0.5 # Umbral de decisión.
if sc >= thr:
    print('Las dos muestras provienen del mismo hablante')
else:
    print('Las muestras provienen de hablantes diferentes')
print('Similitud coseno: ' + str(sc))
# Representamos el resultado en términos del ángulo entre el embedding de
# referencia y el de comparación.
angle_ref = 0 # Referencia.
angle_com = np.acos(sc) # Comparación (arcocoseno de la similitud coseno).
arrow_length = 1.0 # Longitud del embedding.
fig, ax = plt.subplots(subplot_kw={'projection': 'polar'})
# Mostramos sólo el semicírculo de interés.
ax.set_thetamin(0)
ax.set_thetamax(180)
# Embedding de referencia.
ax.annotate("", # Anotación sin texto.
            xy=(angle_ref, arrow_length), # Punta del embedding.
            xytext=(0, 0), # Origen del embedding.
            arrowprops=dict(facecolor='blue', edgecolor='blue', width=2, headwidth=8, head-
length=10, shrink=0),
            annotation_clip=False) # Evita que la flecha se recorte.
# Embedding de comparación.
ax.annotate("",
            xy=(angle_com, arrow_length),
            xytext=(0, 0),
            arrowprops=dict(facecolor='orange', edgecolor='orange', width=1, headwidth=6,
headlength=10, shrink=0),
            annotation_clip=False)
ax.plot([0, np.acos(thr)], [0, arrow_length], linestyle='--', color='gray') # Representamos el
umbral de decisión también.
ax.set_title('Comparativa de embeddings')
ax.grid(True)
plt.show()

```

Implementación de un Sistema de Verificación de Hablante

- Los *embeddings* que forman un ángulo $\theta < \theta_\tau = \arccos(\tau = 0,5) = 60^\circ$ (**umbral de decisión** $\equiv$ línea discontinua) proceden del mismo hablante
- El *embedding de comparación* **y** forma un ángulo $\theta = \arccos(S_c)$ con el *embedding de referencia* **x_{ref}**

Comparativa de embeddings



- Se determina correctamente que **las dos muestras de voz del ejemplo proceden de un mismo hablante**

Introducción a las Tecnologías del Habla

Dr. Iván López-Espejo

DÍA 2: INTRODUCCIÓN A LA VERIFICACIÓN DE HABLANTE

iloes@ugr.es

Tuesday 3rd June, 2025



PONTIFICIA UNIVERSIDAD
CATOLICA
DE VALPARAISO



UNIVERSIDAD
DE GRANADA

Introducción a las Tecnologías del Habla

Dr. Iván López-Espejo

DÍA 3: INTRODUCCIÓN AL RECONOCIMIENTO DEL HABLA

iloes@ugr.es

Wednesday 4th June, 2025



PONTIFICIA UNIVERSIDAD
CATOLICA
DE VALPARAISO



UNIVERSIDAD
DE GRANADA

Tabla de Contenidos

- 1 Introducción al Reconocimiento del Habla
- 2 Implementación de un Sistema de Control por Voz

Tabla de Contenidos

1 Introducción al Reconocimiento del Habla

2 Implementación de un Sistema de Control por Voz

Introducción

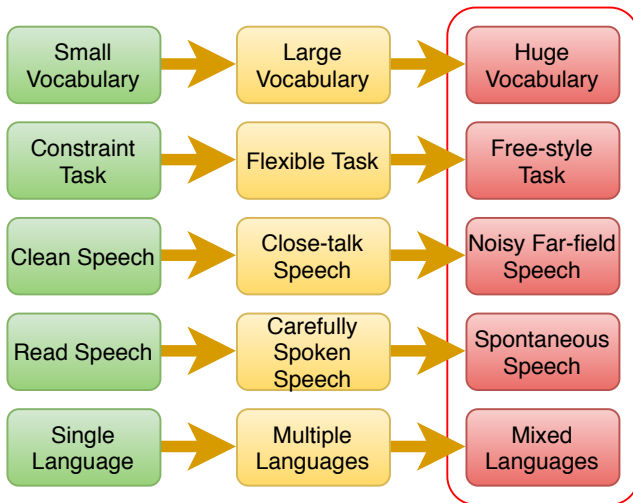
- Renacimiento del **reconocimiento automático del habla (RAH)** a lo largo de la última década: *¡El aprendizaje profundo ha revolucionado el RAH!*
 - ① Disponibilidad de una gran cantidad de datos de voz
 - ② Potentes recursos computacionales (**GPUs**)



[IWSDS] IWSDS, <http://www.iwsds.org/>

- **Multitud de aplicaciones:**
 - ① Búsqueda por voz, asistentes de voz, videojuegos, dictado...
 - ② **Detección de palabras clave (KWS)** con bajos requerimientos computacionales para dispositivos de asistencia auditiva ([Espejo20] I. López-Espejo *et al.*, "Improved External Speaker-Robust Keyword Spotting for Hearing Assistive Devices". IEEE/ACM Transactions on Audio, Speech, and Language Processing, 2020)

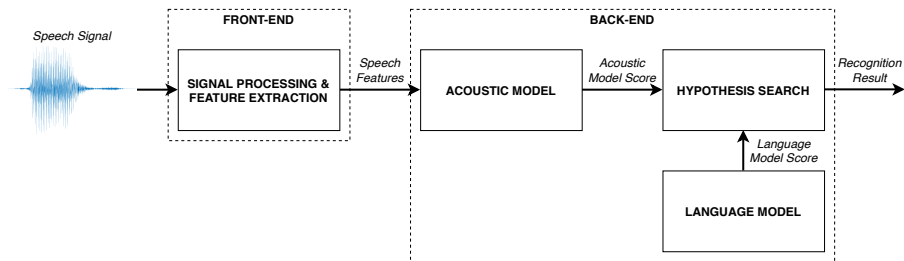
Introducción



[Yu15] D. Yu and L. Deng, "Automatic Speech Recognition: A Deep Learning Approach". Springer, 2015

Panorama General del RAH

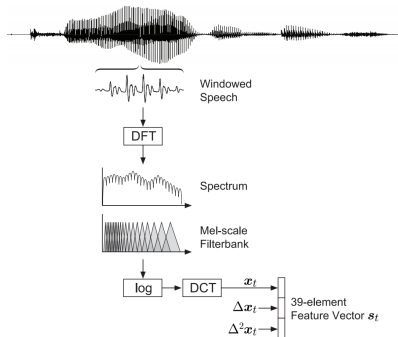
- Panorama general de la arquitectura de los sistemas de RAH:



- Componentes básicos del RAH:

- 1 **Procesamiento de la señal y extracción de características:** Espectro log-Mel, coeficientes cepstrales en escala Mel (**MFCCs**)...
- 2 **Modelo acústico (AM):** Integra información acústica y fonética
- 3 **Modelo de lenguaje (LM):** Estima la probabilidad de una secuencia de palabras (puntuación LM) gracias a que aprende, a partir de corpus de texto, correlaciones entre palabras
- 4 **Búsqueda de hipótesis:** Devuelve la secuencia de palabras con la puntuación más alta como resultado del reconocimiento

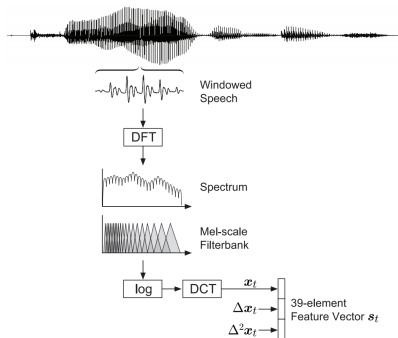
Panorama General del RAH: Front-end



[Liao07] H. Liao, "Uncertainty Decoding for Noise Robust Speech Recognition". Ph.D. thesis (University of Cambridge), 2007

- Propiedades deseables de las características del habla: **Discriminativas**, **compactas** y **robustas** a distorsiones acústicas (p. ej., ruido ambiente)
- Dependiendo del modelado acústico...
 - ① Modelos de mezcla de gaussianas (**GMMs**): Utilización de derivadas de coeficientes
 - ② Redes neuronales profundas (**DNNs**): Utilización de contexto temporal

Panorama General del RAH: Front-end



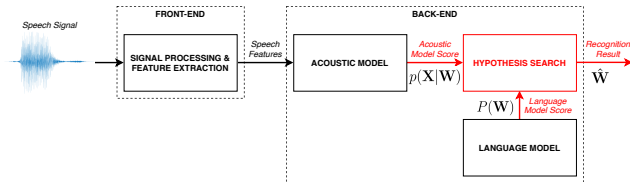
[Liao07] H. Liao, "Uncertainty Decoding for Noise Robust Speech Recognition". Ph.D. thesis (University of Cambridge), 2007

- Los MFCCs se ajustan bien a los modelos acústicos basados en GMM (matrices de covarianza diagonales, menor complejidad)
- El espectro log-Mel se ajusta bien a los modelos acústicos basados en DNN (aprovechamiento de correlaciones espectro-temporales)

Panorama General del RAH: Back-end

- El objetivo del RAH es encontrar la secuencia de palabras $\mathbf{W} = (w_1, w_2, \dots, w_m)$ más probable a partir de un conjunto de vectores de características $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_T)$
- Problema de estimación maximum a posteriori (MAP):

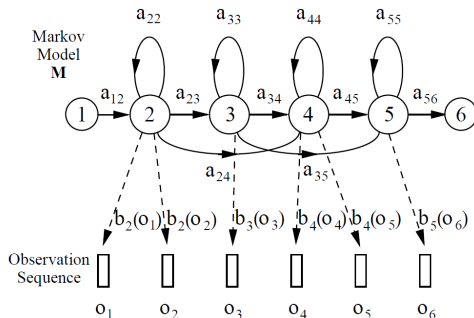
$$\hat{\mathbf{W}} = \arg \max_{\mathbf{W}} P(\mathbf{W}|\mathbf{X}) = \arg \max_{\mathbf{W}} p(\mathbf{X}|\mathbf{W})P(\mathbf{W})$$
- El **algoritmo de Viterbi** nos permite decodificar $\mathbf{W}$ a partir de las observaciones $\mathbf{X}$



- Para encontrar $p(\mathbf{X}|\mathbf{W})$, requerimos tanto el **léxico** (i.e., la correspondencia entre las palabras escritas que pueden ser reconocidas y sus transcripciones fonéticas) como el **modelo acústico**

Panorama General del RAH: Back-end

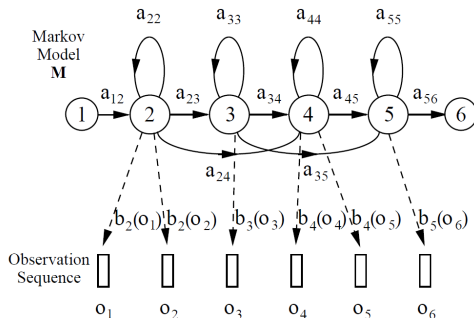
- El **modelo acústico** es el responsable de proporcionar $p(\mathbf{X}|\mathbf{W})$
- Cada palabra $w_i \in \mathbf{W}$, $i = 1, \dots, m$, normalmente se descompone en unidades acústicas más simples (i.e., monofonemas o trifonemas) a partir del léxico
- Cada una de estas unidades acústicas se modela mediante un **modelo oculto de Márkov (HMM)** con funciones de densidad continuas (*velocidad variable del habla*)
- **Recuerda:** Los parámetros de un HMM se obtienen mediante estimación de máxima verosimilitud empleando el algoritmo de **Baum-Welch (EM)**



[Young06] S. Young *et al.*, "The HTK Book (for HTK Version 3.4)". Cambridge University Engineering Department, 2006

Panorama General del RAH: Back-end

- Cada distribución $b_j(\mathbf{o} = \mathbf{x}_t)$ expresa la probabilidad de que el vector de características $\mathbf{x}_t$ sea observado en el estado s_j
- Modelado de las distribuciones de observaciones de salida de los estados del HMM
 - ① Usando GMMs: $b_j(\mathbf{x}_t | s_j) = \sum_{k=1}^K P(k | s_j) \mathcal{N}(\mathbf{x}_t | \boldsymbol{\mu}_{s_j}^{(k)}, \boldsymbol{\Sigma}_{s_j}^{(k)})$
 - ② ¡Es mucho mejor usar **DNNs** para producir las probabilidades de emisión de los estados!



[Young06] S. Young *et al.*, "The HTK Book (for HTK Version 3.4)". Cambridge University Engineering Department, 2006

Panorama General del RAH: Back-end

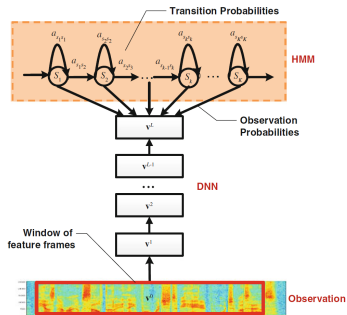
- $p(\mathbf{X}|\mathbf{W})$ se puede calcular sumando todas las posibles secuencias de estados $\mathbf{q} = (q_1, \dots, q_T)$ que pueden producir $\mathbf{W}$:

$$p(\mathbf{X}|\mathbf{W}) = \sum_{\mathbf{q}} \prod_{t=1}^T p(\mathbf{x}_t|q_t)P(q_t|q_{t-1})$$
- $P(\mathbf{W})$ depende de la tarea lingüística. Para N -gramas (usualmente, $N = 2$ o $N = 3$): $P(\mathbf{W}) = \prod_{i=1}^m P(w_i|w_{i-1}, \dots, w_{i-N+1})$
- De N -gramas a aproximaciones conexionistas:** Las redes neuronales recurrentes (**RNNs**) son ampliamente utilizadas para ajustar un modelo de $P(\mathbf{W})$
- El macromodelo λ integra los modelos acústico y de lenguaje
- La secuencia óptima de estados $\hat{\mathbf{q}}$ a partir de la cual se obtiene $\mathbf{W}$, se estima mediante el algoritmo de Viterbi:

$$\hat{\mathbf{q}} = \arg \max_{\mathbf{q}} p(\mathbf{q}, \mathbf{X}|\lambda)$$

RAH Híbrido DNN-HMM

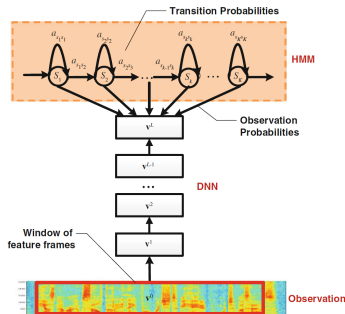
- Los sistemas DNN-HMM dependientes del contexto (CD) rinden significativamente mejor que los sistemas clásicos GMM-HMM en muchas tareas de reconocimiento continuo de voz de gran vocabulario (**LVCSR**):
 - Los nodos de salida de la DNN son senones (i.e., estados de trifonemas ligados) en lugar de estados de monofonemas



[Yu15] D. Yu and L. Deng, "Automatic Speech Recognition: A Deep Learning Approach". Springer, 2015

RAH Híbrido DNN-HMM

- *¡Importante!* **No** utilizamos una DNN por estado: se entrena una sola DNN para estimar la probabilidad a posteriori $p(q_t = s_j | \mathbf{x}_t)$ para todos los estados $\{s_j; j = 1, \dots, S\}$



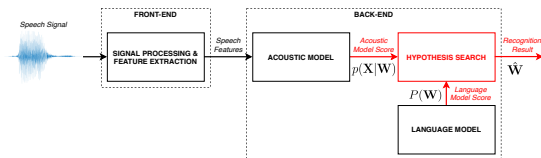
[Yu15] D. Yu and L. Deng, "Automatic Speech Recognition: A Deep Learning Approach". Springer, 2015

RAH Híbrido DNN-HMM

Decodificación en sistemas DNN-HMM:

- Como en GMM-HMM, $\hat{\mathbf{W}} = \arg \max_{\mathbf{W}} p(\mathbf{X}|\mathbf{W})P(\mathbf{W})$, donde la puntuación AM es $p(\mathbf{X}|\mathbf{W}) = \sum_{\mathbf{q}} \prod_{t=1}^T p(\mathbf{x}_t|q_t)P(q_t|q_{t-1})$
- En DNN-HMM,

$$p(\mathbf{x}_t|q_t = s_j) = \frac{p(q_t = s_j|\mathbf{x}_t)P(\mathbf{x}_t)}{P(s_j)} \Rightarrow \bar{p}(\mathbf{x}_t|q_t = s_j) = \frac{p(q_t = s_j|\mathbf{x}_t)}{P(s_j)}$$
- La probabilidad a priori de cada senón, $P(s_j) = T_{s_j}/T$, se estima a partir del conjunto de entrenamiento
- $p(q_t = s_j|\mathbf{x}_t)$ es dada por la DNN!



RAH Híbrido DNN-HMM

● Entrenando sistemas DNN-HMM (I):

- Algoritmo de Viterbi embebido ($\mathcal{S}$ es el conjunto de entrenamiento):

- ① $hmm0 \leftarrow \text{TrainCD-GMM-HMM}(\mathcal{S});$
- ② $stateAlignment \leftarrow \text{ForcedAlignmentWithGMMHMM}(\mathcal{S}, hmm0);$
- ③ $stateToSenoneIDMap \leftarrow \text{GenerateStateToSenoneIDMap}(hmm0);$
- ④ $featureSenoneIDPairs \leftarrow$
 $\text{GenerateDNNTrainingSet}(stateToSenoneIDMap, stateAlignment);$
- ⑤ $ptdnn \leftarrow \text{PretrainDNN}(\mathcal{S});$
- ⑥ $hmm \leftarrow \text{ConvertGMMHMMToDNNHMM}(hmm0, stateToSenoneIDMap);$
- ⑦ $prior \leftarrow \text{EstimatePriorProbability}(featureSenoneIDPairs);$
- ⑧ $dnn \leftarrow \text{Backpropagate}(ptdnn, featureSenoneIDPairs);$
- ⑨ Return $dnnhmm = \{dnn, hmm, prior\}$

- El algoritmo de Viterbi embebido minimiza la entropía cruzada promedio para cada segmento de voz con T tramas:

$$\mathcal{L}_{\text{CE}}(\theta) = - \sum_{t=1}^T \log p(q_t | \mathbf{x}_t; \theta)$$

RAH Híbrido DNN-HMM

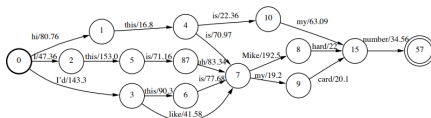
● Entrenando sistemas DNN-HMM (II):

- El criterio de entropía cruzada trata cada trama de forma independiente
- Sin embargo, ¡el RAH es un problema de clasificación de secuencias!
- Técnicas de entrenamiento discriminativas de la secuencia:
 - 1 Maximum mutual information (**MMI**)
 - 2 Boosted maximum mutual information (**BMMI**)
 - 3 Minimum phone error (**MPE**)
 - 4 State minimum Bayes risk (**sMBR**)

● Ejemplo: MMI

- MMI busca maximizar la información mutua entre las distribuciones de las secuencias de observaciones y palabras (*altamente correlacionado con minimizar el error de frase esperado*)

$$\mathcal{J}_{\text{MMI}}(\theta; \mathbb{S}) = \sum_{u=1}^{\mathcal{U}} \log P(\mathbf{W}^u | \mathbf{X}^u; \theta) = \sum_{u=1}^{\mathcal{U}} \log \frac{p(\mathbf{X}^u | \mathbf{s}^u; \theta)^\kappa P(\mathbf{W}^u)}{\sum_{\mathbf{W}} p(\mathbf{X}^u | \mathbf{s}^w; \theta)^\kappa P(\mathbf{W})}$$



[Mohri] M. Mohri, cs.nyu.edu/~mohri/asr12/lecture_12.pdf

RAH Híbrido DNN-HMM

● Entrenando sistemas DNN-HMM (II):

- El criterio de entropía cruzada trata cada trama de forma independiente
- Sin embargo, ¡el RAH es un problema de clasificación de secuencias!
- Técnicas de entrenamiento discriminativas de la secuencia:
 - ① Maximum mutual information (**MMI**)
 - ② Boosted maximum mutual information (**BMMI**)
 - ③ Minimum phone error (**MPE**)
 - ④ State minimum Bayes risk (**sMBR**)

Criterio de entrenamiento	WER (%)
GMM-BMMI	18,6
DNN-CE	14,2
DNN-MMI	12,9
DNN-BMMI	12,9
DNN-MPE	12,9
DNN-sMBR	12,6

Tasa de error de palabras (WER, %) sobre el conjunto de datos Switchboard, [Vesely13] K. Vesely *et al.*, "Sequence-discriminative training of deep neural networks". In Proc. of Interspeech 2013

RAH Híbrido DNN-HMM: Cuestiones Clave

- Directamente modelar senones es clave (*reducción del sobreajuste*):

Modelo	Monofonemas	Senones
CD-GMM-HMM	—	23,6
CD-DNN-HMM (7x2k)	34,9	17,1

Tasa de error de palabras (%) sobre el conjunto de datos Switchboard, [Seide11] F. Seide *et al.*, "Conversational Speech Transcription Using Context-Dependent Deep Neural Networks". In Proc. of Interspeech 2011

- ¡A mayor profundidad, mejor!

$L \times N$	WER (%)	$1 \times N$	WER (%)
1x2k	24,2	—	—
3x2k	18,4	—	—
5x2k	17,2	1x3.772	22,5
7x2k	17,1	1x4.634	22,6
—	—	1x16k	22,1

Tasa de error de palabras (WER, %) sobre el conjunto de datos Switchboard, [Seide11]

- Uso de contexto temporal:

Modelo	1 trama	11 tramas
CD-DNN-HMM (7x2k)	23,2	17,1

Tasa de error de palabras (WER, %) sobre el conjunto de datos Switchboard, [Seide11]

RAH Robusto

- Brecha de rendimiento entre humanos y máquinas debido a la discrepancia entre las condiciones de entrenamiento y evaluación de los sistemas de RAH:
 - Variabilidades debidas al locutor:** Intra-locutor (estado de ánimo, enfermedad...) e inter-locutor (longitud del tracto vocal, timbre...)
 - Variabilidades ambientales:** Ruido de fondo, reverberación...
- Compensación de la variabilidad debida al locutor:
 - Normalización de la longitud del tracto vocal (**VTLN**)
 - Regresión lineal de máxima verosimilitud en el espacio de las características (**fMLLR**)

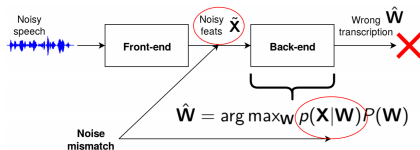
Modelo	Sin compensación	VTLN	fMLLR
CD-GMM-HMM	23,6	21,5	20,4
CD-MLP-HMM (1x2.048)	24,2	22,5	21,5
CD-DNN-HMM (7x2.048)	17,1	16,8	16,4

Tasa de error de palabras (%) sobre el conjunto de datos Switchboard, [Seide11b] F. Seide *et al.*, "Feature engineering in context-dependent deep neural networks for conversational speech transcription". In Proc. of ASRU 2011

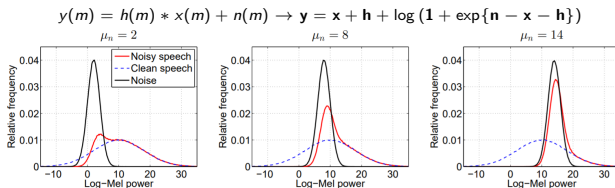
RAH Robusto

Compensación de la variabilidad ambiental:

- Ejemplo:* Modelos acústicos que se entrenan con datos de voz limpia pero se reconoce habla ruidosa → La discrepancia causará una transcripción errónea



- La distribución estadística de la energía de voz se ve afectada en presencia de ruido de fondo (cuando $\mathbf{h} = \mathbf{0}$):



RAH Robusto

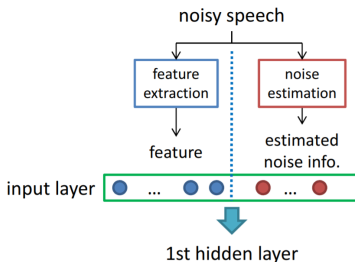
- Compensación de la variabilidad ambiental:

- **¡HAY MULTITUD DE APROXIMACIONES!**

- **Aproximaciones del espacio de las características:** Características robustas al ruido (RASTA-PLP, TANDEM...), normalización de momentos estadísticos de las características (CMN, HEQ...) y realce de voz/características (filtrado de Wiener, realce de voz basado en red neuronal, beamforming...)
- **Aproximaciones basadas en modelo:** Adaptación de modelos (CMLLR...) y entrenamiento adaptativo (fNAT, SAT...)
- **Compensación con modelado explícito de la distorsión:** Adaptación de modelos o compensación de características (VTS...)
- **Aproximaciones de datos perdidos:** Ignorar elementos no fiables durante el reconocimiento (marginalización, SFD...) e imputación de datos (TGI...)
- ...

RAH Robusto

- Compensación de la variabilidad ambiental:
 - **Entrenamiento multicondición:** Entrenar el modelo acústico con datos distorsionados de voz procedentes de diferentes condiciones acústicas (*¡muy efectivo si podemos cubrir las condiciones acústicas de evaluación!*)
 - **Noise-aware training (NAT):**



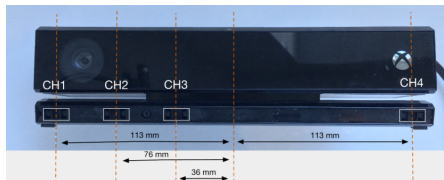
Sistema DNN-HMM (7x2.048)	WER (%)
Entrenamiento multicondición (MC)	13.4
MC+Realce de características	13.8
MC+NAT	13.1
MC+Dropout	12.9
MC+NAT+Dropout	12.4

Tasa de error de palabras (WER, %) sobre el conjunto de datos Aurora-4, [Seltzer13] M. Seltzer *et al.*, "An Investigation of Deep Neural Networks for Noise Robust Speech Recognition". In Proc. of ICASSP 2013

[Abe15] A. Abe *et al.*, "Robust Speech Recognition using DNN-HMM Acoustic Model Combining Noise-aware Training with Spectral Subtraction". In Proc. of Interspeech 2015

RAH de Extremo a Extremo

- **RAH de extremo a extremo:** Se entrena un modelo de aprendizaje profundo para, directamente, mapear una secuencia de entrada de características de voz a una secuencia de caracteres/*tokens*
- Los sistemas de RAH de extremo a extremo son más “simples”/limpios: No hay necesidad de modelos acústicos y de lenguaje específicos con diccionarios de pronunciación
- **Reto CHiME-6:** RAH conversacional usando micrófonos distantes en entornos domésticos cotidianos (<https://chimechallenge.github.io/chime6/overview.html>)



- En CHiME-6, los sistemas de RAH híbridos DNN-HMM aún superaban a las aproximaciones de RAH de extremo a extremo ([jen 2020!](#))

RAH de Extremo a Extremo: Conceptos Básicos

Redes neuronales recurrentes (RNNs)

- RNNs estándar (idea general):

$$\mathbf{h}_t = \sigma(\mathbf{W}_{ih}\mathbf{x}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{b}_h)$$

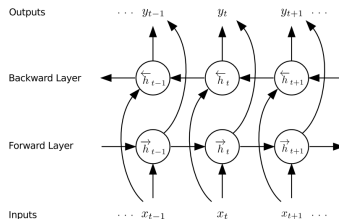
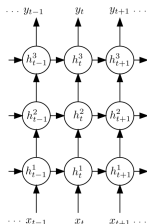
$$\mathbf{y}_t = \mathbf{W}_{ho}\mathbf{h}_t + \mathbf{b}_o$$

- RNNs bidireccionales:

$$\vec{\mathbf{h}}_t = \sigma(\mathbf{W}_{i\vec{h}}\mathbf{x}_t + \mathbf{W}_{\vec{h}\vec{h}}\vec{\mathbf{h}}_{t-1} + \mathbf{b}_{\vec{h}}) \quad \overleftarrow{\mathbf{h}}_t = \sigma(\mathbf{W}_{i\overleftarrow{h}}\mathbf{x}_t + \mathbf{W}_{\overleftarrow{h}\overleftarrow{h}}\overleftarrow{\mathbf{h}}_{t+1} + \mathbf{b}_{\overleftarrow{h}})$$

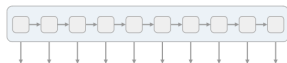
$$\mathbf{y}_t = \mathbf{W}_{\vec{h}o}\vec{\mathbf{h}}_t + \mathbf{W}_{\overleftarrow{h}o}\overleftarrow{\mathbf{h}}_t + \mathbf{b}_o$$

- Long short-term memory (LSTM), LSTM bidireccional (BiLSTM), unidades recurrentes cerradas (GRUs)



[Graves14] A. Graves and N. Jaitly, "Towards End-to-End Speech Recognition with Recurrent Neural Networks". In Proc. of ICML 2014

RAH de Extremo a Extremo: CTC



h	h	h	h	h	h	h	h	h	h
e	e	e	e	e	e	e	e	e	e
l	l	l	l	l	l	l	l	l	l
o	o	o	o	o	o	o	o	o	o
€	€	€	€	€	€	€	€	€	€

h	e	€	l	l	€	l	l	o	o
h	h	e	l	l	€	€	l	€	o
€	e	€	l	l	€	€	l	o	o

h	e	l	l	o
e	l	l	o	
h	e	l	o	

- Sea $\mathbf{C} = (c_1, \dots, c_m)$ la secuencia de caracteres/*tokens* correspondiente a $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_T)$
- Desconocemos un alineamiento preciso entre $\mathbf{C}$ y $\mathbf{X}$, y, además, $m < T$
- **Clasificación temporal conexionista (CTC)** es un algoritmo que *no requiere alineamiento*
- CTC define el *token* vacío (ϵ)
- **Objetivo de CTC:** Maximizar $P(\mathbf{C}|\mathbf{X}) = \sum_{A \in \mathcal{A}_{X,C}} \prod_{t=1}^T P_t(c|\mathbf{X})$ (p. ej., $c = \{h, e, l, o, \epsilon\}$)
- Decodificación como de costumbre, $\hat{\mathbf{C}} = \arg \max_{\mathbf{C}} P(\mathbf{C}|\mathbf{X})$

[Hannun] A. Hannun, <https://distill.pub/2017/ctc/>

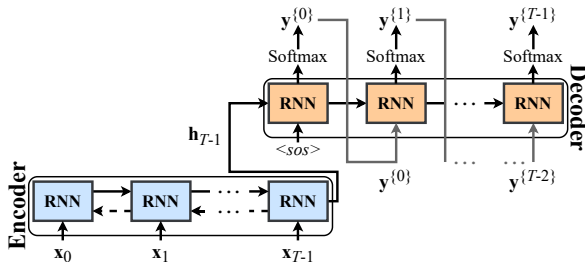
RAH de Extremo a Extremo: Codificador-Decodificador

Arquitectura codificador-decodificador

- El codificador es normalmente una BiLSTM, mientras que el decodificador, una LSTM:

$$\mathbf{h}_t = \text{Encoder}(\mathbf{x}_t, \mathbf{h}_{t-1})$$

$$\mathbf{s}_i = \text{Decoder}(\mathbf{s}_{i-1}, \mathbf{y}_{i-1})$$



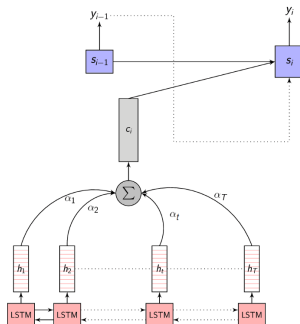
[Espejo21] I. López-Espejo et al., "Deep Spoken Keyword Spotting: An Overview". IEEE Access, 2021

- Problema potencial:** El codificador necesita condensar toda la información requerida (independientemente de la longitud de la secuencia de entrada) en un vector de dimensión fija

RAH de Extremo a Extremo: Atención

- Podemos **prestar atención** a un subconjunto de $\{\mathbf{h}_1, \dots, \mathbf{h}_T\}$ relevante al contexto en lugar de a $\mathbf{h}_T$ para “ayudar” al decodificador:

$$\mathbf{s}_i = \text{Decoder}(\mathbf{s}_{i-1}, \mathbf{y}_{i-1}, \mathbf{C}_i)$$



[Nadig] S. Nadig, <https://medium.com/intel-student-ambassadors/attention-in-end-to-end-automatic-speech-recognition-9f9e42718d21>

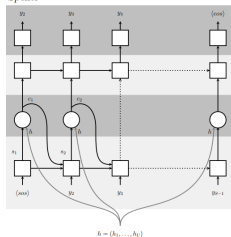
<https://medium.com/intel-student-ambassadors/attention-in-end-to-end-automatic-speech-recognition-9f9e42718d21>

$$\mathbf{C}_i = \sum_{t=1}^T \alpha_{it} \mathbf{h}_t$$

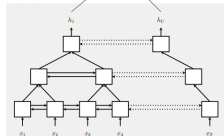
$$\alpha_{it} = \text{softmax}(\text{AttentionFunction}(\mathbf{s}_{i-1}, \mathbf{h}_t))$$

RAH de Extremo a Extremo: Cod.-Dec. con Atención

Speller



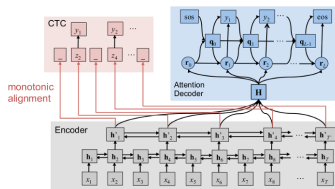
Listener



[Chan16] W. Chan et al., "Listen, Attend and Spell: A Neural Network for Large Vocabulary Conversational Speech Recognition". In Proc. of ICASSP 2016

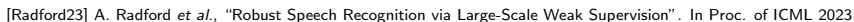
- **Problema con CTC:** Independencia de las etiquetas condicionales durante la decodificación → CTC requiere de un modelo de lenguaje externo para funcionar bien
- **RAH codificador-decodificador basado en atención:** El alineamiento entre $\mathbf{C}$ y $\mathbf{X}$ se aprende usando atención $P(\mathbf{C}|\mathbf{X}) = \prod_{i=1}^m P(c_i|\mathbf{X}, c_1, \dots, c_{i-1})$
- El RAH codificador-decodificador basado en atención es menos robusto al ruido que el RAH basado en CTC → El **RAH basado en CTC y atención** es efectivo para mejorar el reconocimiento:

$$\text{Multitask learning: } \mathcal{L}_{\text{MTL}} = \lambda \mathcal{L}_{\text{CTC}} + (1 - \lambda) \mathcal{L}_{\text{Attention}}$$

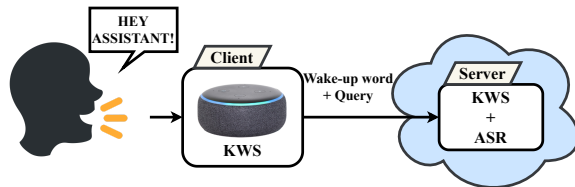
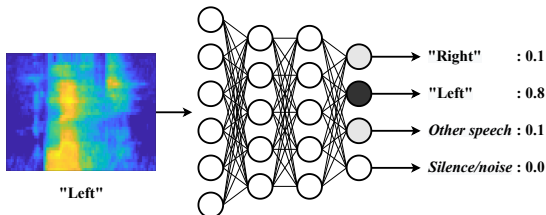


CTC guides attention alignment to be monotonic

[Nadig] S. Nadig, <https://medium.com/intel-student-ambassadors/attention-in-end-to-end-automatic-speech-recognition-9f9e42718d21>



Detección de Palabras Clave



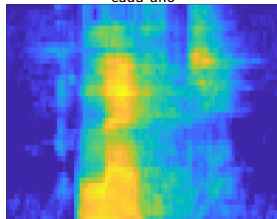
- El control por voz se implementa típicamente mediante tecnología de **detección de palabras clave (KWS)**
- KWS se puede definir como la tarea de identificar palabras clave en flujos de audio que contienen voz (subproblema del RAH)

[Espejo21] I. López-Espejo et al., "Deep Spoken Keyword Spotting: An Overview". IEEE Access, 2021

Google Speech Commands Dataset

- **Google Speech Commands Dataset**^[Warden18], disponible públicamente, se convirtió en el punto de referencia abierto *de facto* para el desarrollo y la evaluación de KWS
 - *Frecuencia de muestreo de 16 kHz*
 - *Grabada mediante micrófonos de ordenadores portátiles y teléfonos*
 - *Ruidosa hasta cierto punto*

Segmentos de voz de un segundo de duración que contienen una palabra cada uno



Versión	Locutores	Palabras	Segmentos
v1	1.881	30	64.727
v2	2.618	35	105.829

[Warden18] P. Warden, "Speech Commands: A dataset for limited-vocabulary speech recognition". *arXiv:1804.03209v1*, 2018

Google Speech Commands Dataset

Version 1 (v1)	Version 2 (v2)	yes	no	up	down	left	KW
		right	on	off	stop	go	
		zero	one	two	three	four	Non-KW
		five	six	seven	eight	nine	
		bed	bird	cat	dog	happy	
		house	Marvin	Sheila	tree	wow	
		backward	forward	follow	learn	visual	

Este punto de referencia también estandariza...

- ...los conjuntos de entrenamiento, desarrollo y evaluación
- ...un procedimiento de **aumento de datos** de entrenamiento que involucra ruidos de fondo
- ...

Métricas de Evaluación: Precisión

- **Precisión:** Cociente entre el número de predicciones correctas y el total de predicciones

$$\text{Precisión} \equiv \frac{TP + TN}{TP + TN + FP + FN} \in [0, 1]$$

- La precisión suele ser una métrica de evaluación *inapropiada*, pues puede conducir a conclusiones equívocas

Ground truth NK NK KW NK NK KW NK NK NK NK

SYS1

NK	NK	KW	NK	NK	NK	KW	NK	NK	NK
----	----	----	----	----	----	----	----	----	----

SYS2

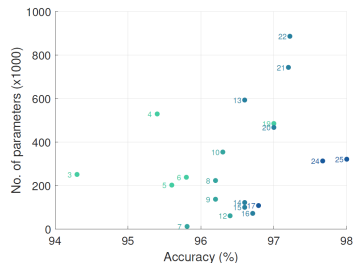
NK	NK	NK	NK	NK	NK	NK	NK	NK	NK
----	----	----	----	----	----	----	----	----	----

Detección de Palabras Clave (II)

Rendimiento sobre Google Speech Commands Dataset:

ID	Description	Year	Accuracy (%)		Computational complexity	
			GSCD v1	GSCD v2	No. of params.	No. of mults.
1	Standard FFNN with a pooling layer [32]	2020	91.2	90.6	447k	–
2	DenseNet with trainable window function and mixup data augmentation [67]	2018	92.8	–	–	–
3	Two-stage TDNN [58]	2018	94.3	–	251k	25.1M
4	CNN with striding [32]	2018	95.4	95.6	529k	–
5	BiLSTM with attention [133]	2018	95.6	96.9	202k	–
6	Residual CNN vasilis [30]	2018	95.8 ± 0.484	–	238k	894M
7	TDNN with shared weight self-attention [16]	2019	95.81 ± 0.191	–	12k	403k
8	DenseNet+BiLSTM with attention [48]	2019	96.2	97.3	223k	–
9	Residual CNN with temporal convolutions TC-ResNet14 [50]	2019	96.2	–	137k	–
10	SVDF [32]	2019	96.3	96.9	354k	–
11	SincCom+(Grouped DS-CNN) [70]	2020	96.4	97.3	62k	–
12	Graph convolutional network GCNet-40 [49]	2019	96.4	–	61k	16.18M
13	GRU [32]	2020	96.6	97.2	593k	–
14	SincCom+(DS-CNN) [70]	2020	96.6	97.4	122k	–
15	Temporal CNN with depthwise convolutions TENet12 [52]	2020	96.6	–	100k	2.90M
16	Residual DS-CNN with squeeze-and-excitation DS-ResNet18 [51]	2020	96.71 ± 0.195	–	72k	285M
17	TC-ResNet14 with neural architecture search NoisyDARTS-TC14 [146]	2021	96.79 ± 0.30	97.18 ± 0.26	108k	6.3M
18	LSTM [32]	2020	96.9	97.5	–	–
19	DS-CNN with striding [32]	2018	97.0	97.1	485k	–
20	CRNN [32]	2020	97.0	97.5	467k	–
21	BiGRU with multi-head attention [32]	2020	97.2	98.0	743k	–
22	CNN with neural architecture search NAS2_6_36 [125]	2020	97.22	–	886k	–
23	Keyword Transformer KWT-3 [90]	2021	97.49 ± 0.15	98.56 ± 0.07	5.3M	–
24	Variant of TC-ResNet with self-attention LG-Net6 [91]	2021	97.67	96.79	313k	–
25	Broadcasted residual CNN BC-ResNet-8 [100]	2021	98.0	98.7	321k	89.1M

- El modelado acústico del estado del arte está basado en **redes neuronales convolucionales (CNNs)**



- Para alcanzar un **alto rendimiento** con una **pequeña huella computacional**, un modelo acústico basado en CNN debería considerar...

- 1 Un mecanismo para explotar dependencias de T-F a largo plazo (p. ej., convoluciones dilatadas)
- 2 Convoluciones separables en profundidad
- 3 Conexiones residuales

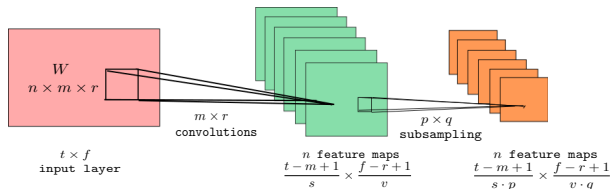
Tabla de Contenidos

1 Introducción al Reconocimiento del Habla

2 Implementación de un Sistema de Control por Voz

Redes Neuronales Convolucionales: Introducción

- Del perceptrón multicapa a las redes neuronales convolucionales → *Un paso natural dado en 2015 para KWS*^[Sainath15]
 - *Explotación de correlaciones locales de tiempo-frecuencia, menos parámetros*



[Sainath15] T. N. Sainath and C. Parada, "Convolutional neural networks for small-footprint keyword spotting". In Proc. of Interspeech 2015

- El número de multiplicaciones del modelo se puede limitar fácilmente para cumplir con las restricciones computacionales:
 - *Stride del filtro, tamaño del kernel, tamaño del pooling...*

Redes Neuronales Convolucionales: Introducción

- Apropriadas para procesar datos con una topología reticular
 - Datos de series temporales como la forma de onda de la voz (1D)
 - Imágenes (2D o 3D)
 - ...

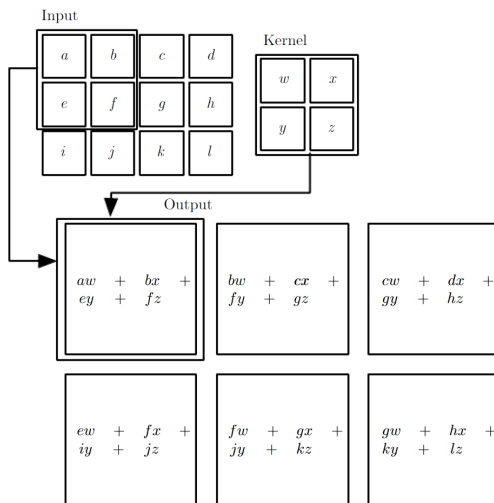
$$s(t) = (x * w)(t) = \int x(a)w(t - a)da$$

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t - a)$$

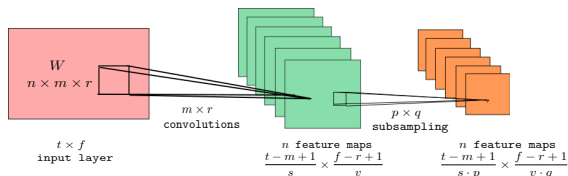
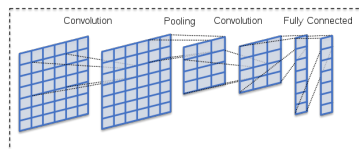
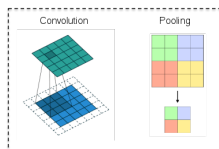
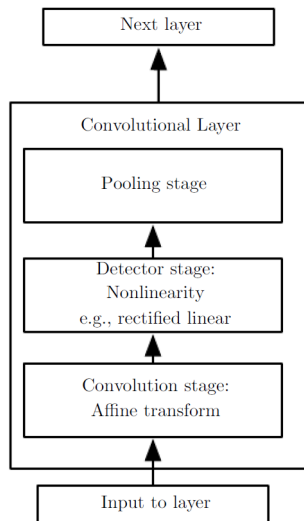
$$\begin{aligned} S(i, j) &= (I * K)(i, j) = \sum_m \sum_n I(m, n)K(i - m, j - n) \\ &= \sum_m \sum_n I(i - m, j - n)K(m, n) \end{aligned} \quad (\text{convolución 2D})$$

$$S(i, j) = (I * K)(i, j) = \sum_m \sum_n I(i + m, j + n)K(m, n) \quad (\text{correlación cruzada 2D})$$

Redes Neuronales Convolucionales: Convolución 2D

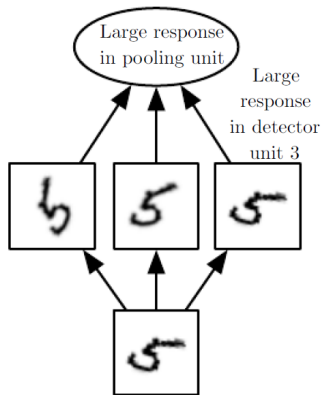
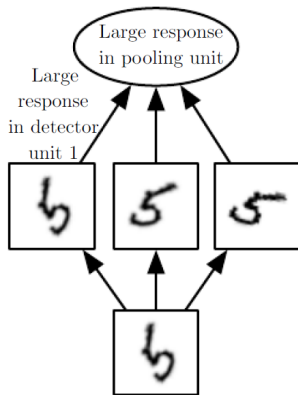


Redes Neuronales Convolucionales: Arquitectura General



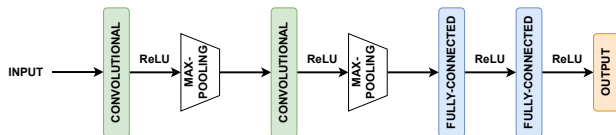
El **pooling** genera invarianza a la traslación local

Redes Neuronales Convolucionales: Ejemplo de Operación



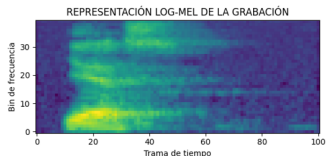
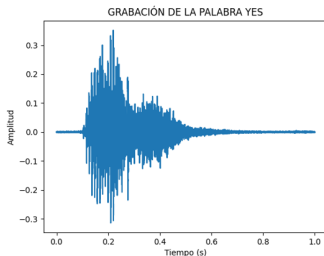
Reconocedor Simple de Voz

- **PyTorch** para el desarrollo de modelos de aprendizaje máquina y redes neuronales
- Construiremos un **reconocedor simple de palabras basado en red neuronal convolucional**:



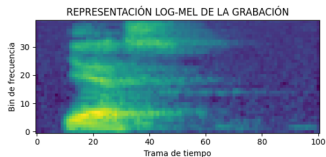
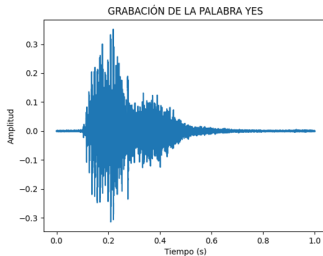
- Primera y segunda capas convolucionales usan 32 y 16 mapas de características, respectivamente
- **Capas convolucionales:** Kernel de 5×5 , *stride* de 1×1 y no se emplea relleno
- **Max-pooling:** *Pooling* de 2×2 y *stride* de 2×2
- **Capas completamente conectadas:** 128 neuronas cada una a excepción de la de salida (3, pues reconoceremos “YES”, “NO” y **otra palabra**)

Datos de Trabajo



- **Datos.zip** contiene los ficheros `pickle` de Python siguientes:
 - ① **X_train.pkl**: 3.426 matrices de características log-Mel para entrenamiento
 - ② **Y_train.pkl**: Etiquetas de referencia para X_train.pkl en codificación *one-hot*
 - ③ **X_valid.pkl**: 393 matrices de características log-Mel para validación
 - ④ **Y_valid.pkl**: Etiquetas de referencia para X_valid.pkl en codificación *one-hot*
 - ⑤ **X_test.pkl**: 486 matrices de características log-Mel para evaluación
 - ⑥ **Y_test.pkl**: Etiquetas de referencia para X_test.pkl en codificación *one-hot*

Datos de Trabajo



<i>Clase / Conjunto de datos</i>	<i>Entrenamiento</i>	<i>Validación</i>	<i>Evaluación</i>
“YES”	1.137	130	155
“NO”	1.147	132	169
Otra palabra	1.142	131	162
TOTAL	3.426	393	486

Poniendo a Punto el Entorno de Trabajo

- Ya instalaste torch (es decir, PyTorch), numpy (para cálculo científico), matplotlib (para la creación de visualizaciones), etc.
- **Necesitamos instalar librerías adicionales:**
 - ① Abre Anaconda Prompt y activa tu entorno de trabajo con el comando `conda activate cursoVoz`
 - ② Instala los siguientes módulos haciendo uso de pip: `pickle` (para serialización y deserialización de estructuras objeto de Python), `librosa` (para análisis de audio) y `scipy` (para cálculo técnico y científico)

Entrenamiento y Evaluación del Reconocedor de Palabras

- Creamos un fichero de código Python `Ej_CNN_CPU.py`
- **Importamos los módulos** que vamos a necesitar:

```
import torch
from torch.utils.data import TensorDataset
from torch.utils.data import DataLoader
import pickle
import numpy as np
import matplotlib.pyplot as plt
import scipy.io as sio
```

Entrenamiento y Evaluación del Reconocedor de Palabras

- Creamos la **clase** que **definirá** nuestro **modelo** de **red neuronal**:

```
class Convolutional(torch.nn.Module):
    def __init__(self, no_channels, kernel_size, pooling_size, flattened_size, hidden_size, output_size):
        super(Convolutional, self).__init__()
        self.no_channels = no_channels
        self.kernel_size = kernel_size
        self.pooling_size = pooling_size
        self.flattened_size = flattened_size
        self.hidden_size = hidden_size
        self.output_size = output_size
        self.conv1 = torch.nn.Conv2d(1, self.no_channels, self.kernel_size)
        self.conv2 = torch.nn.Conv2d(self.no_channels, int(self.no_channels/2), self.kernel_size)
        self.pool = torch.nn.MaxPool2d(self.pooling_size, self.pooling_size)
        self.fc1 = torch.nn.Linear(self.flattened_size, self.hidden_size)
        self.fc2 = torch.nn.Linear(self.hidden_size, self.hidden_size)
        self.fc3 = torch.nn.Linear(self.hidden_size, self.output_size)
        self.relu = torch.nn.ReLU()

    def forward(self, x):
        out = self.conv1(x)
        out = self.relu(out)
        out = self.pool(out)
        out = self.conv2(out)
        out = self.relu(out)
        out = self.pool(out)
        out = out.view(out.size(0), 1, self.flattened_size)
        out = self.fc1(out)
        out = self.relu(out)
        out = self.fc2(out)
        out = self.relu(out)
        out = self.fc3(out)
        return out
```

Entrenamiento y Evaluación del Reconocedor de Palabras

- Incluimos un método para **calcular la precisión de reconocimiento de palabra** de cara a evaluar el rendimiento de nuestra red neuronal:

```
def compute_accuracy(y_pred, y_labels):  
    y_pred = y_pred.data.numpy()  
    y_pred = y_pred.argmax(axis=2)  
    acc = y_pred[:,0] - y_labels.data.numpy()  
    return 100*np.sum(acc==0)/len(acc)
```

Entrenamiento y Evaluación del Reconocedor de Palabras

● Preparación de los datos para entrenamiento discriminativo:

- ① Carga de las matrices log-Mel y sus etiquetas
- ② Definición de la dimensión de los canales/mapas de características
- ③ Normalización de los datos de entrenamiento
- ④ *Etiquetas*: de *one-hot* a números enteros
- ⑤ Conversión a tensores y estructuración del conjunto de entrenamiento

```
print('Preparando datos...')
# Cargamos los datos de entrenamiento.
X_train = pickle.load(open('X_train.pkl', 'rb'))
Y_train_oh = pickle.load(open('Y_train.pkl', 'rb'))
# Reorganizando los datos.
no_ts = X_train.shape[0] # No. de muestras de entrenamiento.
no_fb = X_train.shape[1] # No. de bins de frecuencia.
no_tf = X_train.shape[2] # No. de tramas de tiempo.
X_train = np.expand_dims(X_train,axis=1) # Definimos la dimensión de los canales.
# Normalización de los datos.
uX = np.mean(X_train)
sX = np.std(X_train)
X_train = (X_train - uX) / sX
# De one-hot encoding a enteros.
Y_train = np.zeros(no_ts)
for i in range(no_ts):
    Y_train[i] = np.where(Y_train_oh[i]==1)[0][0]
# A tensores de PyTorch.
X_train = torch.FloatTensor(X_train)
Y_train = torch.LongTensor(Y_train)
train_data = TensorDataset(X_train, Y_train)
train_loader = DataLoader(dataset=train_data, batch_size=128, shuffle=True)
```


Entrenamiento y Evaluación del Reconocedor de Palabras

- Procedimiento similar para los **conjuntos de validación y evaluación**:

```
# Datos de validación.
X_valid = pickle.load(open('X_valid.pkl', 'rb'))
Y_valid_oh = pickle.load(open('Y_valid.pkl', 'rb'))
no_vs = X_valid.shape[0] # No. de muestras de validación.
X_valid = np.expand_dims(X_valid,axis=1)
X_valid = (X_valid - uX) / sX
Y_valid = np.zeros(no_vs)
for i in range(no_vs):
    Y_valid[i] = np.where(Y_valid_oh[i]==1)[0][0]
X_valid = torch.FloatTensor(X_valid)
Y_valid = torch.LongTensor(Y_valid)
```

```
# Datos de test.
X_test = pickle.load(open('X_test.pkl', 'rb'))
Y_test_oh = pickle.load(open('Y_test.pkl', 'rb'))
no_es = X_test.shape[0] # No. de muestras de test.
X_test = np.expand_dims(X_test,axis=1)
X_test = (X_test - uX) / sX
Y_test = np.zeros(no_es)
for i in range(no_es):
    Y_test[i] = np.where(Y_test_oh[i]==1)[0][0]
X_test = torch.FloatTensor(X_test)
Y_test = torch.FloatTensor(Y_test)
```

Entrenamiento y Evaluación del Reconocedor de Palabras

- Creamos una instancia u objeto de nuestra clase `Convolutional()` que define la arquitectura de la red
- **Función de pérdida/coste:** Entropía cruzada categórica (*de uso típico en problemas de clasificación*)
- **Optimizador:** Adam con parámetros por defecto

```
print('Creando el modelo...')
no_cl = 3 # No. de clases ("yes", "no" y "otro").
pooling = 2 # Tamaño del pooling.
ksize = 5 # Tamaño del kernel.
no_ch = 32 # Número de canales.
model = Convolutional(no_ch, ksize, pooling, flattened_size, 128, no_cl)
criterion = torch.nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(model.parameters())
```

Entrenamiento y Evaluación del Reconocedor de Palabras

- **Entrenamos nuestro modelo** por un total de 12 épocas
- Almacenamos, por cada *iteración*, valores de pérdida y precisión

```
print('Entrenando el modelo...')
model.train()
no_epoch = 12 # No. de épocas de entrenamiento.
train_loss = []
val_loss = []
train_acc = []
val_acc = []
for epoch in range(no_epoch):
    # Procesamiento por mini-batch.
    mi = 1
    for x_batch, y_batch in train_loader:
        optimizer.zero_grad()
        y_pred = model(x_batch) # Pasada hacia delante.
        loss = criterion(y_pred.squeeze(), y_batch)
        # Para validación.
        y_pred_val = model(X_valid)
        loss_val = criterion(y_pred_val.squeeze(), Y_valid)
        print('Época {}, Mini-batch: {}, Pérdida de entrenamiento: {}, Pérdida de validación:
        {}'.format(epoch+1, mi, loss.item(), loss_val.item()))
    loss.backward() # Pasada hacia atrás.
    optimizer.step()
    train_loss.append(loss.data.numpy())
    val_loss.append(loss_val.data.numpy())
    train_acc.append(compute_accuracy(y_pred, y_batch))
    val_acc.append(compute_accuracy(y_pred_val, Y_valid))
    mi += 1
```

Entrenamiento y Evaluación del Reconocedor de Palabras

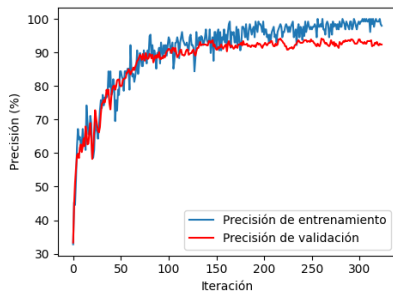
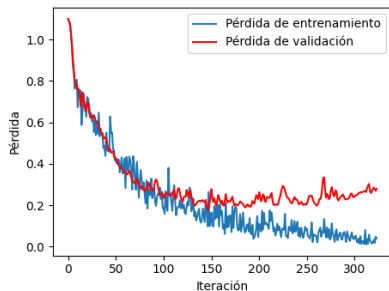
- **Mostramos por pantalla las curvas de pérdida y precisión de reconocimiento de palabra** en función de la iteración:

```
# Pintamos las curvas de pérdida.
fig, ax = plt.subplots()
ax.plot(train_loss, label='Pérdida de entrenamiento')
ax.plot(val_loss, 'r', label='Pérdida de validación')
ax.set_xlabel('Iteración')
ax.set_ylabel('Pérdida')
ax.legend()
plt.show()

# Pintamos las curvas de precisión.
fig2, ax2 = plt.subplots()
ax2.plot(train_acc, label='Precisión de entrenamiento')
ax2.plot(val_acc, 'r', label='Precisión de validación')
ax2.set_xlabel('Iteración')
ax2.set_ylabel('Precisión (%)')
ax2.legend()
plt.show()
```

Entrenamiento y Evaluación del Reconocedor de Palabras

- A partir de la iteración 200, la curva de pérdida de validación tiende a crecer:



- Síntoma de **sobreajuste**
- Parada temprana (*early stopping*)

Entrenamiento y Evaluación del Reconocedor de Palabras

- Medimos el **rendimiento** del reconocedor **sobre el conjunto de evaluación**:

```
model.eval()
y_pred = model(X_test)
acc = compute_accuracy(y_pred, Y_test)
print('Precisión de evaluación: ' + str(acc) + '%')
```

- Salvamos el modelo de red neuronal y los estadísticos de normalización de las matrices de características log-Mel:

```
torch.save(model.state_dict(), 'Mi_Reconocedor.pth')
sio.savemat('Media.mat', {'uX': uX})
sio.savemat('Desv.mat', {'sX': sX})
```

Demo Live

- Implementaremos un código muy básico con el que **podremos grabarnos diciendo una palabra y reconocerla de forma automática**
- Creamos un nuevo fichero de código Python llamado `Demo_Live.py` e importamos los módulos que vamos a requerir:

```
import sounddevice as sd
import time
import torch
import librosa
import numpy as np
import matplotlib.pyplot as plt
import scipy.io as sio
```

Demo Live

- Tenemos que instanciar de nuevo nuestra clase `Convolutional()` para cargar los pesos de nuestro modelo entrenado:
 - Bien copiamos y pegamos el código de la clase en `Demo_Live.py`...
 - o bien disponemos la clase en un fichero de código Python separado para importar
- Establecemos los parámetros de grabación y **cargamos el modelo de red neuronal** entrenado:

```
fs = 16000 # Frecuencia de muestreo de grabación en Hz.  
seconds = 1 # Duración de la grabación.  
  
# Cargamos el reconecedor entrenado.  
model = Convolutional(32, 5, 2, 2464, 128, 3)  
model.load_state_dict(torch.load('Mi_Reconocedor.pth', weights_only=True))  
model.eval()
```


Demo Live

- **Grabamos una muestra de voz** de 1 segundo de duración a 16 kHz de frecuencia de muestreo **y la representamos**:

```
# Grabamos la muestra de audio.  
print('La grabación comienza en')  
print('3')  
time.sleep(1)  
print('2')  
time.sleep(1)  
print('1')  
time.sleep(1)  
print('Grabando...')  
grab = sd.rec(int(seconds * fs), samplerate=fs, channels=1)  
sd.wait()  
print('¡Listo!')
```

```
# Representamos la señal grabada.  
plt.figure()  
plt.plot(np.arange(0,fs)/(fs-1), grab)  
plt.xlabel('Tiempo (s)')  
plt.ylabel('Amplitud')  
plt.title('Señal grabada')  
plt.show()
```

Demo Live

- **Extraemos la matriz 40×101 de características log-Mel** a partir de la grabación de 1 segundo
- Normalizamos la matriz de características

```
# Extraemos la matriz log-Mel.
grab = grab[:,0]
grab += 1e-3 * np.random.randn(len(grab)) # Añadimos ruido blanco gaussiano de fondo.
# 40 canales, ventana de Hann de 32 ms y salto de 10 ms.
mel = librosa.feature.melspectrogram(y=grab, sr=fs, n_mels=40, n_fft=512,
hop_length=160)
lme1 = np.log(mel) # Matriz de características log-Mel final.
uX = sio.loadmat('Media.mat')['uX'][0][0]
sX = sio.loadmat('Desv.mat')['sX'][0][0]
lme1 = (lme1 - uX) / sX # Normalización de la matriz log-Mel.
```

Demo Live

- **Predecimos la palabra contenida en la grabación** y mostramos el resultado por pantalla:

```
# Hacemos la predicción.
lme1 = np.expand_dims(np.expand_dims(lme1, axis=0), axis=0) # Añadimos la dimensión de los canales.
entrada = torch.FloatTensor(lme1)
y_pred = model(entrada)
y_pred = np.argmax(y_pred.data.numpy())
```

```
# Mostramos el resultado por pantalla.
if y_pred == 0:
    print('La palabra reconocida es YES!')
elif y_pred == 1:
    print('La palabra reconocida es NO!')
else:
    print('No se ha reconocido ni YES ni NO!')
```

¿Ha acertado? 

Número de Parámetros de la Red Neuronal

- ¿Serías capaz de calcular manualmente el número de parámetros de tu modelo de red neuronal?
- Puedes comprobar si has hecho correctamente los cálculos apoyándote en el siguiente fragmento de código:

```
def get_no_params(model):  
    nop = 0  
    for param in list(model.parameters()):  
        nn = 1  
        for s in list(param.size()):  
            nn = nn * s  
        nop += nn  
    return nop  
  
no_params = get_no_params(model)  
print('El número de parámetros de mi modelo es ' + str(no_params))
```

Introducción a las Tecnologías del Habla

Dr. Iván López-Espejo

DÍA 3: INTRODUCCIÓN AL RECONOCIMIENTO DEL HABLA

iloes@ugr.es

Wednesday 4th June, 2025



PONTIFICIA UNIVERSIDAD
CATOLICA
DE VALPARAISO



**UNIVERSIDAD
DE GRANADA**