UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO INGENIERÍA ELÉCTRICA

RECONOCIMIENTO ROBUSTO DE VOZ EN SISTEMAS DNN-HMM UTILIZANDO PARAMETRIZACIÓN NORMALIZADA LOCALMENTE

TESIS PARA OPTAR AL GRADO DE MAGÍSTER EN CIENCIAS DE LA INGENIERÍA, MENCIÓN ELÉCTRICA

JOSUÉ ABRAHAM FREDES SANDOVAL

PROFESOR GUÍA:
NÉSTOR BECERRA YOMA

MIEMBROS DE LA COMISIÓN:
JORGE SILVA SÁNCHEZ
MATÍAS ZAÑARTU SALAS

SANTIAGO DE CHILE
2023

RESUMEN DE LA TESIS PARA OPTAR
AL GRADO DE MAGÍSTER EN CIENCIAS
DE LA INGENIERÍA, MENCIÓN ELÉCTRICA
POR: JOSUÉ ABRAHAM FREDES SANDOVAL
FECHA: 2023
PROF. GUÍA: DR. NÉSTOR BECERRA YOMA

RECONOCIMIENTO ROBUSTO DE VOZ EN SISTEMAS DNN-HMM UTILIZANDO PARAMETRIZACIÓN NORMALIZADA LOCALMENTE

En esta tesis se describe las modificaciones a la parametrización de bancos de filtros normalizados localmente que mejoran sustancialmente su rendimiento en la tarea de reconocimiento robusto de voz en la base de datos Aurora-4, utilizando un sistema basado en *Deep Neural Network – Hidden Markov Models* (DNN-HMM). Los coeficientes modificados, denominados *Locally Normalized Filter Banks* (LNFB), son una versión de banco de filtros de los *Locally Normalized Cepstral Coefficients* (LNCC) , propuestos anteriormente.

El uso de LNFB proporciona una reducción relativa media de la tasa de error de 11,4 % y 9,4 % en comparación con el caso *baseline* en condiciones de entrenamiento limpio y multi-ruido. Los resultados presentados aquí sugieren que LNFB es más robusta a diferencias de canal entre datos de entrenamiento y prueba , y es más eficaz para enfrentar la diversidad de canal.

A continuación se estudia la complementariedad de sistemas DNN-HMM entrenados sobre una misma base de datos, pero con distintas parametrizaciones. Se entrenaron sistemas usando cuatro parametrizaciones distintas aplicando en cada caso la técnica de corrección *Weighted Predictor Error*, sobre una base de datos reverberante. La combinación de sistemas lleva a un 17,6 % de reducción relativa de la tasa de error respecto al mejor sistema sin combinar.

Tabla de Contenido

Índice de tablas

Índice de figuras

Capítulo 1

Introducción

1.1. Motivación

El reconocimiento automático de voz, o *Automatic Speech Recognition* (ASR), es una tarea que consiste en procesar la señal de voz de un humano y convertirla a una secuencia de texto, utilizando medios computacionales. Dentro de estos reconocedores existe una clase llamada *Continuous Speech Recognition* o CSR, que tiene como objetivo analizar señales que contienen frases completas, y que además están conformadas por vocabularios que pueden contener miles de palabras distintas.

Uno de los desafíos en ASR consiste en diseñar un sistema robusto a variabilidades y distorsiones que ocurren en las condiciones acústicas en las que funciona el sistema. Para ello, existen diferentes técnicas y métodos que permiten que el sistema pueda mejorar su rendimiento en condiciones adversas.

Uno de los métodos que ha recientemente traído mejoras dramáticas en el rendimiento es el uso de redes neuronales profundas o DNNs (*Deep Neural Networks*). Una DNN puede aprender a representar de manera robusta la señal de voz, traspasando esa robustez al sistema ASR. Sin embargo, aún se pueden aplicar técnicas de robustez en otras etapas del proceso de ASR que contribuyen a mejorar el rendimiento global del sistema y a reducir los requerimientos del tamaño de la base de datos de entrenamiento.

En este trabajo se estudia el efecto de la parametrización aplicada a las señales de voz en el rendimiento de un sistema ASR que tiene un modelo acústico basado en DNN. En particular, se propone y analiza el rendimiento de una parametrización llamada *Locally Normalized Filter Bank* (LNFB), que consiste en aplicar una normalización local en el espectro de potencia de la señal de voz. LNFB debiera ser robusta al *mismatch* o discordancia en las respuestas en frecuencia de los micrófonos, que es un tipo de distorsión en las condiciones acústicas en las que funciona el sistema.

En este trabajo se estudia el efecto de la parametrización aplicada a las señales de voz en el rendimiento de un sistema ASR que tiene un modelo acústico basado en DNN. En particular, se propone y analiza el rendimiento de una parametrización llamada *Locally Normalized Filter Bank* (LNFB) que está basada en una parametrización propuesta anteriormente llamada LNCC [1].

LNFB consiste en aplicar una normalización local en el espectro de potencia de la señal de voz. Esta representación debería proveer robustez ante el *tilt* espectral que ocurre principalmente por *mismatch* en el canal acústico.

1.2. Hipótesis

En este trabajo de investigación se ha definido la siguiente hipótesis: LNFB es una parametrización robusta a distorsiones de canal, lo que puede mejorar el rendimiento de un sistema de ASR basado en DNN.

1.3. Objetivos

Para evaluar la hipótesis, se ha definido un objetivo general y varios objetivos específicos.

1.3.1. Objetivo General

El objetivo principal de este trabajo es mejorar la robustez de sistemas ASR basado en DNN usando LNFB.

1.3.2. Objetivos Específicos

De acuerdo a lo anterior se fijaron los siguientes objetivos específicos:

- Comparar el rendimiento de un sistema de ASR con LNFB y compararla con parametrización basada en filtros Mel.
- Analizar la robustez del sistema ASR en estudio a ruido aditivo.
- Analizar la robustez del sistema ASR en estudio a distorsión de canal.
- Evaluar la combinación de distintas parametrizaciones, incluyendo LNFB, para el problema de robustez a la reverberación.

1.4. Estructura de la Tesis

En este trabajo se estudia el efecto de la parametrización en un sistema DNN-HMM de reconocimiento de voz, haciendo un enfoque en el uso de una parametrización normalizada localmente llamada LNFB.

En el capítulo 2 se repasan brevemente los conceptos involucrados en el reconocimiento robusto de voz. Se mencionan los tipos de distorsiones existentes, el concepto de redes neuronales, los tipos de parametrizaciones utilizados, los aspectos principales relacionados con un reconocedor de voz.

En el capítulo 3 se introduce la nueva parametrización LNFB. Luego se detallan los experimentos realizados para estudiar el rendimiento de un sistema que usa LNFB. Se estudia la robustez del sistema a distorsiones de tipo ruido aditivo y de canal.

En el capítulo 4 se aborda el tema de complementariedad y combinación de DNNs entrenadas usando parametrizaciones distintas. En particular, se usan las parametrizaciones LNFB, MelFB, PNCC y Rasta-PLP. Además, se utilizan dos maneras de combinar DNNs.

Finalmente, en el capítulo 5 se presentan las conclusiones de este trabajo a la luz de los resultados obtenidos, y se proponen posibles líneas de trabajo futuro.

Capítulo 2

Reconocimiento de voz robusto a canal y ruido

2.1. Sistemas de reconocimiento de voz

El esquema típico de un reconocedor de voz se muestra en la figura 2.1. Como se puede observar, el proceso se ha dividido en dos: extracción de características y decodificador [2], [3].

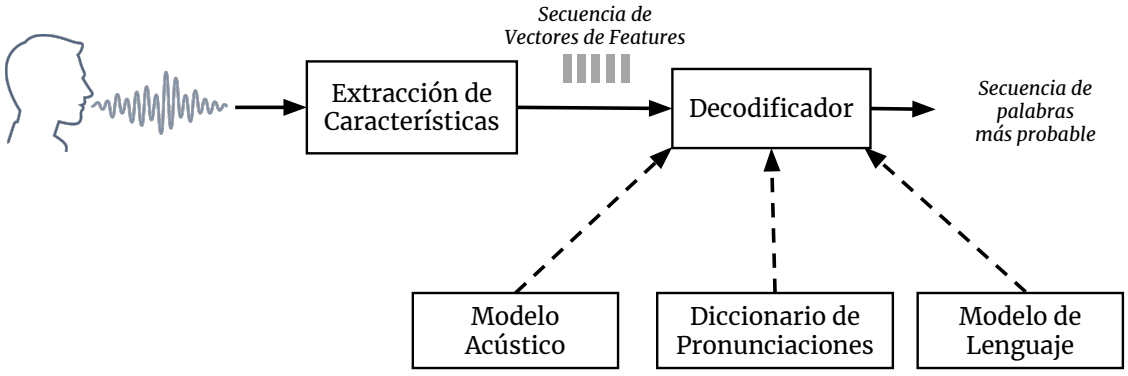


Figura 2.1: Diagrama de un reconocedor de voz.

La señal de voz, ya digitalizada, se convierte en una secuencia de vectores de características de largo fijo $\mathbf{X} = \mathbf{x}_1, ..., \mathbf{x}_T$ a través del proceso de extracción de características o *features* [4]. Estos vectores de características alimentan al decodificador, que intenta encontrar la secuencia de palabras $W = w_1, ..., w_k$ que es más probable que corresponda a $\mathbf{X}$[3]. Es decir, el decodificador entrega una estimación de la secuencia de palabras, definida por la ecuación 2.1:

$$\hat{W} = \arg \max_{W} [P(W \mid \mathbf{X})] \tag{2.1}$$

Debido a que $P(W \mid X)$ es difícil de modelar, se utiliza el teorema de Bayes para reformular la ecuación 2.1, obteniéndose la ecuación 2.2 [2]:

$$\hat{W} = \arg \max_{W} [P(\mathbf{X} \mid W)P(W)] \tag{2.2}$$

El modelo probabilístico que provee la verosimilitud o *likelihood* $P(\mathbf{X} \mid W)$ se denomina "modelo acústico", mientras que el modelo que provee la probabilidad *a priori* $P(W)$ se denomina "modelo de lenguaje" [4]. En un sistema de ASR típico el modelo de lenguaje es un modelo N-grama, en donde la probabilidad de ocurrencia de una palabra sólo depende de las $N-1$ palabras anteriores [2]. Por su parte, el modelo acústico consiste en modelos de fonemas que se concatenan para formar una secuencia dada de palabras W basándose en un diccionario de pronunciaciones, que relaciona la pronunciación de una palabra con su escritura [2].

A fines de los 80 se popularizó el uso de HMMs para implementar el modelo acústico, convirtiéndose en el modelo utilizado en casi todos los sistemas de ASR desde principios de los 90 [5]. La probabilidad de los estados de emisión de estos HMM se modela generalmente con una mezcla de Gaussianas o GMM (*Gaussian Mixture Model*) [6] lo que se denomina modelo GMM-HMM. Recientemente, los modelos GMM han sido reemplazados por una DNN (*Deep Neural Network*), conformando el modelo DNN-HMM [7]. Este modelo ha significado una nueva revolución en el campo de ASR.

A continuación se entra en detalle en los modelos GMM-HMM y DNN-HMM.

2.1.1. GMM-HMM

Como se dijo anteriormente, para cada fonema que existe en el diccionario de pronunciaciones se puede definir y entrenar un modelo HMM. Cuando se define un HMM por fonema, el modelo se denomina "monofono" y contexto-independiente [2]. También se puede tomar en cuenta el contexto, considerando el fonema anterior y el siguiente al fonema que se está modelando, lo que se denomina modelo tri-fonema o tri-fono. El modelo HMM se observa en la figura 2.2.

El HMM se define con tres tipos de elementos[2], [3]:

- **Estados** s_i**.** La secuencia de vectores acústicos $\mathbf{X} = \mathbf{x}_1, ..., \mathbf{x}_T$ puede recorrer el HMM transitando entre estados. Para cada $\mathbf{x}_t$, el HMM estará en un estado s_i determinado. En el siguiente tiempo el vector de entrada será $\mathbf{x}_{t+1}$, y el HMM puede quedarse en el mismo estado s_i o pasar al siguiente. No se permite realizar transiciones a estados anteriores. Para relacionar el estado s_i con la variable temporal t, se define la variable de secuencia temporal $r(t)$ que indica en qué estado está el HMM en el tiempo t.

- **Probabilidades de transiciones entre estados** $\{a_{ij}\}$**.** Representa la probabilidad de transición del estado s_i al estado s_j del HMM: $a_{ij} = P(r(t) = s_i \mid r(t+1) = s_j)$.

- **Distribuciones de probabilidad de emisión** $\{b_j()\}$**.** Estas distribuciones de probabilidad son

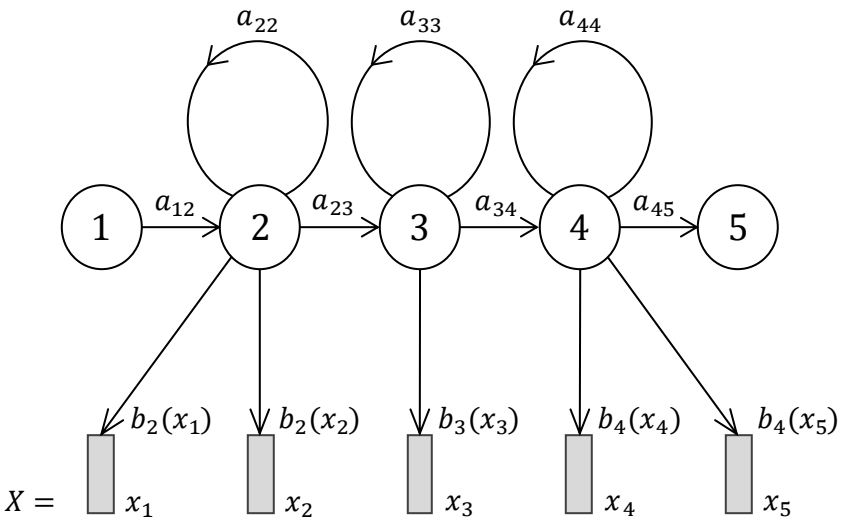


Figura 2.2: Diagrama de un HMM. X representa una secuencia de vectores de características x_i que recorren el HMM en una secuencia temporal particular S (estados 2-2-3-4-4). Para cada x_i se calcula la probabilidad de emisión, que es usada para calcular el *likelihood* de la secuencia según la ecuación 2.6.

las que relacionan los estados del HMM con las variables observables, que en este caso son los vectores acústicos. En otras palabras, $b_j(\mathbf{x_t}) = P(\mathbf{x_t} \mid r(t) = s_j)$.

En el caso de un GMM-HMM, las distribuciones de probabilidad de emisión se modelan con un GMM, como se indica en la ecuación 2.3:

$$b_j(\mathbf{x}) = \sum_{n=1}^{N} \omega_{jn} \cdot \mathcal{N}(\mathbf{x}; \mu_n, \Sigma_n) \tag{2.3}$$

donde N es la cantidad de gaussianas que componen el GMM, $omega_{jn}$ es el peso de combinación n-ésimo, y $\mathcal{N}$ representa la distribución gaussiana multivariante de media μ_{jn} y varianza Σ_{jn}. Nótese que $\mathcal{N}$ se define entonces según la ecuación 2.4:

$$\mathcal{N}(\mathbf{x}; \mu_n, \Sigma_n) = \frac{1}{(2\pi)^{D/2}|\Sigma_i|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x} - \mu_i)^t \Sigma_i^{-1}(\mathbf{x} - \mu_i)\right\} \tag{2.4}$$

donde D es el tamaño del vector de características $\mathbf{x}$. Además, debe cumplirse que:

$$\sum_{n=1}^{N} \omega_{jn} = 1 \tag{2.5}$$

Para evaluar el *likelihood* de que una secuencia de vectores acústicos $\mathbf{X} = \mathbf{x}_1, ..., \mathbf{x}_T$ corresponda

a una secuencia determinada de palabras $W = w_1, ..., w_k$, se deben seguir los siguientes pasos:

- Obtener la secuencia de fonemas $Q = q_1, ..., q_M$ que corresponde a la secuencia de palabras W.
- Concatenar cada uno de los HMM_m correspondientes a cada fonema q_m, resultando en un HMM compuesto total HMM_Q.
- Para cada una de las secuencias temporales $R = r(0), ..., r(T)$ en que se puede recorrer el HMM_Q usando $\mathbf{X}$, calcular el *likelihood* definido como:

$$P(S, \mathbf{X} \mid HMM_Q) = a_{r(0),r(1)} \prod_{t=1}^{T} b_{r(t)}(\mathbf{x}_t) a_{r(t),r(t+1)} \tag{2.6}$$

- Escoger la secuencia con más alta probabilidad.

En el procedimiento anterior existe un problema: se deben explorar todas las posibles secuencias para encontrar la secuencia más probable. Este problema se resuelve usando el algoritmo de Viterbi, que permite encontrar la secuencia más probable de manera eficiente [8].

Los parámetros $\{a_{ij}\}$ y $\{b_j\}$ de los HMM se entrenan usando el algoritmo EM (*Expectation-Maximization*). Para cada audio de entrenamiento, se utiliza un alineamiento entre la secuencia de fonemas (o trifonemas) y los estados de los HMM correspondientes [2].

2.1.2. DNN-HMM

Una de las aplicaciones de *deep learning* en un sistema ASR que ha resultado exitosa es combinar una DNN con HMMs para construir el modelo acústico, lo que se denomina DNN-HMM [9]-[11]. Este sistema es similar a GMM-HMM, con la diferencia que las probabilidades de emisión son modeladas por una DNN en vez de GMMs.

En el sistema DNN-HMM, una sola DNN se entrena para estimar la probabilidad *a posteriori* $P_j(r(t) = s_j \mid \mathbf{x}_t)$ para todos los estados $S = s_1, ..., s_J$. Esto es distinto al modelo GMM-HMM, en donde un GMM distinto se usa para modelar cada estado distinto. Además, la entrada de la DNN no es un único vector de características acústicas $\mathbf{x}_t$, si no una ventana de contexto consistente en $2\nu + 1$ vectores y definida como $\mathbf{O}_t = [\mathbf{x}_{\text{máx}(0,t-\nu)} \cdots \mathbf{x}_t \cdots \mathbf{x}_{\text{mín}(T,t+\nu)}]$[12]. Se dan más detalles sobre las redes neuronales y DNNs en la siguiente sección (2.2).

2.1.3. Evaluación de un sistema ASR

Para evaluar el rendimiento de un sistema de ASR se utiliza una medida de error llamada tasa de error de palabras o WER (*Word Error Rate*). El WER mide el costo de cambiar la secuencia de palabras de salida de un sistema ASR para igualarse a la secuencia de palabras correcta [13].

Para evaluar un sistema ASR, se debe contar con un conjunto de evaluación o test que consiste de varios audios y sus correspondientes transcripciones. Cada audio se ingresa al sistema ASR, obteniéndose como salida una secuencia de palabras llamada "hipótesis". Esta hipótesis se compara con la transcripción o referencia, y se calcula la cantidad de sustituciones, inserciones y eliminaciones de palabras que se deben hacer en la hipótesis para que esta sea igual a la referencia [14].

El WER se define según la ecuación 2.7:

$$WER = \frac{S + D + I}{N_W} \tag{2.7}$$

donde N_W es la cantidad de palabras en la referencia, S es la cantidad de sustituciones, D la cantidad de eliminaciones, e I la cantidad de inserciones.

2.2. Redes Neuronales

Una red neuronal artificial es una máquina inspirada en la manera en que funciona el cerebro, en el sentido en que se definen pequeñas unidades de cómputo llamadas neuronas que se interconectan para realizar el procesamiento deseado [15].

La red neuronal más básica se denomina perceptrón, y fue propuesto por Rosenblatt en 1958 [16]. Básicamente, consiste en una única neurona con pesos y *bias* ajustable, como se observa en la figura 2.3.

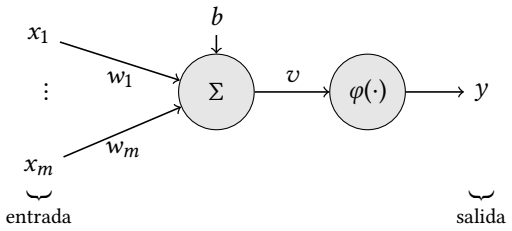


Figura 2.3: Flujo de información a través de un perceptrón.

En la figura 2.3 se tiene un vector de entrada $X = [x_1, ..., x_m]$ de tamaño m, y una salida escalar y. El perceptrón queda definido por los pesos $w_1, ..., w_m$, el *bias* b y la función de activación $\varphi(\cdot)$. La variable v se conoce como pre-activación y está definida por la ecuación 2.8.

$$v = b + \sum_{i=1}^{m} w_i x_i \tag{2.8}$$

Finalmente, se tiene que:

$$y = \varphi(v) = \varphi\left(b + \sum_{i=1}^{m} w_i x_i\right) \tag{2.9}$$

2.2.1. Perceptrón Multicapa

Se pueden organizar múltiples perceptrones en una estructura de red neuronal conocida como perceptrón multicapa o MLP (*Multilayer Perceptron*). Las características básicas del MLP son las siguientes [17]:

- Cada neurona incluye una función de activación no lineal y diferenciable.
- La red contiene una o más capas que están ocultas tanto para los nodos de entrada como los de salida.
- La red tiene un alto grado de conectividad, que está definido por los pesos de cada perceptrón.

El método más utilizado para entrenar los MLP se llama algoritmo de *back-propagation*, y consiste en dos fases [17]:

- **Forward:** En esta etapa los pesos del MLP están fijos y la señal de entrada se propaga por la red, capa a capa, hasta obtener la salida.
- **Backward:** La salida obtenida en la etapa anterior se compara con un *label* o salida deseada, y se mide el error entre ambas con alguna función de error apropiada. Este error se va propagando hacia atrás, desde la salida hacia la entrada. El error propagado en cada capa es usado para ajustar los pesos y *bias* de cada perceptrón.

2.2.2. Autoencoder

Los autoencoders, redes de replicación (*replicator networks*) o mapas de identidad (*identity maps*) son una forma especial de MLP. Tradicionalmente, la arquitectura del autoencoder tiene las siguientes características [17]:

- Las capas de entrada y de salida tienen el mismo tamaño, m.
- El tamaño de la capa oculta, M, es menor que m.
- La red está completamente conectada.

La característica principal del autoencoder es que, en el entrenamiento, una señal de entrada x es aplicada al mismo tiempo como etiqueta o salida deseada. Esto quiere decir que la salida del autoencoder, $\hat{x}$, es un estimado de la entrada, x [17].

Debido a su arquitectura, los autoencoder son capaces de tomar la información contenida en x que es de tamaño m, y comprimirla en la capa oculta de tamaño M. La información de la capa oculta es suficiente para poder reconstruir la señal con un cierto nivel de error aceptable ($\hat{x}$). Los autoencoder se utilizaban principalmente para reducción de dimensionalidad o codificación, usando la salida de la capa oculta en vez de la salida de la red [18].

Existe un tipo de autoencoder que puede ser utilizado para remover ruido de la señal de entrada. Este autoencoder se conoce como DAE (*Denoising Autoencoder*) [18]. El DAE se entrena tomando como entrada una señal con ruido, $\tilde{x}$, tomando como etiqueta la versión limpia de la entrada, x. Así, el DAE aprende a limpiar la entrada en vez de copiarla.

2.2.3. Deep Neural Network

Una DNN o *Deep Neural Network* es simplemente un MLP con varias capas ocultas, como se muestra en la figura 2.4. También se le conoce como *Deep Feedforward Network*, puesto que *feedforward* hace referencia a que la información se propaga desde la entrada por cada una de las capas ocultas de la red hasta llegar a la salida. Es decir, no existen conexiones que hagan retroalimentación (*feedback*) a capas anteriores. Cuando en un MLP se definen conexiones que realizan *feedback* se denominan Redes Neuronales Recurrentes [19].

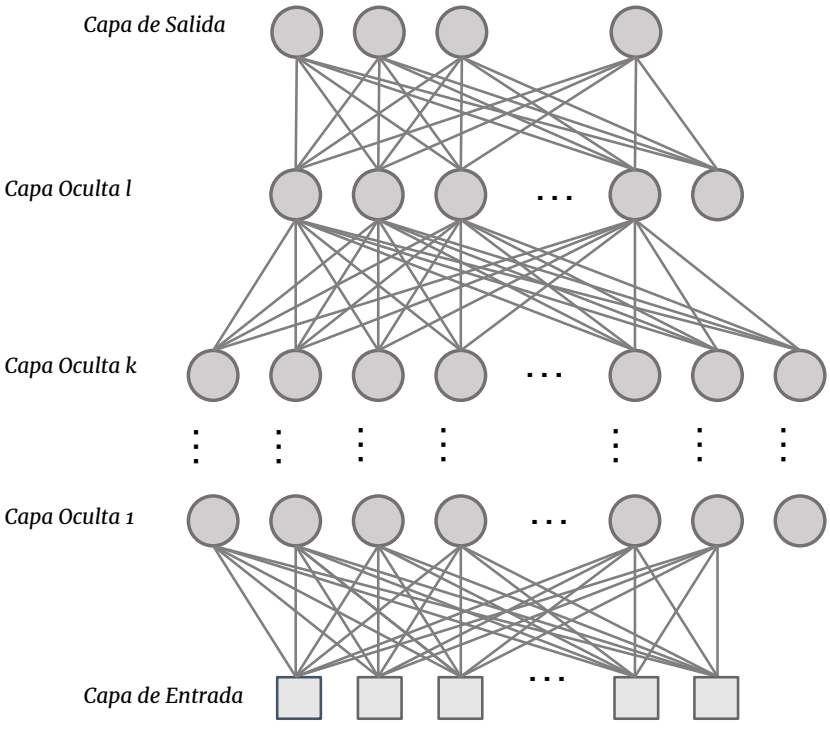


Figura 2.4: Representación de una DNN de $l + 1$ capas. Los cuadrados representan los nodos de entrada, mientras que los círculos representan los nodos de cómputo o perceptrones.

Las DNN son los modelos de *deep learning* por excelencia. De hecho, el concepto de *deep learning* viene del uso de varias capas ocultas (la profundidad o *depth*) en una DNN [19]. Cuando

se aumenta la cantidad de capas de un MLP se dificulta el entrenamiento debido a los fenómenos de *vanishing gradient* y *exploding gradient* , que se puede traducir como "gradiente que se desvanece" y que "explota". Este problema ocurre al aplicar el algoritmo de *backpropagation*. El error de entrenamiento se va propagando desde la salida hacia las capas anteriores, usándose para calcular un gradiente de cambio para ajustar los pesos en cada nodo de la red. Cuando se va llegando a las primeras capas, el gradiente puede o hacerse muy pequeño, o demasiado grande, o inestable, impidiendo un correcto ajuste de los parámetros de la red [20].

Sólo en los últimos años, los avances realizados en *hardware* y en algoritmos de *machine learning* permitieron entrenar una DNN que contiene muchas capas ocultas y con una capa de salida grande [7]. En particular, se descubrió que una adecuada inicialización de los parámetros de la red, junto al algoritmo de Gradiente Descendente Estocástico, permitía entrenar la DNN para alcanzar un bajo nivel de error [21]. En el caso de los sistemas ASR, una buena inicialización de los parámetros de la red se consigue con el uso de RBMs (*Restricted Boltzmann Machines*) apiladas para conformar una *Deep Belief Network* [9].

2.3. El problema de la robustez de ASR a canal y ruido

Cuando un sistema de ASR se prueba en un ambiente real, el audio de entrada al sistema se verá afectado por las condiciones acústicas del ambiente; por lo tanto, los sistemas ASR deben ser robustos a variaciones en las condiciones acústicas. Si un sistema ASR no tiene en cuenta esta variabilidad, su rendimiento disminuirá significativamente debido a las diferencias o *mismatch* entre las condiciones de entrenamiento y prueba del sistema **Junqua2002**, [22], [23].

Existen muchas formas de distorsiones acústicas que pueden afectar las señales de entrada de un modelo ASR. Estas se pueden clasificar principalmente en dos categorías: ruido aditivo y distorsión convolucional [22]. A continuación, se describe someramente cada una de las distorsiones mencionadas.

2.3.1. Ruido Aditivo

El ruido aditivo es un tipo de ruido muy común. Los ruidos aditivos pueden ser a su vez clasificados en dos tipos: estacionario y no estacionario [23]. El ruido estacionario tiene un espectro constante en el tiempo, como por ejemplo el sonido del aire acondicionado o el del ventilador de un computador. Al contrario, el ruido no estacionario tiene un espectro que varía en el tiempo. En la práctica, ningún ruido es perfectamente estacionario, pero se pueden considerar estacionarios dentro de una cierta ventana de tiempo.

Cuando la señal de audio va a ser grabada por un micrófono, la señal limpia de voz se graba junto con los ruidos existentes en el ambiente. Luego, si se denota como $y(t)$ la señal capturada por el micrófono, se tiene que:

$$y(t) = x(t) + n(t) \tag{2.10}$$

donde $x(t)$ corresponde a la señal limpia de voz que hubiera sido capturada por el micrófono si es que no hubiera existido ruido, y $n(t)$ corresponde al ruido en el ambiente capturado por el micrófono [23].

2.3.2. Distorsión Convolucional

La distorsión convolucional se produce cuando una señal de voz es filtrada por un sistema lineal. Este sistema puede representar la respuesta de la sala en la que se está grabando la señal, del micrófono, del canal de comunicación, etc. Luego, la señal vista por el sistema ASR será la señal original convolucionada con dicho sistema [22]. Luego, si la señal limpia de voz $x(t)$ pasa por un sistema lineal $h(t)$, la señal capturada viene dada por la ecuación 2.11:

$$y(t) = x(t) * h(t) \tag{2.11}$$

Al realizar un análisis espectral se obtiene que:

$$Y(f) = X(f) \cdot |H(f)|^2 \tag{2.12}$$

donde $Y(f)$, $X(f)$ y $H(f)$ representan las densidades espectrales de potencia de $y(t)$, $x(t)$ y $h(t)$ respectivamente.

2.3.3. Entrenamiento *clean* vs entrenamiento *multicondition*

El entrenamiento *clean* se refiere a que tanto los audios de entrenamiento como los de test no presentan ningún tipo de ruido. Con este tipo de entrenamiento y test se alcanzan los WER más bajos; sin embargo, en un escenario real es difícil replicar las condiciones *clean*. Cuando un sistema ASR entrenado en condición *clean* intenta decodificar audios con algún tipo de variabilidad o ruido, el rendimiento del sistema cae dramáticamente [24].

Las técnicas de mejoramiento de características o *feature enhancement* intentan eliminar el ruido y la variabilidad acústica en los audios para lograr igualar una condición *clean*. Por otra parte, existe un enfoque completamente distinto: en vez de intentar alcanzar la condición *clean* en entrenamiento y test, se trata de igualar las condiciones de ruido y variabilidad en entrenamiento y test, lo que se conoce como entrenamiento *multi-style* o *multicondition* [25].

Tanto los sistemas GMM-HMM como DNN-HMM se benefician del entrenamiento *multicondition*, pero hay una diferencia entre ambos. En el sistema GMM-HMM, los vectores acústicos se modelan directamente con un GMM, lo que implica que se termina modelando la variabilidad adicional introducida por el ruido presente en el entrenamiento. En el caso de la DNN, esta aprende una representación robusta de los vectores acústicos en sus capas más profundas que es invariante a la variabilidad presente en las diversas condiciones acústicas presentes en el

entrenamiento *multicondition* [24].

2.4. Parametrización

La parametrización de las señales de voz es el primer paso para entrenar cualquier sistema computacional basado en procesamiento de voz. Consiste en transformar las señales temporales captadas por un micrófono en vectores de características que serán las entradas al modelo [26], [27].

En esta sección se abordaran cuatro etapas de la parametrización de voz: obtención de *frames*, extracción de *features* o características, cálculo de coeficientes dinámicos y normalización.

2.4.1. Obtención de *frames*

Este procedimiento es común a otras tareas de procesamiento de voz. Consiste en dividir la señal de voz en tramas o *frames* de entre 20 y 30 [ms] de duración. Luego de aplicar un filtro pasa altos (pre-énfasis), la señal de voz es dividida en *frames* que se traslapan y además se multiplican por una función ventana para evitar discontinuidades que aparezcan como componentes de alta frecuencia en el espectro del *frame* [26].

2.4.2. Extracción de *features*

A cada *frame* obtenido en la etapa anterior se aplica una extracción de *features*. El objetivo de esta extracción es obtener una representación del *frame* más robusta y adecuada para el entrenamiento de un reconocedor de voz.

A continuación se presentan 5 tipos de *features* que se pueden calcular sobre el mismo *frame*: MFCC, banco de filtros Mel, RASTA-PLP, PNCC y LNCC. Cada uno de estos *features* corresponde a diferentes representaciones de la misma información, que pueden ser más o menos efectivas dependiendo de las condiciones acústicas en que fue grabada la señal, y del sistema de reconocimiento que se desee implementar.

2.4.2.1. MFCC

Los coeficientes cepstrales en frecuencia Mel o MFCC (*Mel-Frequency spaced Cepstral Coefficients*) es una manera de caracterizar una señal de voz basada en la percepción auditiva humana. Este método ha sido ampliamente utilizado las últimas dos décadas para reconocimiento de voz [26].

Para obtener los MFCC se realizan los siguientes pasos:

1. **STFT:** Se calcula la transformada de Fourier de tiempo corto o *Short-Term Fourier Transform* (STFT). Esta consiste en aplicar una función ventana que se va moviendo sobre la señal,

obteniendo varias tramas o *frames*. A cada uno de estos *frames* se le aplica la transformada de Fourier rápida o FFT (por *Fast Fourier Transform*) para calcular el espectro de potencia.

2. **Banco de Filtros Mel:** Se aplica un banco de filtros triangular sobre el espectro de potencia. Los centros de los filtros triangulares se separan según la escala Mel definida en la ecuación 2.13.

$$f_{MEL} = \log_{10}\left(1 + \frac{f_{LIN}}{700}\right)] \tag{2.13}$$

   El número de filtros depende de la implementación, pero debe ser menor a la cantidad de elementos de la FFT. A las energías resultantes de la aplicación del banco de filtros se les calcula el logaritmo; estas se denominan log-energías.

3. **DCT:** Finalmente, al vector de log-energías se le aplica la transformada de coseno discreta o DCT (*Discrete Cosine Transform*).

Luego, el coeficiente MFCC n-ésimo se define en la ecuación 2.14:

$$MFCC_n = \sum_{m=1}^{M} \log(S_m) \cos\left[\frac{\pi n}{M}\left(m - \frac{1}{2}\right)\right] \tag{2.14}$$

donde M es la cantidad de filtros del banco de filtros Mel y S_m es la energía del filtro m-ésimo.

2.4.2.2. MelFB: Banco de filtros Mel

Los *features* MelFB también conocidos como *features* Fbank o FBANK se calculan de la misma manera que los *features* MFCC, pero hasta antes de aplicar la DCT. Hasta antes de la aplicación de *Deep Learning* en reconocimiento de voz, los sistemas *state-of-the-Art* para ASR usaban MFCC y no MelFB debido a que estos últimos están fuertemente correlacionados entre sí [28]. Sin embargo, al utilizar DNN y otras arquitecturas de *Deep Learning* en los sistemas de reconocimiento, se encontró que se obtenía mejor rendimiento al usar representaciones más simples como MelFB [24], [28].

2.4.2.3. Rasta-PLP

Los coeficientes PLP [29] se calculan usando la técnica de predicción lineal perceptual (*Perceptual Linear Predictive*). En esta técnica, basada en análisis lineal predictivo, el espectro auditivo es aproximado por un modelo autorregresivo todo-polo de orden 5. Para obtener los PLP, se realiza el siguiente procedimiento:

1. **STFT:** Al igual que MFCC, estos coeficientes se calculan sobre el espectro de potencia $P(\omega)$ la STFT, donde ω es la frecuencia angular.

2. **Modelamiento Auditivo:** En esta etapa se realiza una serie de análisis y transformaciones que toman en cuenta la forma en que el sistema auditivo humano funciona.

En primer lugar se hace una transformación del espectro de potencia a escala Bark. Esta transformación se define con la siguiente ecuación:

$$\Omega(\omega) = 6 \log \left\{ \frac{\omega}{1200\pi} + \left[\frac{\omega}{1200\pi} \right]^{0,5} \right\} \tag{2.15}$$

donde ω es la frecuencia angular en radianes por segundo. Luego de la transformación de escala se aplica un banco de filtros llamado máscara de banda crítica $\Psi(\Omega)$, que tiene la forma de los filtros del sistema auditivo (aproximadamente constantes en escala Bark)[29].

El siguiente paso es aplicar pre-énfasis a la energía de los filtros, $\Theta[\Omega(\omega)]$, utilizando una curva de igual sonoridad (*equal-loudness*) $E(\omega)$. Esta curva representa los distintos niveles de sonoridad con que el oído humano percibe distintas frecuencias. El resultado del pre-énfasis es $\Xi[\Omega(\omega)] = E(\omega)\Theta[\Omega(\omega)]$.

El último paso consiste en comprimir el resultado anterior para simular la ley de potencial de Stevens, que toma en cuenta la relación entre el estímulo físico y la intensidad percibida: $\Phi(\Omega) = \Xi(\Omega)^{0,33}$.

3. **Modelamiento Autorregresivo:** Finalmente, $\Phi(\Omega)$ se aproxima por el espectro de un modelo todo-polos usando el método de autocorrelación. Se aplica la DFT inversa sobre $\Phi(\Omega)$ para calcular la autocorrelación. Los primeros $M + 1$ coeficientes se utilizan para resolver las ecuaciones de Yule-Walker que entregan los coeficientes autorregresivos del modelo todo-polos de orden M. Estos coeficientes autorregresivos corresponden a los *features* PLP.

En la construcción de los *features* PLP se utilizan varias transformaciones que toman en cuenta la forma en que los humanos perciben el sonido, sin embargo no es robusto a cambios en la STFT debido a la respuesta en frecuencia del canal de comunicación [30]. Debido a esto, se desarrolló la técnica RASTA (*RelAtive SpecTrAl*) [31] para dar robustez ante distorsiones espectrales lineales.

La técnica RASTA aplica un filtrado pasa banda en cada banda de frecuencia definida en el espectro, usando un filtro IIR (de respuesta infinita al impulso) con la siguiente función de transferencia:

$$H(z) = 0,1 \cdot \frac{2 + z^{-1} - z^{-3} - 2z^{-4}}{z^{-4} \cdot (1 - 0,98z^{-1})} \tag{2.16}$$

Para construir los *features* RASTA-PLP, se realiza el siguiente procedimiento [30]:

- Se obtiene el espectro de banda crítica y se aplica el logaritmo.
- Se obtiene la derivada temporal del espectro usando regresión lineal sobre 5 valores consecutivos del espectro.
- Se aplica el filtrado RASTA.

- Se agrega la curva de igual sonoridad y la ley de potencial de Stevens (tal como en el análisis PLP).
- Se toma el logaritmo inverso.
- Se calcula el modelo todo-polos y se estiman los coeficientes autorregresivos, que en este caso serían coeficientes RASTA-PLP.

2.4.2.4. PNCC

Los coeficientes cepstrales normalizados en potencia, o PNCC (Power Normalized Cepstral Coefficient) son otro tipo de coeficientes cepstrales robustos a ruido, que cuenta con varios tipos de implementación [32], [33]. La normalización se logra al sustraer un cierto nivel de potencia en cada coeficiente, procedimiento llamado *Power-Bias Subtraction* (PBS). El nivel se elige de manera de mejorar la forma de la distribución de potencia.

Para obtener los PNCC se realiza un procedimiento parecido al de MFCC:

1. **STFT:** Se calcula la STFT al igual que con MFCC.
2. **Banco de Filtros Gammatone:** En vez de aplicar un banco de filtros Mel como en MFCC, se aplica un banco de canales gammatone usando la siguiente expresión:

$$P_{org}(m,l) = \int_0^{\pi} |X(m;e^{j\omega})H_l(e^{j\omega})|^2 d\omega \tag{2.17}$$

   donde $P_{org}(m,l)$ es la potencia del *frame* m en el canal gammatone l, $H_l(e^{j\omega})$ es la respuesta en frecuencia del canal l, y $X(m;e^{j\omega})$ es el espectro de tiempo corto del *frame* m.

3. **Normalización de potencia:** La potencia se normaliza usando la potencia *peak* definida como el percentil 95 dentro de la potencia de tiempo corto:

$$P(m,l) = p_0 \frac{P_{org}(m,l)}{P_{peak}} \tag{2.18}$$

4. **Media Móvil:** Se aplica una media móvil a la potencia obtenida en el paso anterior, antes de aplicar PBS:

$$Q(m,l) = \frac{1}{2M+1} \sum_{l'=l-M}^{l+M} P(m,l') \tag{2.19}$$

5. **PBS:** La sustracción de potencia se obtiene con la siguiente ecuación:

$$\tilde{P}(m,l) = \frac{1}{l_2 - l_1 + 1} \sum_{l'=l_1}^{l_2} \frac{\tilde{Q}(m,l')}{Q(m,l')} P(m,l') \tag{2.20}$$

6. **DCT:** Finalmente, se aplica la DCT tal como en la construcción de los *features* MFCC.

2.4.2.5. LNCC

Los coeficientes cepstrales localmente normalizados, o LNCC, son un set de *features* robustos a ruido de canal [1]. El desarrollo de estos *features* fue inspirado por el detector síncrono generalizado (GSD) de Seneff [34], que realiza una normalización local en el dominio de la frecuencia en cada canal auditivo. Por lo tanto, los LNCC son relativamente invariantes a los cambios en la respuesta de frecuencia del canal de transmisión [1].

La normalización local se logra en el espacio del banco de filtros. Se definen dos bancos de filtros: el primero es un banco de filtros triangular, similar al usado en los coeficientes MFCC convencionales; el segundo es un banco de filtros triangulares inversos. Ambos bancos tienen la misma cantidad de filtros y se corresponden las frecuencias centrales y el ancho de banda en cada canal. En cada canal m, los filtros correspondientes a cada banco de filtros se denominan "filtro numerador" y "filtro denominador". Estos filtros se definen según las ecuaciones 2.21 y 2.22 respectivamente, donde la frecuencia f está en escala Bark [35].

$$Num_m(f) = \begin{cases} -\frac{2}{B}|f - {f_m}^C|, & \text{si} \quad |f - {f_m}^C| \leq \frac{B}{2} \\ 0, & \text{si no} \end{cases} \tag{2.21}$$

$$Den_m(f) = \begin{cases} \frac{2}{B}(1 - d_{min})|f - {f_m}^C| + d_{min}, & \text{si} \quad |f - {f_m}^C| \leq \frac{B}{2} \\ 0, & \text{si no} \end{cases} \tag{2.22}$$

La forma de estos filtros en el canal m (ver figura 2.5) es una aproximación a la respuesta de frecuencia del numerador y el denominador del operador GSD de Seneff [34].

La normalización se logra al dividir la salida del filtro triangular por la salida del filtro inverso, en cada canal [1]. Dado un canal m con frecuencia central ${f_m}^C$ y ancho de banda B, la energía localmente normalizada del canal m es LN_m, definida en la ecuación 2.23:

$$LN_m = \frac{E\_Num_m}{E\_Den_m} \tag{2.23}$$

donde E_Num_{mm} es la energía del filtro numerador Num_m (ecuación 2.21), y E_Den_{mm} es la energía del filtro denominador Den_m (ecuación 2.22).

Finalmente se aplica un procedimiento similar al realizado en el cálculo de *features* MFCC. Se aplica primero el logaritmo a las energías normalizadas LN_m, y luego se les aplica la transformada DCT definida en la ecuación 2.14.

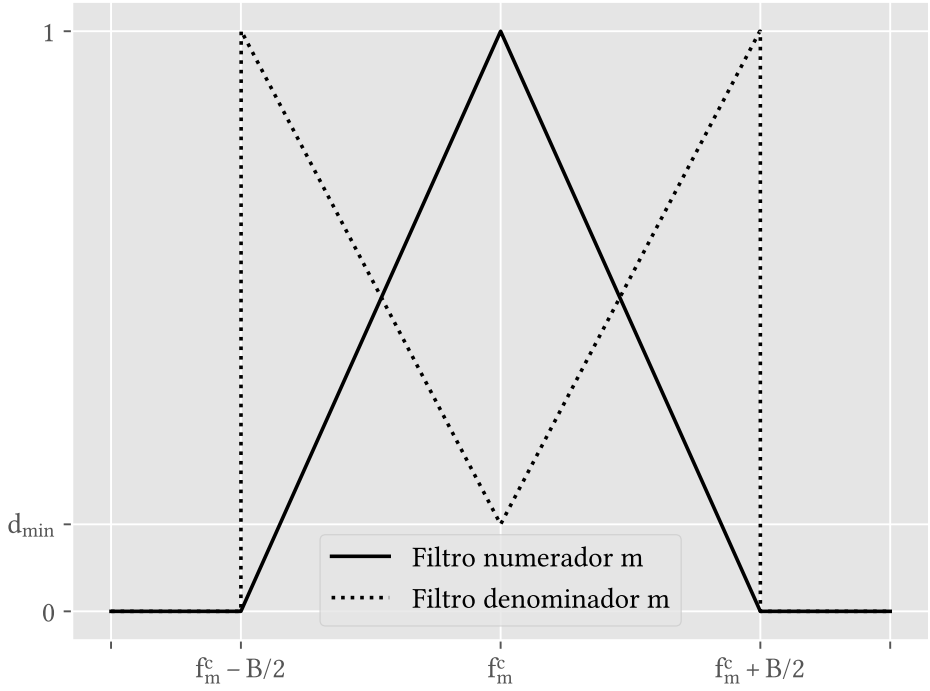


Figura 2.5: Representación gráfica del m-ésimo filtro numerador (en línea sólida) y el m-ésimo filtro denominador (linea punteada). El eje de frecuencia (eje x) está en escala Bark [35] tal como se ha usado en otros trabajos [1], [36].

2.4.3. Coeficientes dinámicos

Los coeficientes dinámicos, también conocidos como coeficientes Δ y $\Delta\Delta$ (delta y delta-delta), son aproximaciones polinomiales de la derivadas temporales de primer y segundo orden calculadas sobre *features* estáticos [37]. En las ecuaciones 2.24 y 2.25 se muestran estos coeficientes calculados sobre el *feature* c n-ésimo, siendo típicamente $K = 2$.

$$\Delta c_n = \frac{\sum_{k=1}^{K} k \cdot (c_{n+k} - c_{n-k})}{2 \sum_{k=1}^{K} k^2} \tag{2.24}$$

$$\Delta\Delta c_n = \frac{\sum_{k=1}^{K} k \cdot (\Delta c_{n+k} - \Delta c_{n-k})}{2 \sum_{k=1}^{K} k^2} \tag{2.25}$$

Estos coeficientes se calculan generalmente sobre *features* basados en el espectro o *cepstrum* de tiempo corto (*short-term*), como los MFCC, Mel o PLP, que proveen una buena estimación del espectro local [27]. Luego, los coeficientes delta y delta-delta entregan información de cómo evoluciona el espectro de la señal a lo largo del tiempo.

2.4.4. Normalización

La normalización de los *features* acústicos es una técnica ampliamente utilizada que puede reducir la variabilidad entre las señales de entrenamiento y test [23]. Se definen a continuación dos técnicas que son ampliamente utilizadas en ASR.

2.4.4.1. *Cepstral Mean Normalization*

La normalización más simple de implementar es la normalización de media, conocida como *Cepstral Mean Normalization*[23], que es referida en este trabajo como MN (Normalización de Media). Si se cuenta con una señal de voz ya parametrizada en un set de vectores de coeficientes $\{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_T\}$, la normalización MN consiste en sustraer el vector de medias $\mu_{\mathbf{c}}$ de cada vector $\mathbf{c}_t$, obteniéndose el vector normalizado $\hat{\mathbf{c}}_t$, como se muestra en las siguientes ecuaciones [23]:

$$\mu_{\mathbf{c}} = \frac{1}{T}\sum_{t=1}^{T}\mathbf{c}_t \tag{2.26}$$

$$\hat{\mathbf{c}}_t = \mathbf{c}_t - \mu_{\mathbf{c}} \tag{2.27}$$

MN permite compensar distorsiones causadas por micrófonos con diferentes funciones de transferencia, distancia variable entre el usuario y el micrófono, la acústica de la sala o los canales de transmisión[23].

2.4.4.2. *Cepstral Mean and Variance Normalization*

La normalización *Cepstral Mean and Variance Normalization*, en adelante MVN (Normalización de Media y Varianza), continúa con la idea de MN pero agregando la varianza en la normalización, como se muestra en las siguientes ecuaciones[23]:

$$\sigma_{\mathbf{c}}{}^2 = \frac{1}{T}\sum_{t=1}^{T}\mathbf{c}_t{}^2 - \mu_{\mathbf{c}}{}^2 \tag{2.28}$$

$$\hat{\mathbf{c}}_t = \frac{\mathbf{c}_t - \mu_{\mathbf{c}}}{\sigma_{\mathbf{c}}{}^2} \tag{2.29}$$

A diferencia de MN, MVN no se asocia con algún tipo particular de ruido. Sin embargo, se ha probado empíricamente que provee robustez ante *mismatch* de canal y ruido aditivo[23].

Capítulo 3

Rendimiento de un sistema DNN-HMM usando LNFB

En esta sección se propone evaluar el rendimiento de reconocimiento de voz usando la parametrización basada en la normalización local de filtros. En vez de usarlos después de la transformada cepstral se va a usar directamente el logaritmo de la razón entre los filtros definidos en LNCC. Esto para probar una parametrización más simple de modo a darle espacio a la red neuronal para que encuentre una representación más óptima.

3.1. LNFB: Banco de filtros localmente normalizado

3.1.1. Definición

En la sección 2.4.2 se describió un set de *features* acústicos de tipo cepstral llamados LNCC[1]. Estos *features* fueron probados en la tarea de verificación de locutor, llevando a mejores resultados que MFCC ante *tilt* espectral, ruido aditivo y reverberación [1], [36], [38].

Se ha mostrado que en un sistema DNN-HMM los *features* más simples llevan a mejores resultados [24], por lo que generalmente se prefieren usar los *features* MelFB en vez de MFCC, dado que MelFB es similar a MFCC pero sin aplicar la transformación DCT. Del mismo modo, se propone un tipo de característica llamado "Banco de filtros localmente normalizado" o LNFB por sus siglas en inglés (*Locally Normalized Filter Bank*), que son similares a los LNCC, pero sin aplicar el cálculo de la DCT. Por lo tanto, siguiendo la nomenclatura de la ecuación 2.23, el componente m de LNFB se define como el logaritmo de la energía localmente normalizada del canal m, según la ecuación 3.1.

$$LNFB_m = log(LN_m) = log(\frac{E\_Num_m}{E\_Den_m}) \tag{3.1}$$

3.1.2. Coeficientes dinámicos para LNFB

Los *features* o coeficientes dinámicos delta y delta–delta [39] representan el método más común para capturar la evolución temporal del *cepstrum* y espectro de frecuencia calculados en corto plazo (*short-term spectrum*) [27]. Estos coeficientes dinámicos se calculan como aproximaciones polinomiales de primer y segundo orden de las derivadas temporales de *features* estáticos. Si $C_m(n)$ es el *feature* estático m en el *frame* n, el coeficiente delta, $\Delta C_m(n)$, se define según la ecuación 3.2:

$$\Delta C_m(n) = \frac{\sum_{k=-K}^{K} k C_m(n+k)}{\sum_{k=-K}^{K} k^2} \tag{3.2}$$

En la ecuación 3.2 se utilizan $2K+1$ *frames* centrados alrededor del *frame* n para calcular la derivada temporal del frame n. La variable K permite controlar la suavidad de la derivada calculada. En este trabajo se utiliza $K = 2$. De manera similar se define la segunda derivada temporal, o delta-delta, como:

$$\Delta\Delta C_m(n) = \frac{\sum_{k=-K}^{K} k \Delta C_m(n+k)}{\sum_{k=-K}^{K} k^2} \tag{3.3}$$

Los coeficientes delta y delta–delta representan las características dinámicas del espectro de la voz, y son utilizadas comúnmente en combinación con coeficientes estáticos como MFCC, LPC, PLP, MelFB, etc. La aplicación directa de las ecuaciones 3.2 y 3.3 para calcular los coeficientes delta sobre los *features* LNFB producen los siguientes coeficientes:

$$\begin{aligned}
\Delta LNFB1_m(n) &= \frac{\sum_{k=-K}^{K} k LNFB_m(n+k)}{\sum_{k=-K}^{K} k^2} \\
&= \frac{\sum_{k=-K}^{K} k \log(\frac{E_Num_m(n+k)}{E_Den_m(n+k)})}{\sum_{k=-K}^{K} k^2} \\
&= \frac{\sum_{k=-K}^{K} k\{\log(E_Num_m(n+k)) - \log(E_Den_m(n+k))\}}{\sum_{k=-K}^{K} k^2} \\
&= \frac{\sum_{k=-K}^{K} k \log(E_Num_m(n+k))}{\sum_{k=-K}^{K} k^2} - \frac{\sum_{k=-K}^{K} k \log(E_Den_m(n+k))}{\sum_{k=-K}^{K} k^2} \\
&= \Delta \log(E_Num_m) - \Delta \log(E_Den_m)
\end{aligned} \tag{3.4}$$

De manera análoga se obtiene la expresión de los coeficientes delta-delta:

$$\Delta\Delta LNFB1_m(n) = \Delta\Delta \log(E\_Num_m) - \Delta\Delta \log(E\_Den_m) \tag{3.5}$$

En las ecuaciones 3.4 y 3.5 se observa que, debido a la propiedad del logaritmo del cociente, las derivadas temporales de LNFB son en realidad la diferencia entre las derivadas temporales de las log-energías de los filtros numerador y denominador.

Debido a que los *features* LNFB ya han sido normalizados, se cree que calcular los coeficientes delta y delta-delta usando las ecuaciones 3.2 y 3.3 no representarán de manera adecuada la evolución temporal del espectro. Como se mencionó, las derivadas temporales se terminan aplicando en forma independiente en las energías de los filtros numerador y denominador, y no es claro que significado tiene esta operación. En conclusión, usar esta definición no es consistente con la motivación original de los coeficientes dinámicos [27], [39].

En lugar de la operación anterior, se propone calcular los coeficientes dinámicos de los *features* LNFB aplicando la regresión linear de 3.2 y 3.3 sólo a la energía de los filtros numeradores, usando las ecuaciones 3.6 y 3.7:

$$\Delta LNFB2_m(n) = \Delta \log(E\_Num_m) \tag{3.6}$$

$$\Delta\Delta LNFB2_m(n) = \Delta\Delta \log(E\_Num_m) \tag{3.7}$$

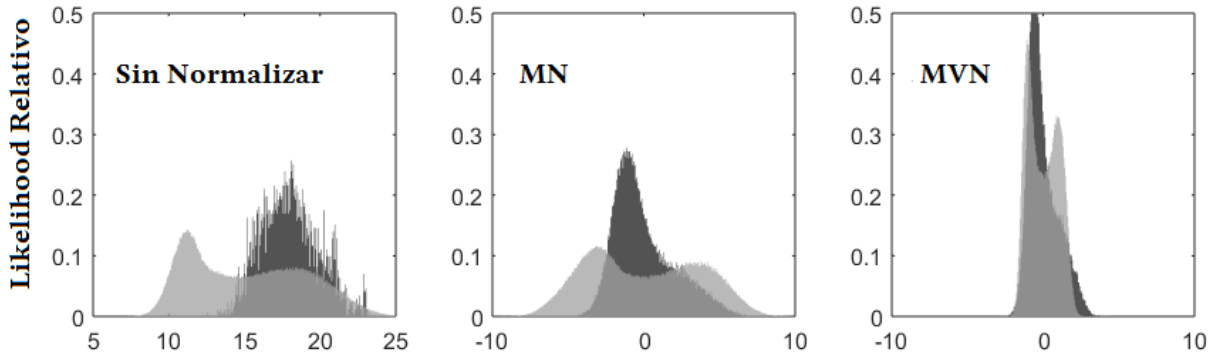
Esta estrategia usada para calcular los coeficientes dinámicos sobre LNFB puede ser fácilmente generalizada a otros *features* acústicos normalizados. La superioridad de los coeficientes dinámicos así calculados es confirmada por los resultados experimentales que se describen más adelante.

3.1.3. LNFB y Normalización

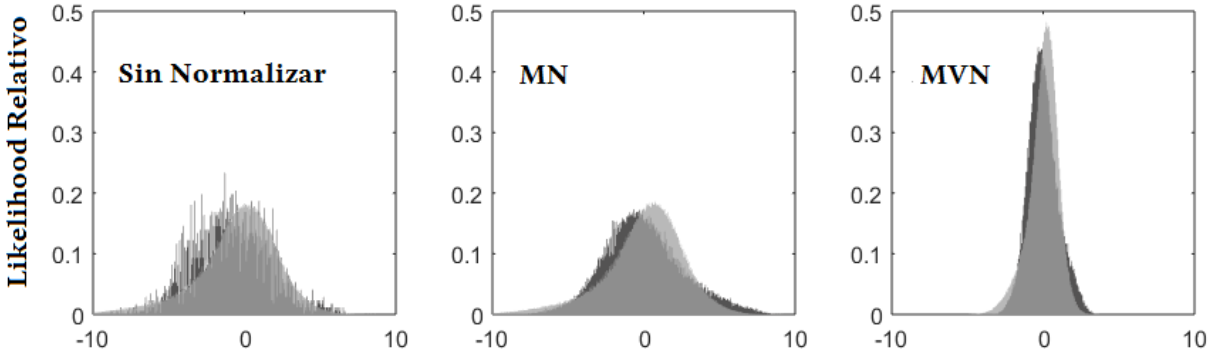
La normalización de los *features* acústicos usando MN (Normalización de Media) y MVN (Normalización de Media y Varianza) es una técnica ampliamente utilizada para realizar reconocimiento robusto de voz [23]. Al aplicar estas técnicas en un sistema basado en DNN-HMM, las medias y/o varianzas se pueden evaluar de distintas maneras. En la literatura se encuentran casos donde estos parámetros se calculan sobre todos los datos de entrenamiento o conjuntamente sobre datos de entrenamiento y test. También se pueden calcular parámetros diferenciando por hablante (*speaker*) o bien por elocución (*utterance*) (por ejemplo [24], [40]-[42]).

Como se mencionó anteriormente, la normalización tiene un efecto combinado en un sistema DNN-HMM: compensación acústica y normalización de escala en la entrada de la DNN. Debido a esto, la normalización óptima puede depender de la tarea que se está ejecutando y del grado de discordancia entre las condiciones acústicas de entrenamiento y test.

En este trabajo, cuatro esquemas de normalización de datos son considerados: MN y MVN aplicados por *speaker* y por *utterance*. En todos los casos se normalizaron los datos de entrenamiento y de test.



(a) Histogramas para coeficientes MelFB



(b) Histogramas para coeficientes LNFB

Figura 3.1: Histogramas para filtros con frecuencia central de aproximadamente 470[Hz], para el set de entrenamiento *clean* de Aurora-4 (en gris claro) y para el set de test grupo D de Aurora-4 (en gris oscuro), y para los *features* a) MelFB y b) LNFB. Las condiciones de normalización aplicadas fueron: sin normalizar (columna izquierda), MN (columna central) y MVN (columna derecha). Estas normalizaciones fueron aplicadas por *speaker* en los datos de entrenamiento y test.

En la figura 3.1 se comparan las amplitudes de las distribuciones de MelFB y LNFB sin normalizar, con MN y con MVN. Las distribuciones se obtuvieron al normalizar los histogramas de los *features* cuyos filtros estaban centrados en los 470 [Hz]. Además, estas distribuciones se calcularon sobre datos de entrenamiento y test por separado, con datos provenientes del corpus Aurora-4 descrito en 3.2.1. Específicamente, se usó la base de datos de entrenamiento *clean* y el grupo D de test, el cual contiene distorsión de canal y seis tipos de ruido aditivo. Finalmente, en todos los datos se aplicó normalización MN y MVN por cada *utterance*.

Para simplificar el análisis, los histogramas construidos con datos de entrenamiento *clean* serán referidos como “histogramas *clean*”, mientras que los histogramas construidos con los datos del grupo D de test serán referidos como “histogramas ruidosos”.

Como se puede ver en la figura 3.1, los histogramas *clean* y ruidoso de los coeficientes MelFB son bimodales o multimodales, mientras que los de LNFB son unimodales. Después de aplicar normalización MN o MVN, los histogramas *clean* y ruidoso continúan siendo claramente distinguibles entre sí cuando se usan los *features* MelFB, a pesar de que han sido escalados a rangos similares de variación. En cambio, los histogramas *clean* y ruidoso obtenidos con los *features* LNFB

son unimodales y muy similares después de aplicar MN o MVN.

Aunque no se muestra en este trabajo, este comportamiento se observó para la gran mayoría de los canales de frecuencia, es decir, para cada *feature*. Este resultado sugiere que LNFB debería ser más robusto que MelFB a perturbaciones como ruido aditivo y distorsión de canal; además, también sugiere que las normalizaciones MN y MVN deberían ser más efectivas para los coeficientes LNFB. Se piensa que la naturaleza unimodal de los histogramas obtenidos con LNFB es una consecuencia de la normalización de la energía del filtro numerador usando la energía del filtro denominador definida en la ecuación 3.1.

3.2. Experimentos

3.2.1. Base de datos

Los experimentos de ASR se realizaron sobre la base de datos de vocabulario medio Aurora-4 [43], [44]. Esta base de datos es un corpus diseñado por la Aurora Working Group del European Telecommunications Standards Institute (ETSI). A su vez, Aurora-4 está basada en la November 1992 ARPA Continuous Speech Recognition Corpora [45], también conocida como Nov'92 ARPA CSR.

Aurora-4 está pensada para evaluar la robustez de sistemas de ASR frente a condiciones de ruido aditivo, variación de micrófono, discordancia o *mismatch* entre condiciones de entrenamiento y evaluación, reducción de la frecuencia de muestreo, y compresión de datos [43]. Los experimentos realizados en esta tesis se enfocan en las tres primeras condiciones.

Los datos utilizados se dividen en bases de datos de entrenamiento, desarrollo y evaluación:

- **Entrenamiento.** Existen tres bases de datos de entrenamiento, llamadas *clean*, *multinoise* y *multicondition*. La base *clean* está compuesta por 7.138 elocuciones o *utterances* (15,2 horas) de 83 hablantes nativos de inglés, y contiene sólo audios limpios grabados con un micrófono de alta calidad: el Sennheiser HMD-414. La base *clean* corresponde a la base de datos WSJ0 SI 84 [46] de la Nov'92 ARPA CSR [45]. La base de datos *multinoise* contiene un cuarto de audios limpios y tres cuartos de audios degradados artificialmente con 6 tipos distintos de ruido agregados a SNRs entre 10 y 20 [dB]. Finalmente, en la base *multicondition* la mitad de los audios fue grabado con el micrófono Sennheiser HMD-414, mientras la otra mitad fue grabada con 18 diferentes micrófonos de baja calidad, y con el ruido agregado como en la base *multinoise*.

- **Desarrollo.** Por cada base de datos de entrenamiento se utilizó una base de datos de desarrollo o "dev" con las mismas cualidades acústicas. Para la base *clean* se utilizó la base de desarrollo llamada `dev_330_01`, que cuenta con 330 *utterances* limpias de 10 hablantes nativos [43]. Estos hablantes son distintos a los de la base de entrenamiento. Para la base *multicondition* se utilizó la base de desarrollo `dev_330` consistente de 4.620 *utterances*. Esta base de

datos viene incluida en Aurora-4 y consiste en 14 versiones de `dev_330_01`, incluyendo la versión original. Las 13 restantes variaciones se distribuyen de la siguiente manera: seis corresponden a `dev_330_01` agregando seis tipos de ruido aditivo; una, llamada `dev_330_08`, corresponde a las mismas *utterances* de `dev_330_01` pero grabadas con los micrófonos de baja calidad; y las 6 restantes corresponden a `dev_330_08` agregando los 6 tipos de ruido aditivo. Finalmente, para la base *multinoise* se utilizan 2.310 *utterances* que corresponden a los 7 primeros grupos de `dev_330`. Esta base de desarrollo fue denominada `dev_330_noisy`.

- **Evaluación.** La base de datos de evaluación está compuesta por 14 sets de test. Cada uno de ellos cuenta con 330 *utterances*. El primer set corresponde a audios limpios grabados en alta calidad; los demás sets son variaciones del primero, que corresponden a cambios en el micrófono, adición de ruido entre 5 y 15 [dB], o ambos. Estos sets pueden ser organizados en cuatro grupos según sus características, como se resume en la tabla 3.1.

Tabla 3.1: Descripción de los sets de evaluación de Aurora-4

Set	Micrófono	Ruido	Grupo
1	Sennheiser HMD-414	Clean	A
2	Sennheiser HMD-414	Car	B
3		Babble	
4		Restaurant	
5		Street	
6		Airport	
7		Train	
8	Diferentes tipos de mic.	Clean	C
9	Diferentes tipos de mic.	Car	D
10		Babble	
11		Restaurant	
12		Street	
13		Airport	
14		Train	

Todos los audios utilizados están muestreados a una frecuencia de 16 [kHz] y no se utilizó compresión de datos.

Por último, se debe indicar que las bases de datos de entrenamiento del corpus Aurora-4 no están etiquetadas, pero se cuenta con las transcripciones de cada elocución. Luego, los sistemas ASR que usen este corpus deben estimar el etiquetado de la base de datos usando las transcripciones disponibles, lo que se conoce como "alineamiento forzado" o simplemente "alineamiento".

3.2.2. Arquitectura del sistema

El sistema de reconocimiento de tipo DNN-HMM fue construido utilizando el software libre Kaldi Speech Recognition Toolkit [47]. Este software ha sido ampliamente utilizado en el ámbito académico en la evaluación e implementación de sistemas de ASR que utilizan técnicas de *Deep*

Learning.

Los componentes del software Kaldi se pueden clasificar en dos categorías: ejecutables y *scripts*. Los ejecutables son programas escritos en `C++` que implementan los objetos y clases necesarias para entrenar y mantener los modelos que hacen parte de un sistema ASR. Por otra parte, los *scripts* son guiones o archivos de procesamiento por lotes, que no requieren compilación, y que manejan las llamadas a los ya mencionados ejecutables. Los *scripts* de más alto nivel son llamados "recetas"; estas permiten diseñar un experimento en un sólo archivo. Dentro de las recetas se van llamando a *scripts* que realizan tareas más específicas como extracción de *features*, entrenamiento, alineamiento, decodificación, etc.

El primer paso en la construcción del sistema DNN-HMM es obtener el alineamiento de la base de datos de entrenamiento. Este se obtiene usando un sistema de reconocimiento GMM-HMM que se entrena utilizando la receta tri2b de Kaldi para Aurora-4 [47]. Se utiliza la base *clean* debido a que se obtiene una mejor tasa de reconocimiento [48]. La receta utiliza *features* MFCC, *linear discriminant analysis* (LDA) y *maximum likelihood linear transforms* (MLLT). Esta receta comienza por entrenar un primer sistema GMM-HMM de mono-fonos para alinear los datos, y usa esos alineamientos para entrenar un segundo GMM-HMM pero de trifonos. Luego se obtienen alineamientos con este segundo sistema para entrenar el tercer GMM-HMM de trifonos definitivo. Finalmente, este tercer sistema GMM-HMM se usa para obtener los alineamientos que servirán de referencia para el sistema DNN-HMM.

Además del sistema GMM-HMM final descrito anteriormente, se entrenaron otros dos sistemas GMM-HMM utilizando el mismo procedimiento, pero usando las bases *multinoise* y *multicondition*. Luego, se cuenta un GMM-HMM de trifonos entrenado en cada condición de entrenamiento.

Además, se evaluaran tres conjuntos de *features*:

1. *Features* Mel Filterbank estáticos y dinámicos, denominados "MelFB+ΔMelFB".

2. *Features* LNFB estáticos y dinámicos calculados usando las ecuaciones 3.2 y 3.3, denominados "LNFB+ΔLNFB1".

3. *Features* LNFB estáticos y dinámicos calculados usando las ecuaciones 3.6 y 3.7, denominados "LNFB+ΔLNFB2".

A pesar de que la notación anterior sólo aparece un operador Δ, siempre se incorporan los coeficientes delta-delta correspondientes; simplemente se dejaron fuera de la notación para hacerla más compacta.

Finalmente, se deben entrenar nueve sistemas DNN-HMM: uno para cada una de las combinaciones entre condiciones de entrenamiento y conjuntos de *features* que se evaluarán. Para construir esos sistemas DNN-HMM se copia el modelo HMM del sistema GMM-HMM correspondiente a la condición de entrenamiento, mientras que los GMM de cada estado se reemplazan por una DNN.

Cada una de las nueve DNN se pre-entrenó usando RBMs apiladas para formar una *Deep Belief Network* [7], y luego se entrenó usando el algoritmo de entropía cruzada (*cross entropy*) [49] y los alineamientos de la base *clean* obtenidos con el sistema GM3. Las bases de datos de desarrollo `dev_330_01`, `dev_330_noisy` y `dev_330` se usaron como datos de validación cruzada en el entrenamiento de las DNN con las bases de entrenamiento *clean*, *multinoise* y *multicondition*, respectivamente.

Finalmente, cada DNN tiene la misma arquitectura interna: constan de siete capas ocultas con 2.048 unidades por capa. El número de unidades en la capa de salida se iguala al número de fdps (funciones densidad de probabilidad) en el sistema GMM-HMM correspondiente, mientras que el número de entrada depende de los *features* utilizados.

3.2.3. Elección del número de *features*

Siguiendo la arquitectura presentada anteriormente, se entrenó un sistema DNN-HMM *baseline* usando *features* MelFB y la base de entrenamiento *multicondition*, que se sabe es la que tiene el mejor rendimiento en el corpus utilizado. Este sistema *baseline* fue entrenado usando 24, 32, 40 y 56 *features* estáticos MelFB. El WER más bajo, 10,9 %, fue obtenido usando 40 *features* y aplicando normalización MN por *speaker*. Este resultado es competitivo al compararlo con rendimientos de otros sistemas que han usado el mismo corpus y que han sido reportados en la literatura recientemente (ver [24], [40], [50]-[55]).

De manera similar, se usaron 40 *features* LNFB estáticos. El ancho de banda de los filtros de cada canal, B, fue de 5,2 [Bark]. El espacio entre filtros contiguos es función del número de filtros, y en este caso fue de aproximadamente 0,4 [Bark].

Con esta elección de *features*, la cantidad de unidades en la capa de entrada de cada DNN corresponde a 1.320. Este número es consecuencia de tener como entrada 11 *frames* consecutivos y 120 *features* representando a cada *frame* (40 estáticos, 40 deltas y 40 delta-deltas). La entrada de 11 *frames* se conoce como ventana de contexto: la salida de la DNN se corresponde con el *frame* central de la ventana, que utiliza un contexto de 5 *frames* hacia adelante y 5 hacia atrás.

3.3. Resultados

La primera evaluación realizada fue la eficacia de la definición de coeficientes dinámicos para LNFB. La normalización de la entrada que se utilizó en estos experimentos fue MN por *speaker*. El WER promedio sobre los 14 sets de test se muestran en la tabla 3.2.

Los resultados de la tabla 3.2 muestran que los coeficientes dinámicos propuestos ΔLNFB2 tienen mejor rendimiento que los calculados de manera directa (ΔLNFB1). Debido a esto, en los siguientes experimentos sólo se evalúan los *features* LNFB + ΔLNFB2.

En segundo lugar se evaluó el efecto de la normalización de la entrada en el rendimiento del sistema. La tabla 3.3 recopila los WER promedio de estos experimentos calculados sobre los 14

Tabla 3.2: Comparación de algoritmos para estimar coeficientes delta y delta-delta, usando el corpus Aurora-4. La tabla muestra el WER promediado sobre los 14 sets de test.

Entrenamiento	MelFB + ΔMelFB	LNFB + ΔLNFB1	LNFB + ΔLNFB2
Clean	32,66	49,21	36,73
Multinoise	16,00	18,42	16,06
Multicondition	10,90	13,97	12,26

sets de test.

Tabla 3.3: Variación del WER con la normalización de los *features* de entrada, en el corpus Aurora-4. La tabla muestra el WER promediado sobre los 14 sets de test.

Entrenamiento	Set de *features*	No Norm.	MN por Spk	MVN por Spk	MN por Utt	MVN por Utt
Clean	MelFB + ΔMelFB	49,95	32,66	30,74	28,74	27,12
	LNFB + ΔLNFB2	51,10	36,73	28,63	33,38	24,03
Multinoise	MelFB + ΔMelFB	18,11	16,00	17,12	13,97	14,73
	LNFB + ΔLNFB2	16,13	16,06	14,85	14,55	13,35
Multicondition	MelFB + ΔMelFB	11,95	10,90	11,93	10,23	10,62
	LNFB + ΔLNFB2	12,27	12,26	12,11	11,62	11,18

Para realizar un análisis más detallado, en la tabla 3.4 se incluyen los WER para cada grupo de test (ver tabla 3.1) para la normalización MVN por *utterance*. Se escogió esta normalización porque llevó al mejor rendimiento del sistema LNFB + ΔLNFB2 para el entrenamiento *multicondition*. En cada fila se ha indicado en negrita cuando la diferencia del WER de cada sistema es significativa al nivel de $p = 0,001$ o mejor. Para calcular el nivel de significancia se aplicó el test NIST MAPSSWE (*matched-pairs sentence-segment word error*) [56]. Este test sirve para determinar si existen diferencias significativas en las transcripciones encontradas por cada sistema.

Tabla 3.4: Resumen de resultados para los grupos de test de Aurora-4. En estos experimentos se usó normalización MVN por *utterance*.

Entrenamiento	Grupo Aurora-4	MelFB + ΔMelFB	LNFB + ΔLNFB2
Clean	A	2,39	2,65
	B	19,70	19,26
	C	21,69	**14,10**
	D	39,55	**34,02**
	Promedio	27,12	24,03
Multinoise	A	2,56	3,19
	B	**6,11**	6,94
	C	16,57	**11,84**
	D	25,06	**21,70**
	Promedio	14,73	13,35
Multicondition	A	3,42	3,62
	B	**6,35**	7,19
	C	7,12	7,38
	D	16,68	17,06
	Promedio	10,62	11,18
Promedio sobre todas las condiciones		17,5	16,2

3.4. Discusión

3.4.1. Rendimiento de los coeficientes dinámicos propuestos

La tabla 3.2 compara el WER promedio sobre todos los sets de test para los tres tipos de conjuntos de *features* en evaluación. En cada sistema, estos tres conjuntos fueron normalizados usando MN por *speaker*. En todos los casos, la combinación de *features* LNFB estáticos con los coeficientes delta y delta-delta basados en la energía del filtro del numerador, LNFB+ΔLNFB2 llevaron a un mejor resultado que LNFB+ΔLNFB1. Las reducciones relativas promedio en la tasa de error son iguales a 25,4 %, 12,8 % y 12,2 % para los sistemas con entrenamiento *clean*, *multinoise* y *multicondition*, respectivamente.

Las reducciones alcanzadas en la tasa de error indican claramente que los coeficientes dinámicos ΔLNFB2 basados solo en los filtros del numerador proveen más información útil que los coeficientes ΔLNFB1 que están basados directamente en los filtros normalizados LNFB.

3.4.2. Efecto del tipo de normalización aplicada

La tabla 3.3 compara los WER promedio que se obtienen al variar la normalización aplicada a los *features* utilizados. El WER promedio se calcula sobre los 14 sets de test de Aurora-4. Se tienen 5 configuraciones de normalización: Sin normalizar (abreviado como No Norm.), y MN y MVN aplicado por *speaker* y por *utterance*.

Es posible observar que la normalización MN siempre lleva a un WER más bajo cuando se aplica por *utterance* en vez de por *speaker*. Lo mismo sucede con la normalización MVN. Aplicar normalización por *utterance* en vez de por *speaker* lleva a una reducción promedio en el WER de 9,1 %. Esto puede deberse a que el SNR en la base de datos Aurora-4 cambia de una *utterance* a la siguiente de manera aleatoria.

La tabla 3.3 también muestra que el uso de MVN en vez de MN provee una normalización más efectiva en un sistema DNN-HMM con los *features* LNFB+ΔLNFB2. Esta diferencia es particularmente dramática en el caso de entrenamiento *clean*. En contraste, MN fue más efectivo que MVN para los *features* MelFB al ser usado en conjunto con los entrenamientos *multinoise* y *multicondition*. Esto puede ser debido a que los *features* LNFB siempre producen histogramas unimodales, y por consiguiente, el uso de MVN es más útil que MN con los *features* MelFB.

3.4.3. Comparación de LNFB con MelFB

La tabla 3.4 compara los WER obtenidos al usar LNFB+ΔLNFB2 y MelFB+ΔMelFB, divididos por condiciones de test y entrenamiento. En cada caso se aplicó normalización MVN por *utterance*. Como se mencionó, un resultado en negrita indica que el *feature* utilizado es significativamente mejor al nivel de $p = 0,001$ o superior, de acuerdo al test MAPSSWE [56].

El foco de este análisis se pone en las comparaciones que son estadísticamente significativas.

Al revisar los resultados obtenidos con entrenamiento *clean* y *multinoise*, se observa que LNFB+ΔLNFB2 provee un rendimiento superior para los grupos C y D de Aurora-4. Esto puede deberse a que, en estos experimentos, los micrófonos usados en los datos de entrenamiento y test son diferentes.

La única combinación entrenamiento/test en donde MelFB+ΔMelFB provee un rendimiento significativamente mejor que LNFB+ΔLNFB2 fue para el grupo B al usar entrenamiento *multinoise* y *multicondition*. En el primer caso, las condiciones acústicas de entrenamiento y test coinciden completamente, luego no se espera ningún beneficio al usar LNFB+ΔLNFB2. En el segundo caso, se piensa que cualquier beneficio potencial proveniente de la disparidad en las condiciones acústicas fue invalidado por el predominio de la adaptación del entrenamiento al ruido aditivo.

En general, los resultados de la tabla 3.4 confirman la observación preliminar de que los coeficientes LNFB son especialmente efectivos en compensar el efecto de *mismatch* en el canal acústico. Esto había sido demostrado previamente para los *features* LNCC (ligados a los LNFB) en la tarea de verificación de locutor [1], [36].

El promedio global del WER (promediado sobre todos los test y todas las condiciones de entrenamiento) obtenido al usar *features* basados en LNFB fue 7,4 % relativamente más bajo que el WER obtenido al usar *features* MelFB.

Capítulo 4

Combinación de sistemas DNN-HMM para robustez en ambientes reverberantes

En esta sección se propone combinar un modelo acústico DNN entrenado con parametrización LNFB junto con otros modelos entrenados con distintas parametrizaciones. El objetivo es evaluar si existe ventaja en usar distintas parametrizaciones acústicas. La efectividad de esta técnica será medida en una base de datos reverberante.

4.1. Combinación y complementariedad de DNNs

4.1.1. Motivación: Importancia de la parametrización acústica en el entrenamiento de una DNN.

Es bien sabido que la modelación acústica que puede lograr una DNN es superior a los típicos modelos GMM. Esto puede ser resultado de que una DNN puede aprender representaciones robustas de la voz en sus capas más profundas. Estas representaciones son aprendidas en conjunto y simultáneamente cuando la DNN es entrenada como clasificador. Como resultado, el uso de señales sin procesamiento (*raw signals*) como entrada de la DNN se ha convertido en una opción admisible. Más aún, varios autores han sugerido que *features* diseñados a mano como los PLP o coeficientes cepstrales no son esenciales para alcanzar un rendimiento alto en tareas ASR [57]-[59]. Estos trabajos han consolidado la idea de que solamente se necesita una gran cantidad de datos para entrenar un sistema ASR basado en *deep learning*, independientemente de cómo las señales de entrada se han pre-procesado. En cambio, algunos autores sostienen que aún es útil combinar distintos tipos de *features* para aprender nuevas representaciones de la voz [60], [61].

Para dimensionar la importancia de la parametrización acústica en el rendimiento de un sistema ASR, se describe a continuación a grandes rasgos cómo es el proceso de entrenamiento de una DNN. De esta manera se puede analizar el impacto de la parametrización en la determinación de los parámetros de la red.

El proceso de entrenamiento de una DNN se formula como un problema de optimización del set de parámetros $\{\boldsymbol{W}, \boldsymbol{b}\}$ (pesos y *biases*) que definen la DNN. Dado un conjunto de datos de entrenamiento y el correspondiente conjunto de referencias, el proceso de entrenamiento debe determinar los valores óptimos de $\{\boldsymbol{W}, \boldsymbol{b}\}$ que minimizan una función de error entre las referencias y la salida de la DNN.

Un procedimiento común es primero aplicar un pre-entrenamiento. Este es un proceso de aprendizaje no-supervisado que estima valores iniciales para $\{\boldsymbol{W}, \boldsymbol{b}\}$, los cuales son un buen punto de partida para la etapa de entrenamiento supervisado. En el entrenamiento, el famoso algoritmo de *backpropagation* es empleado con alguna función de error o criterio para calcular el set final de parámetros. El criterio más comúnmente usado en tareas de clasificación (que es el caso de ASR) es la entropía cruzada o *cross-entropy* (CE)[62].

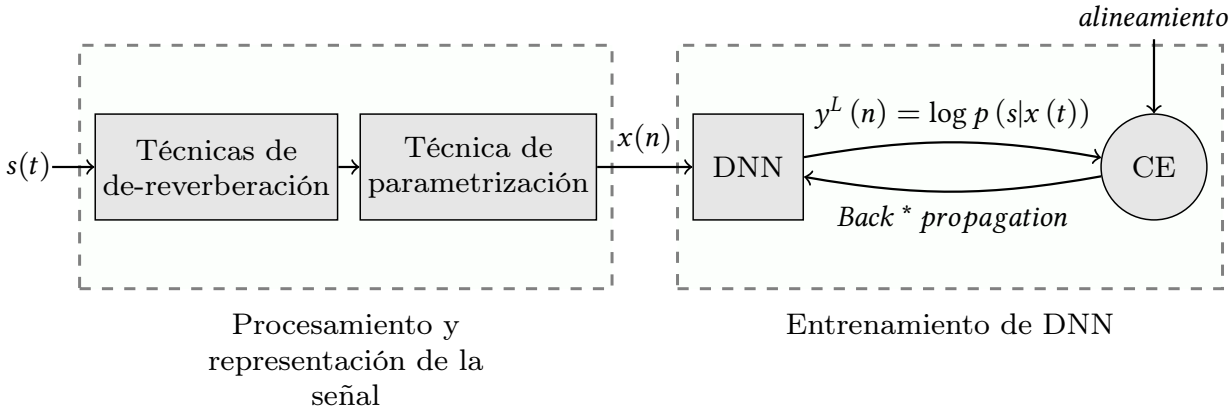


Figura 4.1: Procesamiento y parametrización de la señal, y el proceso de entrenamiento de una DNN. CE representa el cálculo de la entropía cruzada o *cross-entropy*

La figura 4.1 ilustra el proceso de entrenamiento de la DNN. En el bloque de pre-procesamiento se realiza la parametrización de los datos, y también se podría incluir la remoción o compensación de distorsiones en los datos, como por ejemplo técnicas de normalización o de reducción de reverberación. Luego del pre-entrenamiento se aplica el algoritmo de *backpropagation*. Este algoritmo se basa en el cálculo de gradientes locales. En el proceso iterativo de entrenamiento, los gradientes locales son usados para estimar la corrección que se realiza a los parámetros de la DNN:

$$\Delta w_{ji} = \eta \delta_j y_i \tag{4.1}$$

donde Δw_{ji} es la corrección aplicada al peso que conecta a la neurona i con la neurona j, δ_j es el gradiente local asociado a la neurona j, e y_i es la salida de la neurona i y la entrada i-ésima a la neurona j. Si la neurona j pertenece a la primera capa oculta de la red, entonces la ecuación 4.1 se transforma en:

$$\Delta w_{ji} = \eta \delta_j x_i \tag{4.2}$$

donde x_i es el *feature* de entrada i-ésimo. En consecuencia, si cambia la parametrización u otro pre-procesamiento aplicado a los datos de entrada, el proceso de entrenamiento llevará a diferentes soluciones para $\{\boldsymbol{W}, \boldsymbol{b}\}$ incluso si se usan los mismos datos de entrenamiento y la misma inicialización de parámetros. Luego, se podrían obtener DNNs complementarias solo al modificar la representación de los datos de entrada.

Es sabido que el incremento en el tamaño del set de entrenamiento puede llevar a una reducción en el WER, independientemente del pre-procesamiento realizado a los datos. Sin embargo, de acuerdo a lo discutido anteriormente, dos DNNs entrenadas con diferentes representaciones acústicas podrían proveer información complementaria aunque alcancen el mismo WER de forma individual. Luego, la complementariedad no debería depender del tamaño de los datos de entrenamiento. Como ejemplo, en la figura 4.2 se puede ver las distribuciones de las salidas (seudo *log-likelihoods* o LLKs) de DNNs entrenadas con la misma base de datos. Estas distribuciones pueden diferir significativamente dependiendo de la parametrización utilizada.

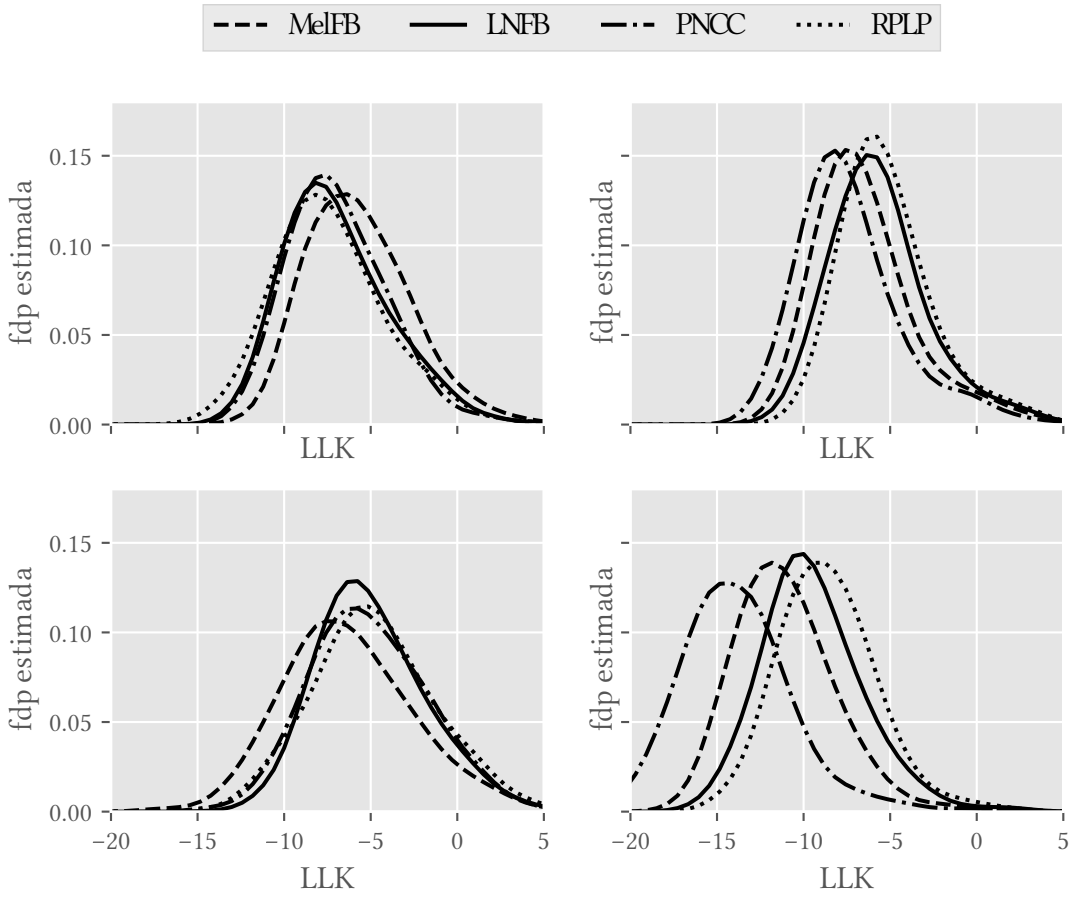


Figura 4.2: Distribuciones de probabilidad (pdf) estimadas para seudo log-likelihoods (LLK) entregadas por DNNs entrenadas con cuatro métodos de parametrización: MelFB, LNFB, PNCC y PLP-Rasta (RPLP). Las cuatro DNN fueron entrenadas con la misma base de datos, y no se aplicó ninguna técnica de pre-procesamiento adicional. Cada subfigura representa las pdf de cuatro *frames* diferentes de una misma *utterance*.

Existen varios métodos para combinar clasificadores, como por ejemplo regla del máximo,

regla del mínimo, regla del promedio, regla del producto, regla de mayoría ponderada de votos y la combinación lineal [63]-[65]. En este trabajo se usará precisamente la combinación lineal de clasificadores para combinar DNNs.

En una tarea de ASR, la capa de salida de la DNN puede ser vista como un vector de puntajes o *scores* que entrega la red para cada *frame* de entrada. Luego, la combinación de DNNs intenta combinar los *scores* entregados por cada una de las redes a combinar. Dadas R DNNs, la combinación lineal de sus *scores* se define según la ecuación 4.3 :

$$\hat{m}(s,n) = \sum_{r=1}^{R} \omega_{r,s,n} \cdot m_r(s,n) \tag{4.3}$$

donde $\hat{m}$ es el *score* combinado para la salida s-ésima de las DNN generada por el *frame* de entrada n-ésimo; $m_r(s,n)$ es el *score* provisto por la DNN r-ésima; y $\omega_(r,s,n)$ es el peso de combinación correspondiente. Estos pesos pueden ser considerados dependientes o independientes de la salida s-ésima o del frame n-ésimo según la técnica utilizada para combinar las redes.

Se debe notar que es pertinente usar el término *score* para referirse a la salida de la DNN, ya que dependiendo de la aplicación, esta salida podría representar el log-*likelihood* o bien la log-probabilidad *a posteriori*.

4.1.2. Combinación basada en autoencoder

Recientemente se ha propuesto un método de combinación de clasificadores basada en *autoencoder* (AE) para la tarea de ASR [66]. En este método, la capa pre-softmax (log-probabilidad *a posteriori*) de cada DNN se modela con un AE entrenado con los mismos datos usados para entrenar la DNN correspondiente. En la etapa de test, el AE asociado con la DNN r provee un error de reconstrucción $e_{r,n}$ para el *frame* n. Este error es usado para calcular los pesos de combinación de la ecuación 4.3 según la ecuación :

$$\omega_{r,n} = \frac{1/\|e_{r,n}\|^2}{\sum_{k=1}^{R} 1/\|e_{k,n}\|^2} \tag{4.4}$$

donde el operador $\|\cdot\|$ representa la norma l_2. Los pesos óptimos son calculados *frame* por *frame*, pero son iguales para todos los estados (salidas de la DNN) dado un *frame* fijo n. Esta técnica fue probada originalmente para combinar dos DNNs entrenadas con diferentes parametrizaciones, RASTA y PLP, en una base de datos de entrenamiento *clean* [66]. Luego, esta técnica fue empleada para combinar seis DNNs entrenadas con diferentes condiciones de reverberación y usando la parametrización MelFB [67].

4.1.3. Estimación de peso de combinación por estado

Tomando como referencia el trabajo presentado en [64] sobre combinación de clasificadores, se realiza una estimación de los pesos de la ecuación 4.3 para cada estado usando la técnica MMSE (*Minimum Mean Square Error*). Los pesos se estiman de manera que se minimice el error cuadrático entre la salida combinada de la DNN y los alineamientos de los datos de entrada.

Se debe recordar que el alineamiento corresponde a las etiquetas de cada *frame* en los datos de entrenamiento, e indican cuál es el estado correcto que corresponde a cada *frame*. Por lo tanto, la etiqueta de cada *frame* puede representarse como un vector de probabilidades del tamaño de la capa de salida de la DNN, y que tiene un 1 en el nodo que representa el estado al que pertenece el *frame*, y un 0 en el resto de estados.

El error cuadrático entre el alineamiento y la salida de la DNN, calculado a lo largo de todos los Q estados del sistema y las U *utterances* de entrenamiento, se expresa en la ecuación 4.5:

$$\mathcal{E} = \sum_{u=1}^{U} \sum_{n=1}^{N_u} \sum_{s=1}^{Q} \left[a_{u,s}\left(x_{u,n}\right) - \hat{p}\left(s|x_{u,n}\right) \right]^2 \tag{4.5}$$

donde $a_{u,s}(x_{u,s})$ representa la etiqueta del *frame* n para la *utterance* u y el estado s, N_u es el número de *frames* en la *utterance* u, y $\hat{p}(s|x_{u,n})$ es la combinación de las probabilidades *a posteriori* para la entrada $x_{n,u}$ y el estado s. Como se mencionó anteriormente, el valor de $a_{u,s}(x_{u,n})$ es 1 si $x_{u,n}$ pertenece al estado s, y 0 si no.

El valor de $\hat{p}$ puede ser calculado usando la ecuación 4.3 si se consideran las probabilidades *a posteriori* como *score*, tal como se ve en la ecuación 4.6:

$$\mathcal{E} = \sum_{u=1}^{U} \sum_{n=1}^{N_u} \sum_{s=1}^{Q} \left[a_{u,s}\left(x_{u,n}\right) - \sum_{l=1}^{R} \omega_{l,s} \cdot p_l\left(s|x_{u,n}\right) \right]^2 \tag{4.6}$$

donde R es la cantidad de DNNs que se están combinando, $\omega_{l,s}$ es el peso de combinación para la DNN l en el estado s, y $p_l\left(s|x_{u,n}\right)$ es la probabilidad *a posteriori* del estado s entregada por la DNN l para la entrada $x_{u,n}$. En la ecuación 4.6 se observa que los pesos $\omega_{l,s}$ son independientes del *frame* de entrada, es decir, no dependen del subíndice n.

Para obtener los pesos de combinación que minimizan el error cuadrático se calcula la derivada parcial del error $\mathcal{E}$ de la ecuación 4.6 con respecto a cada uno de los pesos. Para una DNN r y un estado s, la derivada parcial se calcula según la ecuación 4.7:

$$\frac{\partial \mathcal{E}}{\partial \omega_{r,s}} = \sum_{u=1}^{U} \sum_{n=1}^{N_u} 2 \cdot \left[a_{u,s}\left(x_{u,n}\right) - \sum_{l=1}^{R} \omega_{l,s} \cdot p_l\left(s|x_{u,n}\right) \right] \cdot -p_r\left(s|x_{u,n}\right) \tag{4.7}$$

Para obtener los pesos de combinación que minimizan el error, se iguala cada derivada parcial $\partial\varepsilon/\partial\omega_{r,s}$ a 0. Este procedimiento lleva a tener rxs ecuaciones que conforman un sistema de ecuaciones lineales. Para ello, es útil definir los siguientes vectores:

$$\boldsymbol{\omega_s} = [\omega_{1,s} \dots \omega_{R,s}]^t \tag{4.8}$$

$$\boldsymbol{a_s} = [a_{1,s}(x_{1,1}) \dots a_{1,s}(x_{1,N_1}) \dots a_{U,s}(x_{U,1}) \dots a_{U,s}(x_{U,N_U})]^t \tag{4.9}$$

$$\boldsymbol{p_{r,s}} = [p_r(s|x_{1,1}) \dots p_r(s|x_{1,N_1}) \dots p_r(s|x_{U,1}) \dots p_r(s|x_{U,N_U})]^t \tag{4.10}$$

donde $\boldsymbol{\omega_s}$ es el vector que agrupa los pesos de combinación para el estado s y cada una de las R DNNs, $\boldsymbol{a_s}$ es el vector que concatena los valores de las etiquetas $a_{u,s}$ para todas las *utterances* de entrenamiento, y $\boldsymbol{p_{r,s}}$ es el vector que concatena todos los *scores* entregados por la DNN r para el estado s. Con estas definiciones es posible obtener la expresión 4.11 con la que se pueden calcular los pesos óptimos de combinación:

$$\boldsymbol{\omega_s} = \left(\boldsymbol{P_s^t P_s}\right)^{-1} \boldsymbol{P_s^t a_s} \tag{4.11}$$

donde $\boldsymbol{P_s} = [\boldsymbol{p_{1,s}} \dots \boldsymbol{p_{R,s}}]$ se define como la matriz que agrupa los vectores $\boldsymbol{p_{r,s}}$ correspondientes a todas las DNN en el estado fijo s.

Se debe notar que las dimensiones de las matrices $\boldsymbol{P_s^t P_s}$ y $\boldsymbol{P_s^t a_s}$ son RxR y $Rx1$, respectivamente, lo que no depende del número de *frames* utilizados. Como resultado, los pesos de combinación $\omega_{r,s}$ pueden ser estimados usando la ecuación 4.11 sobre todos los datos de entrenamiento sin necesidad invertir matrices grandes.

También es posible definir un peso promedio a lo largo de todos los estados de una DNN r, lo que se define en la ecuación 4.12:

$$\bar{\omega}_r = \frac{\sum_{s=1}^{Q} \omega_{r,s}}{Q} \tag{4.12}$$

4.1.4. Otros métodos relacionados

Cabe mencionar que recientemente se ha abordado el tema de combinación lineal de DNN en el marco de ASR en ambientes reverberantes [68]. En [68] se propone un método para encontrar el peso de combinación óptimo *frame* a *frame*, basado en información mutua. A su vez, dicha propuesta está basada en un método de combinación lineal de clasificadores basada en información mutua para la tarea de verificación de locutor [65].

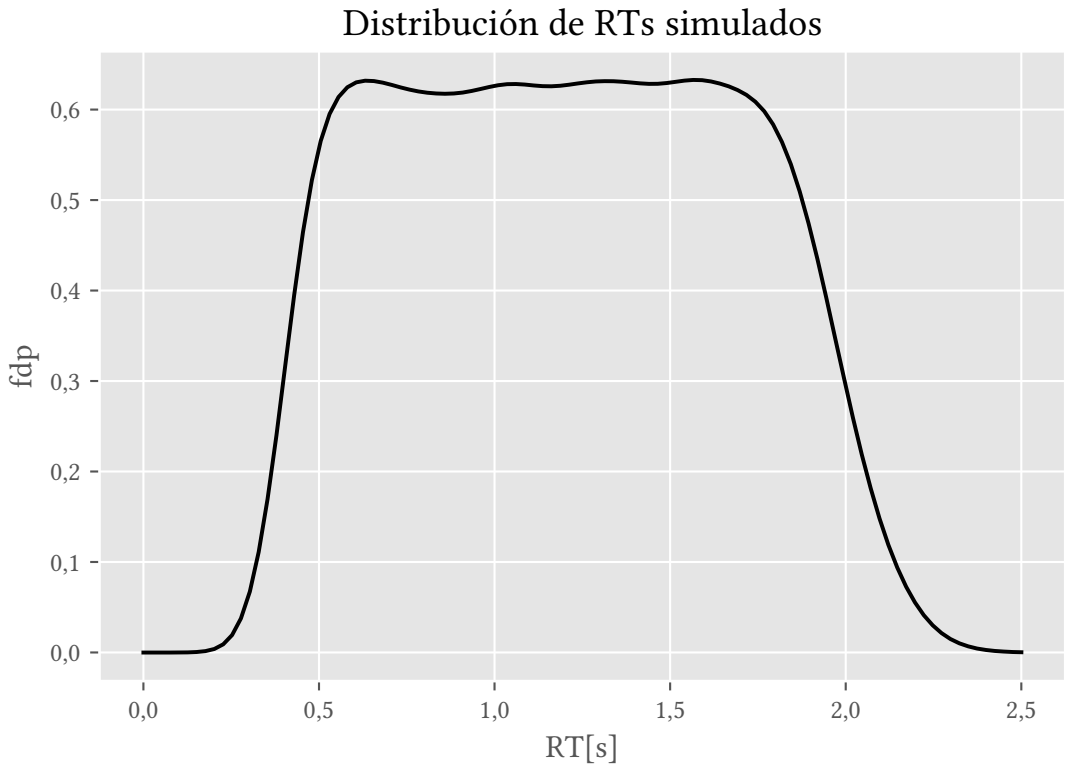


Figura 4.3: Distribución de densidad de probabilidad (fdp) de los tiempos de reverberación (RTs) resultantes del proceso de simulación de los RIRs usados en los experimentos.

4.2. Rendimiento de la combinación de DNNs en ambientes reverberantes

La efectividad de las técnicas de combinación descritas se evaluará en una tarea de ASR bajo condiciones reverberantes. Se elige este ambiente de prueba por ser uno de los más difíciles de abordar. Para estos experimentos se utilizó datos de prueba que están disponibles públicamente y que corresponden a grabaciones reales de audios realizadas en una cámara reverberante con tiempos de reverberación entre 0,47[s] y 1,77[s] [69].

4.2.1. Base de datos

Se generó una base de datos de entrenamiento *multicondition* basada en el la base de datos *clean* WSJ0 SI-284. Esta última está compuesta por 81 horas de audios limpios. La idea de la base de datos *multicondition* es simular distintas condiciones de reverberación. Para ello, cada *utterance* de la SI-284 fue convolucionada con tres diferentes RIRs que se seleccionaron de forma aleatoria desde una lista de 30.000 RIRs simulados.

Todos los RIRs fueron simulados usando el *software Room Impulse Response Generator* [70]. En este *software* se ingresan las dimensiones de una sala virtual para obtener el RIR correspondiente a la sala. Para genera los 30.000 RIRs, los parámetros de la sala se escogieron de la siguiente

manera:

- **Tamaño de la sala:** Se definieron los valores nominales 7,95[m], 5,68[m], y 4,5[m] para las dimensiones largo, ancho y alto de la sala, respectivamente. Luego, para generar cada RIR, las dimensiones se obtuvieron de distribuciones uniformes definidas sobre rangos de más y menos 20 % los valores nominales. La motivación para variar las dimensiones de la sala fue que no hubiera una coincidencia exacta con las dimensiones reales de la sala reverberante que fue usada para grabar los datos de test.

- **Distancia *speaker*-micrófono:** La distancia *speaker*-micrófono se obtuvo de una distribución uniforme entre 0,144[m] y 2,816[m]. Además, para cada RIR, el *speaker* y el micrófono fueron colocados en lugares aleatorios dentro de la sala virtual, usando la distancia escogida, con la restricción de que tanto el *speaker* como el micrófono se ubicaran al menos a 1[m] de cualquier pared, y entre 1[m] y 2[m] del piso.

La aleatoriedad de los parámetros de simulación reduce los potenciales efectos de artefactos causados por el fenómeno de onda estacionaria, que ocurren en simulaciones basadas en el método de imagen [71], como es el caso del *software* utilizado.

Con el procedimiento descrito anteriormente se obtuvo que los tiempos de reverberación (RT) de los RIRs generados varían entre 0,4 y 1,99 segundos; la distribución de estos RTs se muestra en la figura 4.3. La base de datos *multicondition* resultante tiene una duración de 325 horas, de las cuales el 25 % corresponden a *utterances* sin distorsión (*clean*). El tamaño de esta base de datos es comparable a la existente en la tarea Switchboard [72].

Para realizar los tests se utilizó la base de datos *Highly-reverberant Real Environments* (HRRE) [69]. Esta base de datos está compuesta de *utterances* del set de evaluación *clean* de Aurora-4 que fueron re-grabadas en una sala reverberante real. En la grabación se varió la distancia *speaker*-micrófono y el tiempo de reverberación para obtener diferentes condiciones de grabación. Las distancias *speaker*-micrófono fueron 0,16[m], 0,32[m], 0,64[m], 1,28[m] y 2,56[m], mientras que los RTs medidos en la sala fueron 0,47[s], 0,84[s], 1,27[s] y 1,77[s]. En total, se cuenta con 20 combinaciones de distancia *speaker*-micrófono y RT. Cada combinación está compuesta de 330 *utterance* re-grabadas.

4.2.2. Arquitectura del sistema

Para estos experimentos se entrenaron cuatro sistemas para ASR basado en DNN-HMM usando cuatro tipos de parametrizaciones: MelFB convencional, LNFB, PNCC [33] y RASTA-PLP (RPLP) [30].

Los cuatro sistemas se entrenaron usando un procedimiento similar al descrito en la sección 3.2.2. La arquitectura de cada DNN también es equivalente a la descrita en la sección 3.2.2.

4.3. Resultados

En primer lugar se realizaron experimentos *baseline* para los cuatro tipos de *features* en evaluación: MelFB, LNFB, PNCC y RPLP. Para entrenamiento se utilizó la base de datos *multicondition* descrita en la sección 4.2.1, y la base de datos de test HRRE. En estos experimentos (y en los siguientes) se aplicó la técnica de mejoramiento WPE (*Weighted Predictor Error*), tanto en los datos de test como de entrenamiento. Los resultados se muestran en las tablas 4.1 y 4.2, en donde la tasa de error WER se promedió a lo largo de las distancias *speaker*-micrófono y RT, respectivamente.

Tabla 4.1: WER (%) usando *features* MelFB, LNFB, PNCC y RPLP. Los resultados fueron obtenidos al usar entrenamiento *multicondition* y evaluando con la base de datos HRRE. WPE se aplicó a los datos de entrenamiento y test. El WER fue promediado para todas las distancias *speaker*-micrófono para cada RT.

Feature	RT[s]				Promedio
	0,5	0,9	1,3	1,7	
MelFB	3,33	4,80	6,62	9,28	6,01
LNFB	3,69	5,14	6,70	9,74	6,32
PNCC	3,37	5,05	6,77	9,56	6,19
RPLP	4,35	6,79	8,74	12,04	7,98

Tabla 4.2: Las condiciones son las mismas que la tabla 4.1, excepto que el WER está promediado para todos los RTs para cada distancia *speaker*-micrófono.

Feature	Distancia *speaker*-micrófono [m]					Promedio
	0,16	0,32	0,64	1,28	2,56	
MelFB	3,35	3,91	4,85	7,88	10,05	6,01
LNFB	3,15	4,10	5,40	8,26	10,68	6,32
PNCC	2,82	3,81	4,88	8,13	11,32	6,19
RPLP	3,71	4,59	6,29	10,50	14,82	7,98

Luego de obtener los resultados para cada parametrización por sí sola, se procedió a realizar las combinaciones de las salidas de los sistemas previamente entrenados. En la tabla 4.3 se muestran los resultados de todas las posibles combinaciones de pares de parametrizaciones, para cada una de las cuatro técnicas de combinación es estudio. En esta ocasión no se presentan los resultados por RT, si no que se presenta el promedio de los resultados sobre los 5 RTs.

Tabla 4.3: WER (%) promediado sobre todos los 20 subconjuntos de test de la base de datos HRRE. Se probaron todas las posibles combinaciones de dos tipos de *features*. Se ocupó la base de entrenamiento *multicondition* definida en la sección 4.2.1. Se aplicó WPE tanto a los datos de entrenamiento como test.

Combinación	MelFB -LNFB	MelFB -PNCC	MelFB -RPLP	LNFB -PNCC	LNFB -RPLP	PNCC -RPLP	Promedio
Peso constante (0,5)	5,28	5,32	5,35	5,45	5,50	5,74	5,44
Autoencoder	5,26	5,32	5,37	5,45	5,50	5,76	5,44
Peso por estado	5,39	5,29	5,43	5,52	5,68	5,79	5,52
Peso por estado promedio	5,37	5,28	5,42	5,50	5,66	5,78	5,50

Finalmente, se realizó una combinación utilizando los cuatro sistemas entrenados. Estos resultados se muestran en la tabla 4.4.

Tabla 4.4: WER (%) obtenido al combinar las cuatro DNNs.

	Flat Weight	Peso por estado	Peso por estado promedio
MelFB – LNFB – PNCC – RPLP	4,95	5,21	5,14

4.4. Discusión

4.4.1. Experimentos *baseline*

Los resultados de los experimentos *baseline* se muestran en las tablas 4.1 y 4.2, en donde el WER fue promediado por distancia *speaker*-micrófono y RT, respectivamente. Se observa que cada uno de los *features* evaluados tuvieron un rendimiento similar. La excepción es el *feature* RPLP, que llevó a resultados un poco peores.

4.4.2. Combinación basada en autoencoder

Las 4 DNNs entrenadas descritas en la sección 4.2.2 se combinaron en pares usando el método de autoencoder de Mallidi [66]. Esta técnica llevó a una reducción relativa promedio en WER igual a 9,5 % cuando se compara con los resultados *baseline* usando MelFB (tablas 4.1 y 4.2).

Sorprendentemente, el peso de combinación óptimo estimado por los AEs es aproximadamente igual a $0,5 \pm 0,001$ para cada par de combinación. Esto indica que el AE provee una métrica que depende más de la discordancia entre las condiciones de entrenamiento y test que de la

representación de la señal. Debido a que se usó la misma base de datos para entrenar los sistemas asociados a los cuatro *features* evaluados, los errores de reconstrucción provistos por cada AE fue similar. Este resultado sugiere que la combinación basada en AE no discrimina ni la precisión de clasificación de la DNN ni las condiciones de evaluación de los 20 test de HRRE [69].

4.4.3. Estimación de peso de combinación por estado

El uso de pesos estimados por estado llevó a una mejora relativa promedio en WER iguales a 8,2 % con respecto al mejor clasificador presentado en las tablas 4.1 y 4.2. Al usar el peso promedio de combinación a lo largo de los estados (ecuación 4.12) se obtuvo una mejora relativa promedio en WER igual a 8,5 %; sin embargo, este aumento en la mejora no es significativo de acuerdo al test MAPSSWE [56].

4.4.4. Combinación de los cuatro sistemas

Cuando se combinaron dos DNNs, el mejor resultado se obtuvo cuando se combinaron los sistemas asociados a MelFB y LNFB utilizando el método del AE (ver sección 4.1.2, obteniéndose a una mejora relativa de 12,5 %. Por otro lado, al usar la estimación de peso por estado, el mejor resultado se obtuvo al combinar los sistemas con MelFB y PNCC, llegando a una mejora relativa de 12 %.

La tabla 4.4 recopila los resultados obtenidos al combinar los clasificadores asociados a los cuatro tipos de *features* probados (MelFB, LNFB, PNCC y RPLP). El mejor resultado se obtuvo al usar la combinación plana, la cual esencialmente promedia los *scores* entregados por cada clasificador. Por razones prácticas, no se realizó el barrido de parámetros para determinar los pesos de combinación óptimos.

La combinación uniforme produjo una mejora relativa en WER de 17,6 % relativa al clasificador *baseline* asociado a MelFB. Por otro lado, la estimación de pesos por estado y el peso promedio por estado llevan a mejoras de 13,3 % y 14,5 %, respectivamente.

Es cierto que la combinación uniforme tuvo un mejor rendimiento, pero los métodos de combinación por estado podrían haber sido superiores si los WERs de los clasificadores individuales hubieran sido menos parecidos.

Capítulo 5

Conclusiones y Trabajo Futuro

En esta tesis se aplicaron a ASR los *features* correspondientes a filtros normalizados localmente o LNFB (*Locally Normalized Filter Banks*). Se utilizó un vocabulario de tamaño medio de la base de datos Aurora-4. El sistema de ASR fue basado en DNN-HMM. Al realizar un análisis de los coeficientes dinámicos delta y delta-delta, se mostró que estos debían ser estimados usando sólo el término numerador de la definición de LNFB. Adicionalmente, se mostró que la normalización MVN fue más efectiva que MN para estos *features*.

Al comparar los *features* LNFB con los MelFB, se encontró que el WER promedio global sobre todas las condiciones acústicas evaluadas fue un 7,4 % relativamente menor. Estos resultados indican que los LNFB proveen una mejor precisión en el reconocimiento de voz en sistemas DNN-HMM al compararse con los *features* MelFB, especialmente cuando hay *mismatch* de canal entre entrenamiento y test.

También se evaluó la complementariedad de las DNNs de los sistemas de ASR entrenadas con la misma base de datos, pero con diferentes parametrizaciones usando señales reverberadas. Se presentó un método basado en MMSE para combinar las salidas de las DNNs mostrando que se pueden alcanzar mejoras significativas en WER al combinar los *scores* de sistemas ASR que fueron entrenados usando distintos tipos de *features*, obteniendo una mejora relativa de 8 % al combinar dos clasificadores, y de 15 % al combinar cuatro. Estas mejoras se obtuvieron al probar el sistema propuesto usando audios grabados en un ambiente real altamente reverberante. Sin embargo, realizar la combinación usando pesos uniformes (promediando los *scores*) llevó a reducciones en WER de 11 % y 18 % con dos y cuatro clasificadores, respectivamente, comparando con el mejor clasificador individual. Este resultado puede atribuirse a que todos los sistemas DNN-HMM tenían un elevado nivel de complementariedad por lo que la combinación más simple resultó suficiente.

Se sugiere en general que los *features* diseñados no agregan valor adicional a los procesos de reconocimiento de patrones, particularmente cuando la cantidad de datos de entrenamiento es suficientemente grande [57], [59], [61]. Esto debido a que un clasificador basado en DNN tiene la habilidad de aprender representaciones complejas de su entrada automáticamente. De hecho, la

mejora que se alcanza en WER al pasar de un modelo acústico basado en GMM a uno basado en DNN es del orden de 30 %[7]. Sin embargo, en este trabajo se mostró que *features* diseñados y la combinación de estos pueden llevar a reducciones significativas en WER con un sistema ASR basado en DNN-HMM, sobre todo cuando la cantidad de datos de entrenamiento es limitada.

Finalmente, los resultados mostrados en esta tesis permiten definir algunas líneas de investigación como trabajo futuro: se propone evaluar distintas arquitecturas de autoencoders para la combinación de sistemas de ASR; se propone realizar un estudio del impacto del tamaño de la base de datos en el uso de *features* diseñados y en la combinación de estos en sistemas DNN-HMM; y se propone aplicar las técnicas propuestas en este trabajo en tareas de ASR con bases de datos en español.

Glosario

AE *Auto Encoder*. 36
ASR *Automatic Speech Recognition* o Reconocimiento automático de voz. 1, 10

DCT *Discrete Cosine Transform* o Transformada de coseno discreta. 14
DNN *Deep Neural Network* o Redes neuronales profundas. 1

FBANK Parametrización Filter Bank. 14

GMM *Gaussian Mixture Model* o Modelo de mezcla de gaussianas. 5

HMM *Hidden Markov Models* o Modelos ocultos de markov. 2
HRRE *Highly-reverberant Real Environments*. 41

LNCC *Locally Normalized Cepstral Coefficients*. 13
LNFB *Locally Normalized Filter Bank* o Banco de filtros localmente normalizado. 1, 2

MFCC *Mel Filter Bank Cepstral Coefficients*. 13
MLP *Multilayer Perceptron* o Perceptrón multicapa. 9
MMSE *Minimum Mean Square Error* o Mínimo error cuadrático medio. 37
MN *Mean Normalization* o Normalización de media. 19, 23
MVN *Mean and Variance Normalization* o Normalización de media y varianza. 19, 23

PLP *Perceptual Linear Predictive*. 13
PNCC *Power-Normalized Cepstral Coefficients*. 13

RASTA *Relative Spectral Transform*. 13
RT *Reverberation Time* o Tiempo de reverberación. 41

STFT Short-Term Fourier Transform o Transformada de Fourier de tiempo corto. 14

WER *Word Error Rate* o Tasa de error de palabras. 8
WPE *Weighted Predictor Error*. 41

Bibliografía

[1] V. Poblete, F. Espic, S. King et al., "A perceptually-motivated low-complexity instantaneous linear channel normalization technique applied to speaker verification," *Computer Speech & Language*, vol. 31, n.º 1, págs. 1-27, 2015.

[2] S. Young, "HMMs and related speech recognition technologies," en *Springer Handbook of Speech Processing*, Springer-Verlag Berlin Heidelberg, 2008, cap. 27, págs. 539-558.

[3] M. Gales, S. Young et al., "The application of hidden Markov models in speech recognition," *Foundations and Trends® in Signal Processing*, vol. 1, n.º 3, págs. 195-304, 2008.

[4] D. Yu y L. Deng, "Introduction," en *Automatic speech recognition, a deep learning approach*, Springer-Verlag London, 2015, cap. 1, págs. 1-9.

[5] L. Rabiner y B.-H. Juang, "Historical perspective of the field of ASR/NLU," en *Springer Handbook of Speech Processing*, Springer-Verlag Berlin Heidelberg, 2008, cap. 27, págs. 521-538.

[6] J. A. Bilmes, "What HMMs can do," *IEICE Transactions on Information and Systems*, vol. 89, n.º 3, págs. 869-891, 2006.

[7] G. Hinton, L. Deng, D. Yu et al., "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, n.º 6, págs. 82-97, 2012.

[8] B. H. Juang y L. R. Rabiner, "Hidden Markov models for speech recognition," *Technometrics*, vol. 33, n.º 3, págs. 251-272, 1991.

[9] G. E. Dahl, D. Yu, L. Deng y A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *IEEE Transactions on audio, speech, and language processing*, vol. 20, n.º 1, págs. 30-42, 2012.

[10] I. Gavat y D. Militaru, "Deep learning in acoustic modeling for automatic speech recognition and understanding-an overview," en *Proceedings of Speech Technology and Human-Computer Dialogue (SpeD)*, IEEE, Bucarest, Rumania, 2015, págs. 1-8.

[11] J. Li, L. Deng, Y. Gong y R. Haeb-Umbach, "An overview of noise-robust automatic speech recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, n.º 4, págs. 745-777, 2014.

[12] D. Yu y L. Deng, "Deep neural network-hidden markov model hybrid systems," en *Automatic Speech Recognition*, Springer-Verlag London, 2015, cap. 6, págs. 99-116.

[13] I. A. McCowan, D. Moore, J. Dines et al., "On the use of information retrieval measures for speech recognition evaluation," IDIAP, inf. téc., 2004.

[14] A. C. Morris, V. Maier y P. Green, "From WER and RIL to MER and WIL: improved evaluation measures for connected speech recognition," en *Proceedings of INTERSPEECH-ICSLP*, ISCA, Isla Jeju, Corea del Sur, 2004.

[15] S. Haykin, "Introduction," en *Neural networks and learning machines*, 3.ª ed., Pearson, 2009, págs. 1-46.
[16] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, n.º 6, págs. 386-408, 1958.
[17] S. Haykin, "Multilayer perceptrons," en *Neural networks and learning machines*, 3.ª ed., Pearson, 2009, cap. 4, págs. 122-229.
[18] I. Goodfellow, Y. Bengio, A. Courville e Y. Bengio, "Autoencoders," en *Deep learning*, MIT Press, 2016, cap. 14, págs. 493-516.
[19] I. Goodfellow, Y. Bengio, A. Courville e Y. Bengio, "Deep feedforward networks," en *Deep learning*, MIT Press, 2016, cap. 6, págs. 163-220.
[20] Y. Bengio, P. Simard y P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, n.º 2, págs. 157-166, 1994.
[21] I. Sutskever, J. Martens, G. Dahl y G. Hinton, "On the importance of initialization and momentum in deep learning," en *International conference on machine learning*, 2013, págs. 1139-1147.
[22] A. Acero y R. M. Stern, "Environmental robustness in automatic speech recognition," en *Proceedings of ICASSP*, IEEE, Albuquerque, NM, USA, 1990, págs. 849-852.
[23] J. Droppo y A. Acero, "Environmental robustness," en *Springer Handbook of Speech Processing*, Springer-Verlag Berlin Heidelberg, 2008, cap. 33, págs. 653-680.
[24] M. L. Seltzer, D. Yu e Y. Wang, "An investigation of deep neural networks for noise robust speech recognition," en *Proceedings of ICASSP*, Vancouver, Canadá, 2013, págs. 7398-7402.
[25] R. Lippmann, E. Martin y D. Paul, "Multi-style training for robust isolated-word speech recognition," en *Proceedings of ICASSP*, IEEE, vol. 12, Dallas, TX, USA, 1987, págs. 705-708.
[26] R. Togneri y D. Pullella, "An overview of speaker identification: accuracy and robustness issues," *Circuits and Systems Magazine, IEEE*, vol. 11, n.º 2, págs. 23-61, 2011.
[27] B. Gold, N. Morgan y D. Ellis, "Feature extraction for ASR," en *Speech and audio signal processing: processing and perception of speech and music*, 2.ª ed., John Wiley & Sons, 2011, cap. 22, págs. 301-318.
[28] A. Mohamed, G. Hinton y G. Penn, "Understanding how deep belief networks perform acoustic modelling," en *Proceedings of ICASSP*, IEEE, Kyoto, Japón, 2012, págs. 4273-4276.
[29] H. Hermansky, "Perceptual linear predictive (PLP) analysis of speech," *the Journal of the Acoustical Society of America*, vol. 87, n.º 4, págs. 1738-1752, 1990.
[30] H. Hermansky, N. Morgan, A. Bayya y P. Kohn, "RASTA-PLP speech analysis technique," en *Proceedings of ICASSP*, IEEE, vol. 1, San Francisco, CA, USA, 1992, págs. 121-124.
[31] H. Hermansky, N. Morgan, A. Bayya y P. Kohn, "Compensation for the effect of the communication channel in auditory-like analysis of speech (RASTA-PLP)," en *Proceedings of EUROSPEECH*, ISCA, Génova, Italia, 1991, págs. 1367-1370.
[32] C. Kim y R. M. Stern, "Feature extraction for robust speech recognition based on maximizing the sharpness of the power distribution and on power flooring," en *Proceedings of ICASSP*, IEEE, 2010, págs. 4574-4577.
[33] C. Kim y R. M. Stern, "Power-normalized cepstral coefficients (PNCC) for robust speech recognition," *IEEE / ACM Transactions on Audio, Speech and Language Processing*, vol. 24, n.º 7, págs. 1315-1329, 2016.
[34] S. Seneff, "A joint synchrony/meanrate model of auditory speech processing," *Journal of Phonetics*, vol. 16, págs. 55-76, 1988.

[35] E. Zwicker y E. Terhardt, "Analytical expressions for critical-band rate and critical bandwidth as a function of frequency," *The Journal of the Acoustical Society of America*, vol. 68, n.º 5, págs. 1523-1525, 1980.

[36] J. Fredes, J. Novoa, V. Poblete, S. King, R. M. Stern y N. B. Yoma, "Robustness to additive noise of locally-normalized cepstral coefficients in speaker verification," en *Proceedings of INTERSPEECH*, Dresden, Alemania, 2015, págs. 3011-3015.

[37] J. H. Hansen y B. D. Womack, "Feature analysis and neural network-based classification of speech under stress," *IEEE Transactions on Speech and Audio Processing*, vol. 4, n.º 4, págs. 307-313, 1996.

[38] V. Poblete, J. P. Escudero, J. Fredes et al., "The Use of Locally Normalized Cepstral Coefficients (LNCC) to Improve Speaker Recognition Accuracy in Highly Reverberant Rooms.," en *Proceedings of INTERSPEECH*, San Francisco, CA, USA, 2016, págs. 2373-2377.

[39] S. Furui, "Speaker-independent isolated word recognition using dynamic features of speech spectrum," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 34, n.º 1, págs. 52-59, 1986.

[40] T. Drugman, Y. Stylianou, L. Chen, X. Chen y M. J. Gales, "Robust excitation-based features for automatic speech recognition," en *Proceedings of ICASSP*, Brisbane, Australia, 2015, págs. 4664-4668.

[41] S. Xue, O. Abdel-Hamid, H. Jiang, L. Dai y Q. Liu, "Fast adaptation of deep neural network based on discriminant codes for speech recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, n.º 12, págs. 1713-1725, 2014.

[42] Y. Miao, H. Zhang y F. Metze, "Distributed learning of multilingual DNN feature extractors using GPUs," en *Proceedings of INTERSPEECH*, Singapur, 2014, págs. 830-834.

[43] N. Parihar y J. Picone, "DSR front end LVCSR evaluation AU/384/02," Institute for Signal and Information Processing, Mississippi State Universitys, inf. téc., dic. de 2002.

[44] G. Hirsch, "Experimental framework for the performance evaluation of speech recognition front-ends on a large vocabulary task, version 2.0, AU/417/02," ETSI STQ Aurora DSR Working Group, inf. téc., dic. de 2002.

[45] P. Woodland, J. Odell, V. Valtchev y S. Young, "Large vocabulary continuous speech recognition using HTK," en *Proceedings of ICASSP*, vol. II, Adelaide, SA, Australia, abr. de 1994, págs. II/125-II/128.

[46] D. B. Paul y J. M. Baker, "The design for the Wall Street Journal-based CSR corpus," en *Proceedings of the Workshop on Speech and Natural Language*, Harriman, NY, USA, 1992, págs. 357-362.

[47] D. Povey, A. Ghoshal, G. Boulianne et al., "The Kaldi speech recognition toolkit," en *Proceedings of ASRU*, EPFL-CONF-192584, Waikoloa, HI, USA, dic. de 2011.

[48] S. Sivasankaran, E. Vincent e I. Illina, "A combined evaluation of established and new approaches for speech recognition in varied reverberation conditions," *Computer Speech & Language*, vol. 46, págs. 444-460, 2017.

[49] K. Veselỳ, A. Ghoshal, L. Burget y D. Povey, "Sequence-discriminative training of deep neural networks.," en *Proceedings of INTERSPEECH*, Lyon, Francia, 2013, págs. 2345-2349.

[50] B. Li y K. C. Sim, "A spectral masking approach to noise-robust speech recognition using deep neural networks," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, n.º 8, págs. 1296-1305, 2014.

[51] Z. Tang, D. Wang y Z. Zhang, "Recurrent neural network training with dark knowledge transfer," en *Proceedings of ICASSP*, Shanghai, China, 2016, págs. 5900-5904.

[52] Y. Qian, M. Bi, T. Tan y K. Yu, "Very deep convolutional neural networks for noise robust speech recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, n.º 12, págs. 2263-2276, 2016.

[53] A. Bayestehtashk, I. Shafran y A. Babaeian, "Robust speech recognition using multivariate copula models," en *Proceedings of ICASSP*, Shanghai, China, 2016, págs. 5890-5894.

[54] H. B. Sailor y H. A. Patil, "Unsupervised deep auditory model using stack of convolutional RBMs for speech recognition," en *Proceedings of INTERSPEECH*, San Francisco, CA, USA, 2016, págs. 3379-3383.

[55] S. Kundu, K. C. Sim y M. J. Gales, "Incorporating a generative front-end layer to deep neural network for noise robust automatic speech recognition," en *Proceedings of INTERSPEECH*, San Francisco, CA, USA, 2016, págs. 2359-2363.

[56] D. S. Pallet, W. M. Fisher y J. G. Fiscus, "Tools for the analysis of benchmark speech recognition tests," en *Proceedings of ICASSP*, Albuquerque, NM, USA, 1990, págs. 97-100.

[57] L. Deng, J. Li, J.-T. Huang et al., "Recent advances in deep learning for speech research at Microsoft," en *Proceedings of ICASSP*, Vancouver, Canadá, 2013, págs. 8604-8608.

[58] C. Weng, D. Yu, M. L. Seltzer y J. Droppo, "Single-channel mixed speech recognition using deep neural networks," en *Proceedings of ICASSP*, Florencia, Italia, 2014, págs. 5632-5636.

[59] D. Yu y L. Deng, "Feature representation learning in deep neural networks," en *Automatic speech recognition, a deep learning approach*, Springer-Verlag London, 2015, cap. 9, págs. 157-175.

[60] C. Plahl, R. Schlüter y H. Ney, "Improved acoustic feature combination for LVCSR by neural networks," en *Proceedings of INTERSPEECH*, Florencia, Italia, 2011.

[61] Z. Tüske, P. Golik, R. Schlüter y H. Ney, "Acoustic modeling with deep neural networks using raw time signal for LVCSR," en *Proceedings of INTERSPEECH*, Singapur, 2014.

[62] D. Yu y L. Deng, "Deep neural networks," en *Automatic speech recognition, a deep learning approach*, Springer-Verlag London, 2015, cap. 4, págs. 57-77.

[63] R. P. Duin y D. M. Tax, "Experiments with classifier combining rules," en *International Workshop on Multiple Classifier Systems*, Cagliari, Italia, 2000, págs. 16-29.

[64] X. Li y R. M. Stern, "Training of stream weights for the decoding of speech using parallel feature streams," en *Proceedings of ICASSP*, IEEE, vol. 1, Hong Kong, China, 2003, págs. I-832-I-835.

[65] F. Huenupán, N. B. Yoma, C. Garretón y C. Molina, "On-Line Linear Combination of Classifiers Based on Incremental Information in Speaker Verification," *ETRI Journal*, vol. 32, n.º 3, págs. 395-405, 2010.

[66] S. H. Mallidi, T. Ogawa, K. Veselỳ, P. S. Nidadavolu y H. Hermansky, "Autoencoder based multi-stream combination for noise robust speech recognition," en *Proceedings of INTERSPEECH*, Dresden, Alemania, 2015.

[67] F. Xiong, S. Goetze y B. T. Meyer, "On DNN posterior probability combination in multi-stream speech recognition for reverberant environments," en *Proceedings of ICASSP*, New Orleans, LA, USA, 2017, págs. 5250-5254.

[68] J. E. Novoa Ilic, "Robust speech recognition in noisy and reverberant environments using deep neural network-based systems," Tesis doct., Universidad de Chile, 2018.

[69] J. P. Escudero, V. Poblete, J. Novoa et al., "Highly-Reverberant Real Environment database: HRRE," *ArXiv e-prints*, ene. de 2018. arXiv: 1801.09651.

[70] E. A. Habets, “Room impulse response generator,” Technische Universiteit Eindhoven, inf. téc., sep. de 2006.

[71] J. B. Allen y D. A. Berkley, “Image method for efficiently simulating small-room acoustics,” *The Journal of the Acoustical Society of America*, vol. 65, n.º 4, págs. 943-950, 1979.

[72] V. Peddinti, D. Povey y S. Khudanpur, “A time delay neural network architecture for efficient modeling of long temporal contexts,” en *Proceedings of INTERSPEECH*, ISCA, Dresden, Alemania, 2015, págs. 3214-3218.

Anexo

Publicaciones realizadas en el marco de la tesis.

Robustness to Additive Noise of Locally-Normalized Cepstral Coefficients in Speaker Verification

Josué Fredes[1], José Novoa[1], Victor Poblete[1, 2], Simon King[3], Richard Stern[4], Nestor Becerra Yoma[1]

[1] Speech Processing and Transmission Laboratory, Electrical Engineering Department, University of Chile, Santiago, Chile
[2] Institute of Acoustics, Universidad Austral de Chile, Valdivia, Chile
[3] Centre for Speech Technology Research, University of Edinburgh, Edinburgh, UK
[4] Department of Electrical and Computer Engineering and Language Technologies Institute, Carnegie Mellon University, Pittsburgh, PA 15206, USA

jfredes@ing.uchile.cl, jose.novoa@ing.uchile.cl, vpoblete@uach.cl, Simon.King@ed.ac.uk, rms@cs.cmu.edu, nbecerra@ing.uchile.cl

10.21437/Interspeech.2015-98

Abstract

In this paper the performance of a new feature set, Locally Normalized Cepstral Coefficients (LNCC) is evaluated for a speaker verification task with short testing utterances in additive noise. The results presented here show that LNCC outperforms baseline MFCC features when SNR is lower than 15 dB. The average relative reduction in EER achieved by LNCC is 33%. The use of LNCC in combination with spectral subtraction provides a reduction in EER averaging 18% when compared to MFCC features also with spectral subtraction. In addition, sub-band LNCC is proposed to improve the estimation of noise energy and hence the effectiveness of spectral subtraction. When compared with MFCC features, the use of sub-band LNCC led to greater reductions in EER than LNCC with non-stationary noise.

Index Terms: LNCC, additive noise, speaker verification.

1. Introduction

One of the main problems faced by speech technologies in real applications is noise robustness. Noise introduces a mismatch between training and testing conditions in the features representing a signal. In particular, additive noise generates a distortion that has a greater effect in those intervals with low segmental SNR. Additionally, additive noise causes loss of information because the original information cannot be recovered with 100% accuracy [1] [2].

Many techniques have been proposed to improve robustness to additive noise. A group of these techniques attempt to compute a statistical estimate of noise, and to compensate the noisy signal or to adapt acoustic models to noisy conditions. The compensation methods include, for example, spectral subtraction, Wiener filtering and Kalman filtering [3] [4] [5] [6] and the Stochastic Weighted Viterbi algorithm that estimates and makes use of the reliability in noise cancelling [2]. The adaptation methods include statistical re-estimation and parallel model combination [7] [8] [9]. There is also another group of techniques that attempts to introduce features that are more robust to additive noise [10] [11]; LNCC belongs to this group.

Locally-Normalized Cepstral Coefficients (LNCC) are a novel set of speech features introduced in [12] that are perceptually-motivated and an extremely simple but effective way to instantaneously normalize speech features. The effectiveness of the proposed features has been demonstrated for a speaker verification task across a wide variety of linear channel conditions. In this paper we evaluate the robustness to additive noise of LNCC in a speaker verification task. It is worth highlighting that the task addressed here involves short testing utterances (*i.e.*, shorter than five seconds), as required in typical commercial applications, and is very different from the Speaker Recognition Evaluations (SRE) organized by NIST in recent years. In NIST evaluations several minutes of speech samples for verification or identification are available. In this context, as we see in the literature, some of the techniques proposed for long utterances (e.g., JFA and i-Vectors) are simply not appropriate for short utterances. Results presented here suggest that LNCC provides an average error reduction of 33% when SNR is lower than 15 dB, when compared with ordinary MFCC. In combination with spectral subtraction (SS) the difference between LNCC and MFCC processing is reduced but LNCCs still provides an EER that is 18% lower on average at moderate and low SNRs.

2. LNCC and additive noise

LNCC are a new set of cepstral-type features, inspired by Seneff's Generalized Synchrony Detector (GSD). These features perform a local normalization in the frequency domain in each auditory channel, being relatively invariant to changes in the transmission channel frequency response. This normalization is achieved by dividing the energy of two types of filters. The numerator filter is defined as a triangular filter, similar to that used in the typical Mel filterbank, while the denominator filter captures energy from either side of this. The numerator and denominator filters are defined in Eq. (1) and Eq. (2) respectively, for a channel j with central frequency f_j^C and bandwidth B.

$$Num_j(f) = \begin{cases} -\frac{2}{B}\left|f - f_j^C\right| & , when \quad \left|f - f_j^C\right| \le \frac{B}{2} \\ 0 & , otherwise \end{cases} \quad (1)$$

$$Den_j(f) = \begin{cases} \frac{2}{B}(1-d_{min})\left|f - f_j^C\right| + d_{min} & , when \quad \left|f - f_j^C\right| \le \frac{B}{2} \\ 0 & , otherwise \end{cases} \quad (2)$$

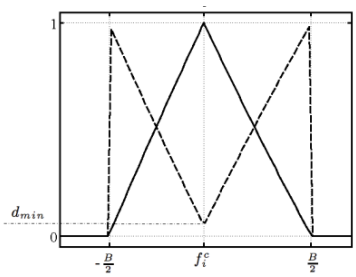


Figure 1: *Frequency response of numerator filter (solid line) and denominator filter (dashed line).*

Given a channel j, the locally-normalized channel energy LN_j is defined as:

$$LN_j = \frac{Num\_Energy_j}{Den\_Energy_j} \quad (3)$$

where Num_Energy_j and Den_Energy_j are the energies of the filters defined in Eq. (1) and Eq. (2). LNCC has already been tested on spectral tilt transmission channel conditions [12], achieving better performance than ordinary MFCC. In some conditions, LNCC can even outperform MFCC+CMN and MFCC+RASTA. Nevertheless, the robustness of LNCC to additive noise has not yet been assessed.

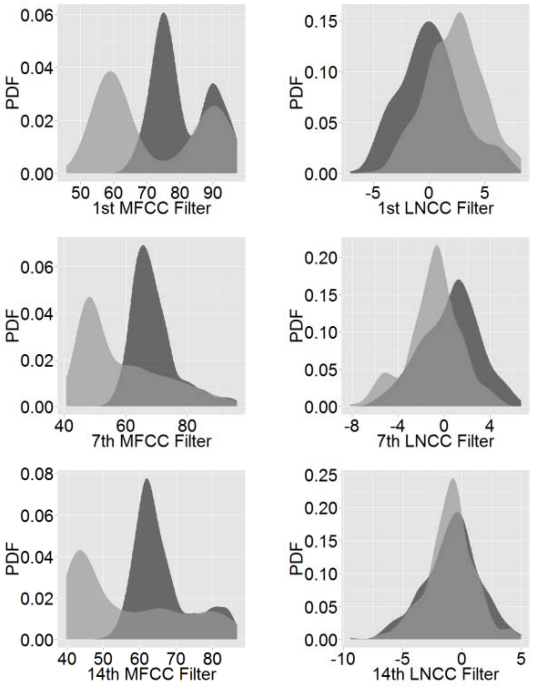


Figure 2: *Probability density functions of LNCC and MFCC filter energy at channels 1, 7 and 14. Lighter gray represents the clean signal and darker gray represents the noisy signal.*

Given that $s(i)$, $n(i)$ and $x(i)$ are the clean speech, the noise and the resulting noisy signal, respectively, the additiveness condition in the temporal domain is expressed as:

$$x(i)=s(i)+n(i) \quad (4)$$

where i is the discrete time index. Initially, additive babble noise was artificially added to clean speech signals at an SNR of 10 dB. The probability density functions (pdfs) for energies of three channels from the ordinary Mel filterbank and the normalized filters from LNCC were estimated under clean and noisy conditions. Fig. 2 shows the results for the 1st, 7th and 14th filter.

As can be seen in Fig. 2, the perturbation due to additive noise on the pdf for the normalized LNCC filters is less pronounced than the perturbation observed for the Mel Filter energies. The estimated pdfs for clean and noisy signals with LNCC's normalized filters are visibly more similar than for the MFCC filter energy in all the three filters. The Kolmogorov-Smirnov Statistic [13] between the clean and noisy energy pdfs are shown in Table 1. According to Table 1, the difference between clean and noisy pdfs is 37% to 82% lower with LNCC than with MFCC. The results shown in Table 1 were obtained using the statistical software R [14] and 'dgof' package [15].

	Kolmogorov-Smirnov Statistic	
	MFCC	LNCC
1st filter	0.546	0.346
7th filter	0.578	0.330
14th filter	0.562	0.098

Table 1: *Kolmogorov-Smirnov Statistic between the clean and noisy energy pdfs.*

The effect of additive noise was also examined at the frame level. Fig. 3 shows the energy of all channels for a given voiced frame with MFCC and LNCC. As we see in Fig. 3, the difference between clean and noisy speech with LNCC is lower than with MFCC.

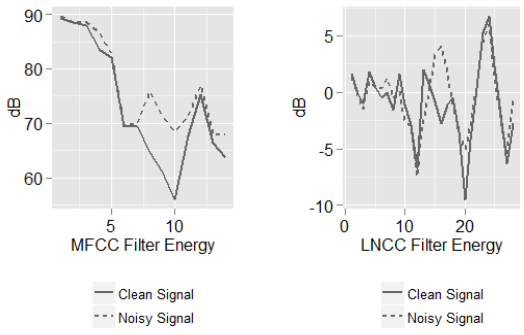


Figure 3: *LNCC and MFCC energy in a voiced frame from clean and noisy utterance at SNR equal 10 dB.*

The differences observed between clean and noisy signals in Figs. 2 and 3 represent the distortion introduced by the additive noise. This distortion leads to mismatch between training and testing conditions that negatively affects the speaker and speech recognition accuracy. The results presented

in this section suggest that LNCCs should be more robust to additive noise than MFCCs.

3. LNCC and spectral subtraction

Spectral subtraction (SS) is a widely employed compensation technique for additive noise [16]. It is based on the subtraction of an estimation of the additive noise energy spectrum from the noisy signal power energy. If $E[\overline{n_j}]$ is the estimated noise energy for channel j, and FE_j is the corresponding filter energy, then the compensated filter energy obtained with SS, FE_j^{SS}, can be defined as [16]:

$$FE_j^{SS} = max\{FE_j - \alpha(SNR_j) \times E[\overline{n_j}] \; ; \; \beta \times FE_j\} \qquad (5)$$

where $\alpha(SNR_j)$ controls the amount of noise subtracted from the noisy signal and depends on the channel SNR; and β defines a lower bound to the energy of the compensated filter energy to avoid negative values. The applicability of (5) to Mel filter banks is straightforward. However, in the case of LNCC, SS must be applied to the numerator and denominator energies separately. Then, the compensated *LN* energy for channel j is written as:

$$LN_j^{SS} = \frac{Num\_Energy_j^{SS}}{Den\_Energy_j^{SS}} \qquad (6)$$

Due to the denominator filter shape in Fig. 1, a more accurate estimate of noise can be achieved if the denominator filter is split into two sub-bands: right and left denominator sub-bands. Then, using the definition in (2), the right and left denominator sub-band filters in channel j are defined as:

$$Den\ Left\ SB_j(f) = \begin{cases} Den_j(f), \ when \ f_j^C - f \le \frac{B}{2} \\ 0, \ otherwise \end{cases} \qquad (7)$$

$$Den\ Right\ SB_j(f) = \begin{cases} Den_j(f), \ when \ f - f_j^C \le \frac{B}{2} \\ 0, \ otherwise \end{cases} \qquad (8)$$

The sum of the right and left denominator sub-band energies is equal to the denominator filter energy defined in (2). Consequently, the compensated *LN* energy obtained by applying SS on a sub-band basis in channel j, $SB\text{-}LN_j^{SS}$, can be expressed as:

$$SB\text{-}LN_j^{SS} = \frac{Num\_Energy_j^{SS}}{LDen\_Energy_j^{SS} + RDen\_Energy_j^{SS}} \qquad (9)$$

The compensated LNCC features obtained with (9) are denoted by SB-LNCC. It is worth mentioning that SS compensation according to (9) would make use of a more accurate estimation of noise.

4. Experiments

LNCC performance in additive noise environments was tested with text-independent speaker verification experiments. All experiments used the entire YOHO Speaker Verification Corpus [17]. Three types of noise (babble, car, and restaurant) were selected from the AURORA database [18], which were artificially added to the YOHO corpus to generate noisy versions of test speech signals at SNR equal to 0 dB, 5 dB, 10 dB, 15 dB and 20 dB. As mentioned above, the task addressed in this paper is completely different from those in the NIST challenges, where several minutes of speech samples are available for verification or identification. In contrast, we employ only two or three second utterances as required by many commercial applications As already mentioned, many techniques proposed for long utterances do not work well for short utterances. The baseline system provides EER equal to 0.71% and 0.56% with LNCC and MFCC, respectively. The latter result is competitive with (and actually lower than) EERs found in the literature [19] [20] [21] [22].

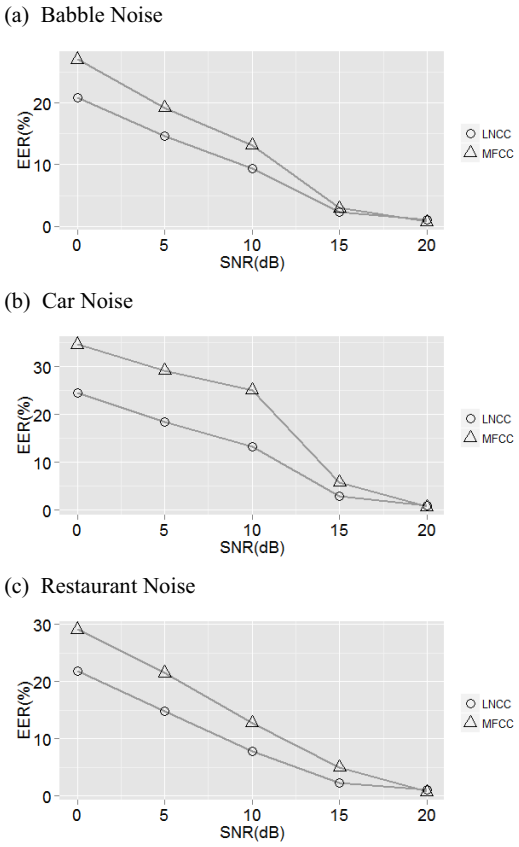


Figure 4: *EER with LNCC and MFCC without SS.*

The experiments were carried out with the ALIZE library [23] and LIA-SpkDet toolkit [23], state-of-the-art open source software for speaker recognition. The GMM-UBM system (Gaussian Mixture Model – Universal Background Model) was used. ALIZE supports newer speaker verifications approaches such as JFA and i-Vectors, but these methods do not perform well when short speech utterances (shorter than five sec) are available, as is shown in [12] [24] [25]. For instance, the baseline system with JFA and i-Vectors give EER equal to 3.6% and 2.1%, respectively, with MFCC and clean speech.

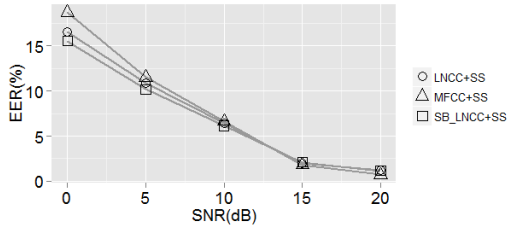
Fig. 4 shows the EERs obtained for babble, car and restaurant noise without SS. In the experiments with SS

parameters α and β in (5) where tuned with babble noise at an SNR of 10 dB with MFCC, LNCC, and SB-LNCC. The optimum α and β were kept constant in the experiments. Fig. 5 presents the results with SS.

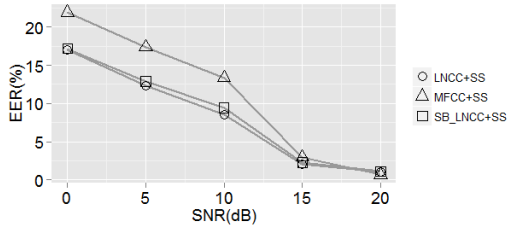
5. Discussion

As seen in Fig. 4, LNCC leads to substantial improvements in EER at SNR equal to 0 dB, 5 dB, 10 dB and 15 dB when compared with MFCC. LNCC provides relative reductions in EER between 19.3% and 28.6% for babble noise, between 29.2% and 49.2% for restaurant noise, and between 25.0% and 54.9% for car noise when SNR is lower than 15 dB. At SNR equal to 20 dB, LNCC shows an increase in EER with respect to MFCC: 0.31% absolute with babble noise; 0.28% absolute with restaurant noise; and, 0.30% absolute for car noise. These results are consistent with the analysis presented in Section 2.

(a) Babble Noise



(b) Car Noise



(c) Restaurant Noise

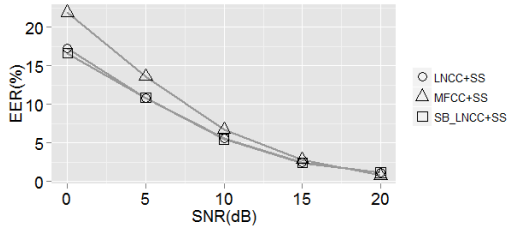


Figure 5: *EER with LNCC, SB-LNCC and MFCC with SS.*

In Fig. 5, we observe that the application of SS improves both LNCC and MFCC performance. Despite the fact that the difference between LNCC and MFCC is reduced, the trend observed in Fig. 4 is preserved in Fig. 5: at SNRs between 0 dB and 15 dB, LNCC outperforms MFCC. The relative reductions in EER obtained with LNCC are between 3.6% and 11.5% for babble noise, between 22.5% and 35.9% for car noise, and between 11.2% and 21.2% for restaurant noise. There is a single exception with babble noise at SNR equal to 15 dB. In this case the absolute difference between LNCC and MFCC is just 0.14%. As in Fig. 4, LNCC with SS also shows an increase in EER when compared with MFCC at 20 dB SNR: 0.48% absolute with babble noise; 0.36% absolute with car noise; and, 0.38% absolute for restaurant noise.

The sub-band compensated feature according to (9), SB-LNCC, presents a similar behavior as LNCC. In the range of 0 dB to 15 dB SNR, SB-LNCC outperforms MFCC with relative improvements in EER between 7.4% to 16.9% for babble noise, between 21.7% to 29.4% for car noise, and between 14.83% to 24.06% for restaurant noise. The same exception is observed with babble noise at SNR equal to 15 dB where the absolute difference in EER is equal to 0.29% absolute when compared with MFCC. SB-LNCC also shows an increase in EER at SNR equal to 20 dB which is 0.43% absolute for babble noise, 0.46% absolute for car noise, and 0.38% absolute for restaurant noise

SB-LNCC was expected to outperform LNCC in additive noise environments. This is indeed the case with babble noise and restaurant noise at several SNRs. However, this is not observed with car noise. Observe that the babble and restaurant additive noise are less stationary than car noise, which is largely stationary [18]. This suggests that SB-LNCC introduces improvements in the system performance when the degrading additive noise is non-stationary, implying that the division into sub-bands proposed in (7) and (8) yields to a better estimation of the noise energy with non-stationary signals.

6. Conclusions

In this paper LNCC is tested with additive corrupting processes by using babble, restaurant and car noise. LNCC outperformed MFCC with these three types of noise when SNR is between 0 dB to 15 dB. The average relative reduction in EER achieved by LNCC was equal to 23.6% for babble noise, 37.3% for car noise and 40.6% for restaurant noise. By using spectral subtraction in both LNCC and MFCC, the LNCC reductions in EER was in average 6.9% for babble noise, 17.2% for restaurant noise, and 29.1% for car noise. The highest improvement takes place when the noise is stationary (*i.e.* car). Sub-band LNCC was proposed to improve the noise energy estimation and hence the SS effectiveness. When compared with MFCC, Sub-band LNCC led to higher reductions in EER than LNCC with no-stationary noise: 11.94% and 19.38% for babble and restaurant noise, respectively. Improving the applicability of LNCC and its evaluation on ASR tasks are proposed for future research.

7. Acknowledgements

The research leading to these results was funded by CONICYT-ANILLO project ACT 1120, CONICYT-FONDECYT project 1151306 and ONR grants N62909-12-1-7095 and N62909-12-1-7118. S. King was partly funded by EPSRC grant EP/I031022/1 (Natural Speech Technology). Richard Stern was partially funded by the Defense Advanced Research Projects Agency (DARPA) under Contract No.D10PC20024. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the view of DARPA or its Contracting Agent, the U.S. Department of the Interior, National Business Center, Acquisition & Property Management Division, Southwest Branch. V. Poblete was partly funded by DID-UACh 2015-63.

8. References

[1] A. De La Torre, A. M. Peinado, J. C. Segura, J. L. Pérez-Córdoba, M. C. Benítez and A. J. Rubio, "Histogram equalization of speech representation for robust speech recognition," *IEEE Transactions on Speech and Audio Processing,* vol. 13, no. 3, pp. 355-366, 2005.

[2] N. B. Yoma and M. Villar, "Speaker verification in noise using a stochastic version of the weighted Viterbi algorithm," *Speech and Audio Processing, IEEE Transactions on,* vol. 10, no. 3, pp. 158-166, 2002.

[3] C. Cole, M. Karam and H. Aglan, "Increasing Additive Noise Removal in Speech Processing Using Spectral Subtraction," in *Information Technology: New Generations, 2008. ITNG 2008. Fifth International Conference on*, 2008.

[4] A. Panda, N. Tripathi and T. Srikanthan, "Improved spectral subtraction technique for text-independent speaker verification," in *Digital Signal Processing, 2007 15th International Conference on*, 2007.

[5] M. I. Mandasari, M. McLaren and D. A. van Leeuwen, "The effect of noise on modern automatic speaker recognition systems," in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, 2012.

[6] S. Stan, T. Fingscheidt and C. Beaugeant, "An evaluation of VTS and IMM for speaker verification in noise," in *Eighth European Conference on Speech Communication and Technology*, 2003.

[7] M. J. Gales and S. J. Young, "Robust continuous speech recognition using parallel model combination," *Speech and Audio Processing, IEEE Transactions on,* vol. 4, no. 5, pp. 352-359, 1996.

[8] J. C. Segura, Á. de la Torre, M. C. Benitez and A. M. Peinado, "Model-based compensation of the additive noise for continuous speech recognition. experiments using the Aurora II database and tasks.," in *INTERSPEECH*, 2001.

[9] X. Zhao and Z. Ou, "Closely coupled array processing and model-based compensation for microphone array speech recognition," *Audio, Speech, and Language Processing, IEEE Transactions on,* vol. 15, no. 3, pp. 1114-1122, 2007.

[10] R. Saeidi, J. Pohjalainen, T. Kinnunen and P. Alku, "Temporally weighted linear prediction features for tackling additive noise in speaker verification," *Signal Processing Letters, IEEE,* vol. 17, no. 6, pp. 599-602, 2010.

[11] A. Panda, "Psychoacoustic model compensation with robust feature set for speaker verification in additive noise," in *Chinese Spoken Language Processing (ISCSLP), 2014 9th International Symposium on*, 2014.

[12] V. Poblete, F. Espic, S. King, R. M. Stern, F. Huenupán, J. Fredes and N. B. Yoma, "A perceptually-motivated low-complexity instantaneous linear channel normalization technique applied to speaker verification," *Computer Speech & Language,* vol. 31, no. 1, pp. 1-27, 2015.

[13] A. Ivanov and G. Riccardi, "Kolmogorov-Smirnov test for feature selection in emotion recognition from speech," in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, 2012.

[14] R Core Team, "R: A Language and Environment for Statistical Computing," Vienna, Austria, 2015.

[15] T. B. Arnold and J. W. Emerson, "Nonparametric Goodness-of-Fit Tests for Discrete Null Distributions," *The R Journal,* vol. 3, no. 2, pp. 34-39, 2011.

[16] S. V. Vaseghi and B. P. Milner, "Noise compensation methods for hidden Markov model speech recognition in adverse environments," *Speech and Audio Processing, IEEE Transactions on,* vol. 5, no. 1, pp. 11-21, 1997.

[17] J. Campbell and A. Higgins, "YOHO speaker verification," *Linguistic Data Consortium, Philadelphia,* 1994.

[18] D. Pearce, H.-G. Hirsch and others, "The aurora experimental framework for the performance evaluation of speech recognition systems under noisy conditions," in *INTERSPEECH*, 2000.

[19] S. Chakrabartty and S.-C. Liu, "Exploiting spike-based dynamics in a silicon cochlea for speaker identification," in *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, 2010.

[20] M.-S. Kim, I.-H. Yang and H.-J. Yu, "Maximizing Distance between GMMs for Speaker Verification," in *Natural Computation, 2008. ICNC '08. Fourth International Conference on*, 2008.

[21] S. Memon, M. Lech and L. He, "Using information theoretic vector quantization for inverted MFCC based speaker verification," in *Computer, Control and Communication, 2009. IC4 2009. 2nd International Conference on*, 2009.

[22] Z. Lei, Y. Yang and Z. Wu, "An UBM-Based Reference Space for Speaker Recognition," in *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, 2006.

[23] J.-F. Bonastre, N. Scheffer, D. Matrouf, C. Fredouille, A. Larcher, A. Preti, G. Pouchoulin, N. W. Evans, B. G. Fauve and J. S. Mason, "ALIZE/spkdet: a state-of-the-art open source software for speaker recognition," in *Odyssey*, 2008.

[24] T. Kinnunen and H. Li, "An overview of text-independent speaker recognition: From features to supervectors," *Speech communication,* vol. 52, no. 1, pp. 12-40, 2010.

[25] C. Hanilçi and F. Ertaş, "Investigation of the effect of data duration and speaker gender on text-independent speaker recognition," *Computers \& Electrical Engineering,* vol. 39, no. 2, pp. 441-452, 2013.

Locally Normalized Filter Banks Applied to Deep Neural-Network-Based Robust Speech Recognition

Josué Fredes, José Novoa, Simon King, Richard M. Stern, *Fellow, IEEE*, and Nestor Becerra Yoma, *Member, IEEE*

***Abstract*—This letter describes modifications to locally normalized filter banks (LNFB), which substantially improve their performance on the Aurora-4 robust speech recognition task using a Deep Neural Network-Hidden Markov Model (DNN-HMM)-based speech recognition system. The modified coefficients, referred to as LNFB features, are a filter-bank version of locally normalized cepstral coefficients (LNCC), which have been described previously. The ability of the LNFB features is enhanced through the use of newly proposed dynamic versions of them, which are developed using an approach that differs somewhat from the traditional development of delta and delta–delta features. Further enhancements are obtained through the use of mean normalization and mean–variance normalization, which is evaluated both on a per-speaker and a per-utterance basis. The best performing feature combination (typically LNFB combined with LNFB delta and delta–delta features and mean–variance normalization) provides an average relative reduction in word error rate of 11.4% and 9.4%, respectively, compared to comparable features derived from Mel filter banks when clean and multinoise training are used for the Aurora-4 evaluation. The results presented here suggest that the proposed technique is more robust to channel mismatches between training and testing data than MFCC-derived features and is more effective in dealing with channel diversity.**

***Index Terms*—Automatic speech recognition (ASR), Aurora-4, channel mismatch, deep neural network (DNN), locally normalized filter bank (LNFB).**

I. Introduction

THE use of deep neural networks (DNNs) has produced major improvements in the recognition accuracy of automatic speech recognition (ASR) systems. DNNs have the ability to learn internal features, which are robust to many sources of variability in speech signals (e.g., [1] and [2]). In this context, simple features like log-Mel filter banks (MelFB) favor the DNN learning process and provide greater recognition accuracy than traditional MFCC features [3], [4]. Nevertheless, when mismatches between training and testing conditions are too large, the learning ability of DNNs is limited and the recognition accuracy degrades significantly [1]. Well-known techniques for reducing the train-test mismatch include the application of input normalization such as mean normalization (MN) and mean and variance normalization (MVN). MN assumes that the mean of data is invariant, and MVN uses the stronger assumption that the mean and variance of data are invariant, so standardizing mean and/or variance removes irrelevant information [5]. These techniques reduce the mismatch between features representing clean and noisy utterances [6], [7], and they reduce distortion introduced by linear filtering or spectral tilt [5]. MN and MVN are also used for DNN input normalization [8] because DNN training is sensitive to input scale [9]. Because un-normalized features with greater variance would dominate the DNN learning process, scaling each dimension of the input data to a similar range improves DNN performance [8]. Another effective approach for reducing training–testing mismatches is to employ multicondition training with noisy or distorted data, so noise is present in both the training and testing data. However, this strategy is not practical in some applications where the channel between the speaker and the ASR system may vary over time. Examples of such applications include human–robot interaction, meeting transcription, lecture transcription, etc.

Locally normalized cepstral coefficients (LNCC) [10] were designed to be robust to channel mismatches, and they provided better accuracy than traditional MFCC features on a speaker verification task with mismatches between training and testing conditions [10], [11]. The development of locally normalized filter bank (LNFB) features is motivated by the observations that the performance of DNN-HMM ASR systems is typically better when spectrogram-like features (such as MelFB parameters) are used, rather than features in a pseudo-cepstral domain (such as MFCC parameters), despite the fact that MFCC coefficients are simply the discrete cosine transform (DCT) of the MelFB parameters. Similarly, LNCC features are the DCT of LNFB features.

In this letter we consider the use of LNFB parameters for the Aurora-4 ASR task. The sections below discuss the application of LNFB features to robust DNN-HMM-based ASR, the development of LNFB delta and delta–delta coefficients, the combination of static and dynamic LNFB parameters, and an analysis and comparison of results obtained using speaker-based and utterance-based based MN and MVN input normalization.

Manuscript received June 30, 2016; revised November 4, 2016; accepted January 14, 2017. Date of publication January 31, 2017; date of current version February 24, 2017. This work was supported by CONICYT-FONDECYT under Project 1151306 and CONICYT-ANILLO under Project ACT 1120. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Peter K. Willett.

J. Fredes, J. Novoa, and N. B. Yoma are with the Speech Processing and Transmission Laboratory, Electrical Engineering Department, University of Chile, Santiago, Chile (e-mail: jfredes@ing.uchile.cl; jose.novoa@ing.uchile.cl; nbecerra@ing.uchile.cl).

S. King is with the Centre for Speech Technology Research, University of Edinburgh, Edinburgh, EH8 9YL, U.K. (e-mail: Simon.King@ed.ac.uk).

R. M. Stern is with the Department of Electrical and Computer Engineering and Language Technologies Institute, Carnegie Mellon University, Pittsburgh, PA 15213 USA (e-mail: rms@cs.cmu.edu).

Digital Object Identifier 10.1109/LSP.2017.2661699

II. LNFB Coefficients and Delta Coefficients

A. LNFB Features

LNCCs are a set of cepstral-type features, inspired by Seneff's generalized synchrony detector (GSD) [12], which

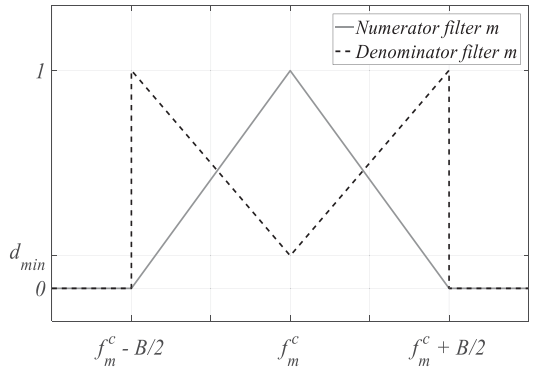


Fig. 1. Graphical representation of the *m*th numerator filter (solid line) and the *m*th denominator filter (dashed line). The frequency axis is in the Bark scale [13] as used in [10].

perform a local normalization in the frequency domain in each auditory channel, and hence are relatively invariant to changes in the frequency response of the transmission channel [10]. Accordingly, LNFB features are LNCC features before the final DCT computation. The local normalization is achieved in the filter-bank space by dividing the output of a triangular frequency-weighting filter (which is similar to the triangular filter in conventional MFCC coefficients) by the output of a second frequency-weighting filter [10]. This normalization removes very coarse variations in the spectral shape that can be considered constant within both filters, such as overall tilt, which we assume arise mostly from channel variability. We refer to these two filters as the "numerator filter" and the "denominator filter," and their shape is an approximation to the frequency response of the numerator and denominator of the Seneff GSD operator

$$\mathrm{Num}_m\,(f) = \begin{cases} -\frac{2}{B}\left|f - f_m^C\right|, & \text{if } \left|f - f_m^C\right| \le \frac{B}{2} \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

$$\mathrm{Den}_m\,(f) = \begin{cases} \frac{2}{B}\left(1 - d_{\min}\right)\left|f - f_m^C\right| + d_{\min}, & \text{if } \left|f - f_m^C\right| \le \frac{B}{2} \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

where the frequency variable *f* is in the Bark scale [13]. The shapes of these filters are shown in Fig. 1. Given a channel *m* with center frequency f_m^C and bandwidth *B*, the LNFB feature *m* is defined as the log of the locally normalized energy for channel *m*, LN_m

$$\mathrm{LNFB}_m = \log\left(\mathrm{LN}_m\right) = \log(\mathrm{LNNum}_m / \mathrm{LNDen}_m) \quad (3)$$

where LNNum_m is the numerator filter energy and LNDen_m is the denominator filter energy.

B. Dynamic Features with LNFB

Delta and delta–delta features [14] represent the most common method for capturing the temporal evolution of the short-term spectrum or cepstrum [15]. Delta and delta–delta dynamic features are computed as polynomial approximations of the first- and second-order time derivatives of the static features. If $C_m(n)$ is the static feature *m* at discrete time *n*, the delta features can be expressed as [14]

$$\Delta C_m(n) = \frac{\sum_{k=-K}^{K} k C_m(n+k)}{\sum_{k=-K}^{K} k^2} \quad (4)$$

where *2K + 1* frames centered around frame *n* are used to compute the numeric time derivative at frame *n*. The delta–delta coefficients are obtained by repeating (4) using the delta coefficients as input. Delta and delta–delta features represent the dynamic characteristics of the speech spectra over time, and they are usually employed in combination with static coefficients such as MFCC, LPC, PLP, MelFB, etc.

C. Deltas Delta–Deltas for LNFB Features

Direct application of (4) to the LNFB features in (3) would produce dynamic features according to the following equations:

$$\Delta\mathrm{LNFB1}_m(n) = \Delta\log(\mathrm{LNNum}_m(n)) - \Delta\log(\mathrm{LNDen}_m(n)) \quad (5)$$

$$\Delta\Delta\mathrm{LNFB1}_m(n) = \Delta\Delta\log(\mathrm{LNNum}_m(n)) - \Delta\Delta\log(\mathrm{LNDen}_m(n)). \quad (6)$$

Because LNFB features have already been normalized, we believe that computing delta and delta–delta LNFB coefficients using (5) and (6), as mentioned earlier, would not represent spectral evolution properly, because independent delta operations would be applied to both the numerator and denominator filter of (1) and (2). It is not clear what the result of such an operation might represent, but it would not be consistent with the original motivation of dynamic features [14], [15]. Instead, we believe that computation of dynamic versions of the LNFB features should be accomplished by applying the linear regression of [14] to the numerator filter only, using the following equations:

$$\Delta\mathrm{LNFB2}_m(n) = \Delta\log(\mathrm{LNNum}_m(n)) \quad (7)$$

$$\Delta\Delta\mathrm{LNFB2}_m(n) = \Delta\Delta\log(\mathrm{LNNum}_m(n)). \quad (8)$$

The superiority of this approach to computing the △LNFB features is confirmed by experimental results described later. This strategy could be generalized easily to other self-normalizing features.

III. Data Normalization in DNN-HMM-Based ASR

MN and MVN are widely used to achieve robust ASR. In applying these approaches to a DNN-HMM based system, the means and/or variances could be evaluated over the training set only or over both the training and testing data, and on a per-speaker or per-utterance basis (e.g., [3], [4], [16], and [17]). The optimal normalization could be dependent on the task and on the degree of mismatch between training and testing conditions due to the combined effect of these normalizations (acoustic compensation and scale normalization). In this letter, four input normalization schemes are considered: MN and MVN applied on a per-speaker and on a per-utterance basis, normalizing both the training and testing data in all cases.

Fig. 2 compares the amplitude distributions of MelFB and LNFB coefficients with no normalization, with MN, and with MVN. The plots compare the normalized histograms for filters with center frequencies of approximately 470 Hz using the

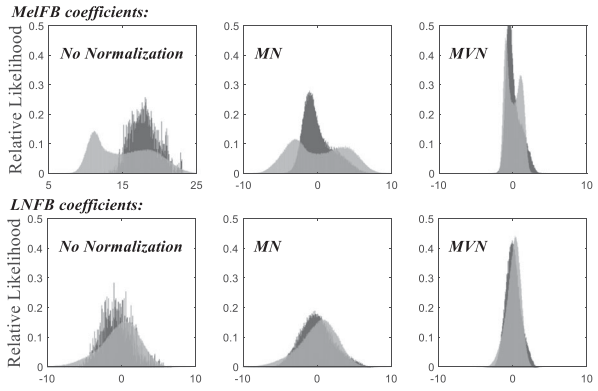


Fig. 2. Histograms for filters with center frequencies of approximately 470 Hz, from the Aurora-4 clean train set (in lighter gray) and the Aurora-4 Group D test subset (in darker gray), for (upper row) MelFB and (lower row) LNFB features. No normalization (left column), mean normalization (center column), and MVN (right column) were applied to training and test data on a per-speaker basis.

Aurora-4 clean train set and the Aurora-4 Group D degraded test set (as defined later), which includes both additive noise and channel distortion. The MN and MVN normalizations were applied on an utterance-by-utterance basis. As can be seen in Fig. 2, the clean and noisy histograms of MelFB coefficients are bimodal or multimodal while those of the LNFB coefficients are unimodal. After applying MN or MVN, the clean and noisy histograms remain clearly distinguishable from each other using MelFB features, despite the fact that they are scaled to similar ranges of variation. In contrast, the clean and noisy histograms from LNFB coefficients are unimodal and very similar after applying MN or MVN. While not shown in this letter, this behavior is observed for the vast majority of frequency channels. This result suggests that LNFB should be more robust than MelFB to additive noise and channel distortion and that MN and MVN should be more effective for the LNFB coefficients. We believe that the unimodal nature of the LNFB histograms is a consequence of the normalization of the numerator by the denominator filter energy according to (3).

IV. Experiments

The effectiveness of the LNFB features was validated by performing ASR experiments on the Aurora-4 corpus [18] using the Kaldi Speech Recognition Toolkit [19]. Three training sets from Aurora-4 were employed: the clean, multinoise and multiconditions. Each training set contains 7137 utterances from 83 speakers. The clean training set contains only clean data recorded with the high-quality Sennheiser HMD-414 microphone. The multinoise set contains clean and artificially degraded utterances with 6 different noises added at SNRs between 10 and 20 dB. All data in this training set were recorded with the Sennheiser HMD-414 microphone. Finally, half the multicondition training set was recorded by the Sennheiser HMD-414 microphone, while the other half was recorded by one out of 18 different microphones, with noise added as in the multinoise data. The evaluation database is composed of 14 testing sets with 330 utterances each, clustered in four groups, as summarized in Table I [18].

Three sets of features were evaluated: static and dynamic Mel Filterbank features labeled as MelFB+ΔMelFB; static and dynamic LNFB parameters, using (5) and (6), labeled as LNFB+ΔLNFB1; and static LNFB features combined with dynamic LNFB parameters based on the numerator filters only, using (7) and (8), labeled as LNFB+ΔLNFB2. Note that for the rest of this letter, we use the labels $\triangle$MelFB, $\triangle$LNFB1, and $\triangle$LNFB2 for compactness; the appropriate delta–delta coefficients are always incorporated into all the three features. The DNN-HMM models were trained making use of the same data alignment obtained with a GMM-HMM recognition system trained in clean conditions employing MFCC features, linear discriminant analysis (LDA), and maximum likelihood linear transforms (MLLT), according to the tri2b Kaldi Aurora-4 recipe [19]. This recipe begins by training a uniphone system, uses alignments from that system for an initial triphone system, and finally uses those triphone alignments to train the final triphone system.

TABLE I
Description of Aurora-4 Testing Data Sets [18]

Test Set	Microphone	Noise	Group
1	Sennheiser HMD-414	Clean	A
2	Sennheiser HMD-414	Car	B
3		Babble	
4		Restaurant	
5		Street	
6		Airport	
7		Train	
8	different types of mic.	Clean	C
9	different types of mic.	Car	D
10		Babble	
11		Restaurant	
12		Street	
13		Airport	
14		Train	

In a previous optimization step, the DNN-HMM baseline system with multicondition training was tested with 24, 32, 40, and 56 MelFB filters. The lowest WER, 10.9%, was obtained with 40 filters using MN on a per-speaker basis. This WER is very competitive with current results in the literature for the same task (e.g., [3], [4], and [20]–[25]). The number of LNFB filters was also set to 40. The filter bandwidth for each channel, B, was set equal to 5.2 Barks. The spacing between contiguous filters is a function of the number of filters and is approximately equal to 0.40 Barks. Each DNN in the DNN-HMM systems consisted of seven hidden layers and 2048 units per layer. The number of units of the output layer equaled the number of pdfs of the corresponding GMM-HMM system. The Aurora-4 databases dev_330_01 and dev_330 were used as DNN cross-validation data for clean and multicondition training, respectively. For noisy training, datasets from dev_330_01 to dev_330_07 were employed for cross validation. Results are shown in Tables II, III, and IV.

V. Discussion

A. Dynamic Coefficients Based on the Numerator Filters

Table II compares the WER averaged over all test data for the three feature sets described above. MN was applied on a per-speaker basis. In all cases, the combination of static LNFB features with delta and delta–delta parameters based

TABLE II
COMPARISON OF ALGORITHMS FOR DELTA FEATURES FOR AURORA-4 DATA

Training	MelFB + ΔMelFB	LNFB + ΔLNFB1	LNFB + ΔLNFB2
Clean	32.66	49.21	36.73
Multinoise	16.00	18.42	16.06
Multicondition	10.90	13.97	12.26

TABLE III
DEPENDENCE OF WER FOR AURORA-4 ON MN AND MVN

Train Type	Feature Set	No Norm	MN per Spk	MVN per Spk	MN per Utt	MVN per Utt
Clean	MelFB + ΔMelFB	49.95	32.66	30.74	28.74	27.12
	LNFB + ΔLNFB2	51.10	36.73	28.63	33.38	24.03
Multinoise	MelFB + ΔMelFB	18.11	16.00	17.12	13.97	14.73
	LNFB + ΔLNFB2	16.13	16.06	14.85	14.55	13.35
Multicond.	MelFB + ΔMelFB	11.95	10.90	11.93	10.23	10.62
	LNFB + ΔLNFB2	12.27	12.26	12.11	11.62	11.18

TABLE IV
SUMMARY OF RESULTS FROM AURORA-4 TEST SETS

Training	Aurora Group	Mel FB + ΔMelFB	LNFB + ΔLNFB2
Clean (as in Group A)	A	2.39	2.65
	B	19.70	19.26
	C	21.69	**14.10**
	D	39.55	**34.02**
	Average	27.12	24.03
Multinoise (as in Group B)	A	2.56	3.19
	B	**6.11**	6.94
	C	16.57	**11.84**
	D	25.06	**21.70**
	Average	14.73	13.35
Multicondition (as in Group D)	A	3.42	3.62
	B	**6.35**	7.19
	C	7.12	7.38
	D	16.68	17.06
	Average	10.62	11.18
All-Condition Average		17.5	16.2

on the numerator filter energy, as defined in (6) and (7), LNFB+ΔLNFB2, leads to average relative reductions in WER equal to 25.4%, 12.8% and 12.2% for clean, noisy, and multicondition training, respectively, when compared with LNFB+ΔLNFB1. These results indicate clearly that the dynamic △LNFB2 coefficients based only on the numerator filters provide more useful information than coefficients based directly on the original △LNFB1 features.

B. Effect of Input Normalization

Table III compares averaged WERs over all of the Aurora-4 test sets, broken out by training type, obtained using MVN or MN applied by utterance and by speaker. As can be seen, MN and MVN always lead to lower WERs when applied utterance by utterance rather than speaker by speaker. (The Kaldi toolkit applies MN speaker by speaker by default.) The average reduction in WER provided by normalization per utterance compared to normalization per speaker is 9.1% for MN and 11.8% for MVN. This could be due to the fact that the SNR in the Aurora-4 database changes from one utterance to the next.

Table III also shows that the use of MVN rather than MN provides more effective normalization in the DNN-HMM system with the LNFB+△LNFB2 features, with the difference being particularly dramatic in the case of clean training. In contrast, MN was more effective than MVN when MelFB features were used in conjunction with multinoise and multicondition training. This may be because LNFB features always produce unimodal histograms (see Fig. 2), and hence, the use of MVN is more helpful than with MelFB features.

C. Comparison With MelFB Features

Table IV compares results obtained using the LNFB + △LNFB2 and MelFB+△MelFB features, broken out according to training and testing conditions, all with MVN invoked per utterance. The best feature set for a pair of train/test conditions is indicated in bold when the difference is statistically significant at the level of $p = .001$ or better, according to the NIST matched pairs test for sentence segment word error (MAPSSWE) [26].

In interpreting Table IV, we focus on the subset of comparisons that are statistically significant. With clean and multinoise training, we believe that the LNFB+△LNFB2 features provided superior performance for Aurora Groups C and D because of differences in the microphones used between the training and testing data. The only train/test combinations for which MelFB+△MelFB features provided significantly better performance than LNFB+△LNFB2 features were for the Group B testing data with multinoise and multicondition training. For the former case, training and testing conditions were completely matched, so no benefit was expected from LNFB+△LNFB2 features. In the latter case, the training data had multiple microphones and additive noise while the testing data had additive noise only. In this case, we suspect that any potential benefit from channel mismatch would be vitiated by the dominance of the matching noise. In general, the results of Table IV confirm our previous observations that LNFB coefficients are especially effective in compensating for the effects of channel mismatches, as had been demonstrated previously for the related LNCC features in speaker verification [10].

The global WER for LNFB-based features 7.4% relative smaller than the average WER obtained from MelFB features.

VI. CONCLUSION

A filter-bank version of LNCCs, LNFB, is described and applied to the Aurora-4 robust DNN-HMM-based speech recognition task. It is shown that the "Delta" and "Delta–Delta" versions of the LNFB features should be developed from the numerator term only in the LNFB expression. In addition, it is shown that MVN is more effective than MN for the LNFB features. The relative global WER over all conditions for LNFB features was 7.4% smaller than the average WER obtained using MelFB features. These results indicate that LNFB features provide better recognition accuracy for DNN-HMM ASR systems compared to Mel filterbank features, and that they are especially helpful in providing robustness to channel mismatches between training and testing data.

REFERENCES

[1] D. Yu, M. L. Seltzer, J. Li, J.-T. Huang, and F. Seide, "Feature learning in deep neural networks—Studies on speech recognition tasks," in *Proc. Int. Conf. Learning Representations*, 2013, Scottsdale, AZ, USA, pp. 1–9. [Online]. Available: https://arxiv.org/abs/1301.3605

[2] S. Haykin, "Multilayer perceptrons," in *Neural Networks and Learning Machines*, 3rd ed. London, U.K.: Pearson Prentice Hall, 2009.

[3] T. Drugman, Y. Stylianou, L. Chen, X. Chen, and M. Gales, "Robust excitation-based features for automatic speech recognition," in *Proc. IEEE Int. Conf. Acoustics, Speech Signal Process.*, 2015, Brisbane, Qld., Australia, pp. 4664–4668.

[4] M. Seltzer, D. Yu, and Y. Wang, "An investigation of deep neural networks for noise robust speech recognition," in *Proc. IEEE Int. Conf. Acoustics, Speech Signal Processing*, 2013, Vancouver, BC, Canada, pp. 7398–7402.

[5] Y. Obuchi and R. M. Stern, "Normalization of time-derivative parameters using histogram equalization," in *Proc. INTERSPEECH*, 2003, Lyon, France, pp. 665–668.

[6] O. Viikki and K. Laurila, "Cepstral domain segmental feature vector normalization for noise robust speech recognition," *Speech Commun.*, vol. 25, no. 1, pp. 133–147, 1998.

[7] X. Xiao, J. Li, E. S. Chng, H. Li, and C.-H. Lee, "A study on the generalization capability of acoustic models for robust speech recognition," *IEEE Trans. Audio, Speech, Language Process.*, vol. 18, no. 6, pp. 1158–1169, Aug. 2010.

[8] D. Yu and L. Deng, "Deep neural networks," in *Automatic Speech Recognition: A Deep Learning Approach*. London, U.K.: Springer, 2015, pp. 57–76.

[9] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient BackProp," in *Neural Networks: Tricks of the Trade,* 2nd ed. Berlin, Germany: Springer, 2012, pp. 9–48.

[10] V. Poblete *et al.*, "A perceptually-motivated low-complexity instantaneous linear channel normalization technique applied to speaker verification," *Comput. Speech Language*, vol. 31, no. 1, pp. 1–27, 2015.

[11] J. Fredes, J. Novoa, V. Poblete, S. King, R. Stern, and N. B. Yoma, "Robustness to additive noise of locally-normalized cepstral coefficients in speaker verification," in *Proc. INTERSPEECH*, 2015, Dresden, Germany, pp. 3011–3015.

[12] S. Seneff, "A joint synchrony/mean-rate model of auditory speech processing," *J. Phonetics*, vol. 16, pp. 55–76, 1988.

[13] E. Zwicker and E. Terhardt, "Analytical expressions for critical-band rate and critical bandwidth as a function of frequency," *J. Acoustical Soc. Amer.*, vol. 68, pp. 1523–1525, 1980.

[14] S. Furui, "Speaker-independent isolated word recognition using dynamic features of speech spectrum," *IEEE Trans. Acoustics, Speech Signal Process.*, vol. 34, no. 1, pp. 52–59, Feb. 1986.

[15] B. Gold, N. Morgan, and D. Ellis, "Feature extraction for ASR," in *Speech and Audio Signal Processing: Processing and Perception of Speech and Music*, 2nd ed. Hoboken, NJ, USA: Wiley, 2011, pp. 301–318 [Online Library].

[16] S. Xue, O. Abdel-Hamid, H. Jiang, L. Dai, and Q. Liu, "Fast adaptation of deep neural network based on discriminant codes for speech recognition," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 22, no. 12, pp. 1713–1725, Dec. 2014.

[17] Y. Miao, H. Zhang, and F. Metze, "Distributed learning of multilingual DNN feature extractors using GPUs," in *Proc. INTERSPEECH*, 2014, Singapore, pp. 830–834.

[18] G. Hirsch, "Experimental framework for the performance evaluation of speech recognition front-ends on a large vocabulary task, version 2.0, AU/417/02," *ETSI STQ Aurora DSR Working Group*, 2002.

[19] D. Povey *et al.*, "The Kaldi speech recognition toolkit," in *Proc. IEEE Autom. Speech Recognit. Understanding Workshop*, Dec. 2011, Waikoloa, HI, USA, N° EPFL-CONF-192584.

[20] B. Li and K. C. Sim, "A spectral masking approach to noise-robust speech recognition using deep neural networks," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 22, no. 8, pp. 1296–1305, Aug. 2014.

[21] Z. Tang, D. Wang, and Z. Zhang, "Recurrent neural network training with dark knowledge transfer," in *Proc. IEEE Int. Conf. Acoustics, Speech Signal Process.*, 2016, Shanghai, China, pp. 5900–5904.

[22] Y. Qian, M. Bi, T. Tan, and K. Yu, "Very deep convolutional neural networks for noise robust speech recognition," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 24, no. 12, pp. 2263–2276, Dec. 2016.

[23] A. Bayestehtashk, I. Shafran, and A. Babaeian, "Robust speech recognition using multivariate copula models," in *Proc. IEEE Int. Conf. Acoustics, Speech Signal Process.*, 2016, Shanghai, China, pp. 5890–5894.

[24] H. B. Sailor and H. A. Patil, "Unsupervised deep auditory model using stack of convolutional RBMs for speech recognition," in *Proc. INTERSPEECH*, 2016, San Francisco, CA, USA, pp. 3379–3383.

[25] S. Kundu, K. C. Sim, and M. Gales, "Incorporating a generative front-end layer to deep neural network for noise robust automatic speech recognition," in *Proc. INTERSPEECH*, 2016, San Francisco, CA, USA, pp. 2359–2363.

[26] D. S. Pallett, W. M. Fisher, and J. G. Fiscus, "Tools for the analysis of benchmark speech recognition tests," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Process.*, 1990, Albuquerque, NM, USA, pp. 97–100.