Universidad Zaragoza

Trabajo Fin de Grado

Separación de música y voz en grabaciones mono
Singing-voice separation from monoaural recordings

Autor

Ignacio Zay Pinilla

Director

Alfonso Ortega Giménez

ESCUELA DE INGENIERÍA Y ARQUITECTURA

2018

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

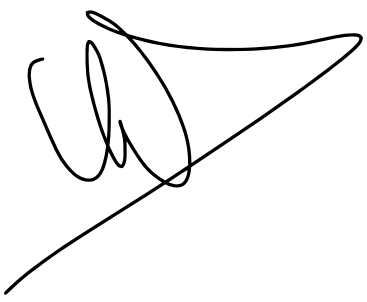
D./Dª. IGNACIO ZAY PINILLA,

con nº de DNI 73027997F en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) GRADO, (Título del Trabajo) SEPARACIÓN DE MÚSICA Y VOZ EN GRABACIONES MONO

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 20 NOVIEMBRE 2018



Fdo: _____

Índice

Capítulo 1

Motivación y objetivos

La música es una de las mayores formas artísticas de expresión que existen hoy en día y juega un papel fundamental formación del ser humano. La separación de fuente de audio consiste en aislar las distintas fuentes de sonido en una pista de audio. Algunos ejemplo de la separación de fuente serían aislar la guitarra en una canción, aislar la voz en un entorno ruidoso o extraer la voz principal de una canción. La separación automática de dichas fuentes en una escena auditiva posee numerosas aplicaciones como la identificación de instrumentos, remezcla de audio, efectos de disk jockey, karaoke o transcripción de la melodía.

La separación de la fuente de música es uno de los grandes temas a investigar dentro del área del procesado de señal. En este proyecto, se ha estudiado dicho problema concretamente, la separación de la parte vocal y la parte instrumental en una grabación de tipo mono.

El objetivo principal de este trabajo es utilizar diferentes tipos de procedimientos basados en métodos de procesado de señal clásico y sistemas basados en inteligencia artificial así como un sistema de evaluación objetivo que nos permita medir la calidad de la separación.

Previamente, se llevará al cabo, un estudio de conocimiento general en el área de la separación de fuentes auditivas y el aprendizaje profundo ("deep learning"). El objetivo de este estudio se centrará en entender la dificultad de separación de fuente, transformaciones al dominio frecuencial, estudio de método de separación fuente clásicos, funcionamiento básico de redes neuronales así como algunos de sus diferentes tipos y sus diferentes parámetros.

Una vez adquirido los conocimientos teóricos básicos necesarios para llevar al cabo este trabajo se procederá a la familiarización del entorno del trabajo, en concreto al lenguaje de programación Python y en la biblioteca de código abierto Tensorflow así como la configuración de Google Colaboratory utilizada para el entrenamiento de inteligencia artificial.

Se utilizarán secciones de código externo que se han adaptado para el desarrollo de los diferentes experimentos en algoritmos de RPCA, REPET y diferentes configuraciones de redes neuronales densas y posteriormente recurrentes. Para ello, se modificarán los valores parámetros de configuración y se evaluarán mediante métricas específicas, el comportamiento de dichos algoritmos.

Finalmente se realizará una comparación objetiva con los resultados obtenidos con los diferentes métodos.

1.1. Estructura de la memoria

A parte de este primer capítulo donde se han mencionado la motivación y los objetivos de este trabajo, este documento está formado de cuatro capítulos más descritos a continuación:

Capítulo 2 - Introducción en este capítulo se resalta una explicación al reto de separar la voz de la musica en una canción además de los conceptos teóricos que se van a llevar al cabo para poder afrontar este reto.

Capítulo 3 - Entorno experimental en este capítulo se muestran los diferentes experimentos que se van a llevar al cabo así como las herramientas utilizadas en dichos experimentos y los datos utilizados durante el estudio.

Capítulo 4 - Resultados este capítulo contiene los resultados obtenidos a través de los experimentos realizados en este trabajo, así como un análisis y una interpretación de los mismos.

Capitulo 5 - Conclusión y líneas futuras en este capítulo se recogen las conclusiones extraídas tras la realización de este trabajo además de comentar de forma breve las posibles lineas a seguir.

Capítulo 2

Introducción

2.1. Separación de voz y música

La mayoría de las canciones que se producen hoy en día, suelen estar formadas por una voz acompañada de instrumentos. Normalmente, cuando se escucha una canción de estas características es casi inevitable centrar nuestra atención en la voz producida por el cantante. La voz es una onda de presión producida por el aparato fonador y que es percibida y analizada por el oído para posteriormente, ser procesada por el cerebro.

El rango de frecuencias audibles por el ser humano en el mejor de los casos es de 20Hz a 20KHz [1]. El ser humano posee una gran capacidad para identificar varias fuentes de sonido producidas en un mismo instante de tiempo es decir, puede separar frecuencias de diferentes fuentes solapadas en ese mismo instante de tiempo. Sin embargo, para las máquinas se trata de una tarea muy complicada.

El análisis de la escena auditiva (ASA) es un modelo propuesto para la base de la percepción auditiva. Esto se entiende como el proceso por el cual el sistema auditivo humano organiza el sonido en elementos significativos perceptualmente. El término fue acuñado por el psicólogo Albert Bregman [2]. El concepto relacionado en la percepción de la máquina es el análisis de la escena auditiva computacional (CASA), que está estrechamente relacionado con la separación de fuentes y separación de la señal ciega [3].

Otro problema propuesto muy conocido es la separación de fuente de audio ciega (BASS) [4]. Este problema trata la separación de un conjunto de señales fuente de un conjunto de señales mezcladas, sin la ayuda de información (o con muy poca información) sobre las señales fuente o el proceso de mezcla . En general, este problema está muy poco determinado, pero se pueden derivar soluciones útiles en una sorprendente variedad de condiciones. Uno problemas de este modelo se produce cuando uno o varios micrófonos graban un sonido que es la mezcla de sonidos provenientes de varias fuentes.

A la hora de separar la voz de una canción podemos encontrar varias dificultades. Las fuentes musicales pueden ser de instrumentos de diferente tipo, éstos a su vez pueden producir sonidos en diferentes tonos y sonoridad e incluso puede variar la posición espacial de estos. Éstos, a su vez pueden repetirse siguiendo un patrón a lo largo de la canción o pueden aparecer en determinados momentos de manera aislada. La voz del cantante puede tener diferentes variaciones de todo tipo que pueden ser debidas a la canción, al entorno e incluso a la calidad del micrófono. Además dichas frecuencias pueden solaparse con la frecuencia de algún instrumento [5].

2.2. Estado del arte

El espectro de una señal es una herramienta básica en el análisis de audio y otros campos. La representación del espectro en el dominio frecuencial puede ayudar a entender mejor su contenido, que con una representación en el dominio temporal. Todos los experimentos realizados en este trabajo tienen como entrada al sistema el espectro de la señal por lo que se va a realizar una breve descripción de dicha transformación.

2.2.1. Espectrograma

El espectrograma, es el valor absoluto de la STFT. En una señal de audio la parte mas importante es el valor absoluto del espectro ya que el oído humano es menos sensible a la fase. En el espectrograma gráficamente consiste en representar entre los ejes tiempo y frecuencia el valor absoluto le la STFT mediante un mapa de colores expresados en decibelios. Uno de los problemas al realizar esta transformación es que no podemos recuperar la señal original a partir del espectrograma ya que perdemos la información de la fase. Por lo tanto la fase deberá ser guardada antes de realizar el espectrograma para posteriormente poder recuperar la señal original.

Análisis espectral localizado

La transformada de Fourier en tiempo discreto es un herramienta matemática para representar las señales en tiempo discreto en el dominio de la frecuencia. Esta transformada viene dada por la expresión:

$$X(\omega) = \sum_{n=-\infty}^{\infty} x(n)e^{j\omega n} \tag{2.1}$$

Físicamente, $X(\omega)$ representa el contenido en frecuencia de la señal $x(n)$. En otras palabras, $X(\omega)$ es una descomposición de $x(n)$ en sus componentes de frecuencia [6].

Si muestreamos la transformada de Fourier en tiempo discreto en N frecuencias discretas $\omega(k) = 2\pi F_s/N$ donde F_s es la frecuencia de muestreo y N el número total de muestras de la señal, obtenemos la transformada de Fourier discreta cuya expresión es:

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{j\frac{2\pi}{N}kn} \tag{2.2}$$

Debido a que las señales de voz son no estacionarias, no se realiza la transformada de Fourier así definida sino que utilizaremos la trasformada de Fourier localizada (STFT) que consiste en el enventanado de la señal utilizando la propiedad de que la voz es cuasi-estacionaria en intervalos cortos de tiempo [7]. Esta transformada puede definirse de la siguiente manera:

$$X(m, \omega) = \sum_{n=-\infty}^{\infty} x(n) w[n-m] e^{j\omega n} \tag{2.3}$$

Donde $x[n]$ es la señal y $w[n]$ es la ventana. Una vez definido el tamaño de ésta, se procede a seleccionar el porcentaje de solapamiento entre ventanas junto con el tipo de ventana y se multiplica la señal por dicha ventana. Acto seguido se realiza la transformada rápida de Fourier (FFT) y finalmente conseguimos las tramas en el dominio frecuencia.

2.2.2. Robust Principal Component Analisis

Descripción

El primer método utilizado en ésta investigación ha sido el algoritmo Robust Principal Component Analysis (RPCA). Como se explica en [8] consiste en resolver el problema de optimización minimizando $|L\|_* + \lambda\|S\|_1$ dado $L + S = M$, donde $M \in \mathbb{R}^{n_1 \times n_2}$ se trata de una matriz dispersa, es decir, una matriz donde la mayoría de sus componentes son cero siendo esta también una matriz de mayor rango que $L \in \mathbb{R}^{n_1 \times n_2}$ $S \in \mathbb{R}^{n_1 \times n_2}$. $\|.\|_*$ y $\|.\|_1$ son la suma de los valores singulares y la suma de los valores absolutos respectivamente. El valor escogido por lambda ha sido $\lambda = 1/\sqrt{\text{máx}\,(n_1 n_2)}$ como se aconseja en [8].

Procedimiento

A cada clip de la base de datos se le aplica la STFT para posteriormente extraer el módulo. A éste modulo se le ha aplicado Augmented Lagrange Multiplier (ALM) que se trata de un algoritmo eficiente para solventar el problema de RPCA resolviendo, $L+S = |M|$. A partir de aquí obtenemos las dos matrices S y L donde L correspondería con la parte instrumental debido a que el sonido de los instrumentos siguen un patrón

que se repite a lo largo del tiempo por lo que poseerá un rango menor mientras que S que correspondería con la parte vocal posee mayor variedad y por lo tanto, mayor rango.

Para poder volver a obtener las formas de onda después de aplicar el algoritmo se concatena las fases de cada uno de las señales con las correspondientes matrices L y S para posteriormente realizar la STFT inversa (ISTFT) [8].

2.2.3. REpeating Pattern Extraction Technique

Descripción

Como hemos mencionado anteriormente la parte instrumental de una canción se caracteriza por seguir unos patrones que se repiten a la largo de ésta. La base del algoritmo REPET se centra en extraer patrones musicales repetidos dentro de la señal de audio [9].

Existen varios tipos de implementaciones y mejoras de este algoritmo. Sin embargo, por simplicidad, hemos estudiado la primera implementación que se hizo de este algoritmo llamado Original REPET. [10]

Procedimiento

La figura 2.1 ilustra los pasos a seguir para llevar al cabo esta implementación donde se puede observar que consta de 3 fases.

La primera fase consiste en extraer el espectrograma de la señal de entrada (en nuestro caso sería la señal mezclada tanto con la voz como con la música) para después obtener el espectrograma de los ritmos de la canción (beat spectrum) representado en la imagen de la parte superior derecha, donde se puede apreciar cual es la duración del patrón que se repite a lo largo de la canción y por lo tanto obtener el patrón de repetición

En la segunda fase, se fracciona el espectrograma de la señal en segmentos dados por la duración del patrón de repetición obtenido en la fase anterior. Una vez obtenidos los diferentes segmentos se realiza la media entre todos ellos ilustrado en la parte central derecha.

En la tercera fase, se replica el patrón obtenido promedio de tal forma que se obtenga la dimensión del espectrograma original. Obteniendo el espectrograma repetido para posteriormente aplicarle una máscara tiempo-frecuencia obteniendo finalmente la parte del espectro que tiene relación con la parte instrumental de la canción. Por último se vuelve convertir el espectrograma obtenido utilizando la ISTFT obteniendo la señal en

tiempo. Para poder obtener la parte vocal de la canción se resta a la señal original la parte instrumental aislada obtenida en este algoritmo obteniendo finalmente la parte vocal de la canción aislada.

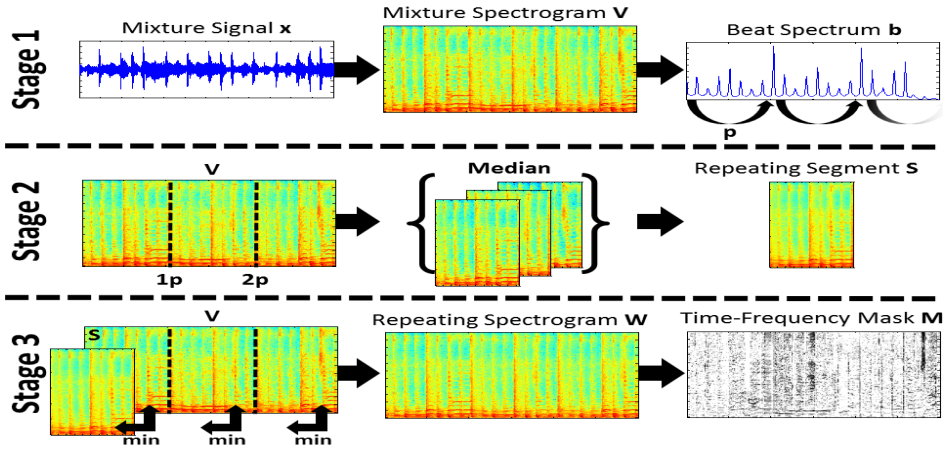


Figura 2.1: Fases del algortimo REPET. Fuente: http://zafarrafii.com/

2.2.4. Redes neuronales

Las redes neuronales artificiales han tenido un gran auge durante los últimos años, quedando ampliamente demostrada la posibilidad de ser empleadas para resolver tareas ingenieriles que requieran un cierto grado de inteligencia, no alcanzable mediante los algoritmos informáticos convencionales.

Introducción

Las redes neuronales son un modelo computacional basado en un gran conjunto de unidades neuronales simples (neuronas artificiales) de forma aproximadamente análoga al comportamiento observado en las neuronas en los cerebros biológicos. Las redes neuronales artificiales han generado mucho entusiasmo en la investigación y la industria del aprendizaje automático, gracias a los muchos resultados innovadores en reconocimiento de voz [11], visión artificial [12] y procesamiento de texto [13]. La figura 2.2 ilustra el esquema de una neurona cerebral biológica. Cada neurona recibe la información desde sus dendritas y produce su salida a través de su único axón. El axón finalmente se ramifica y se conecta a través de sinapsis con las dendritas de otras neuronas [14].

La estructura de una neurona artificial puede verse ilustrada en la figura 2.3 donde se aprecian tres partes: las entradas, los pesos y las salidas. Las entradas son las señales introducidas en la neurona. Los pesos miden el grado de importancia de la conexión de cada una de las entradas con la neurona cuyos valores se van modificando durante la fase de entrenamiento de la red hasta llegar a los valores óptimos. La salida de una

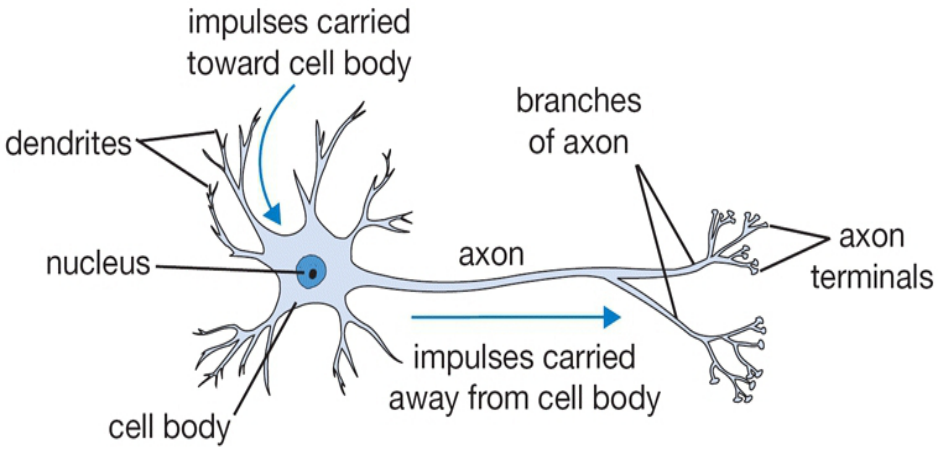


Figura 2.2: Esquema de una neurona biológica. Fuente: http://cs231n.github.io/neural-networks-1/

neurona, a modo de resumen, es el resultado de sumar todas sus entradas multiplicadas por sus pesos y aplicar a esta suma lo que se conoce como función de activación. A este modelo de red neuronal tan simple se le conoce también como perceptrón.

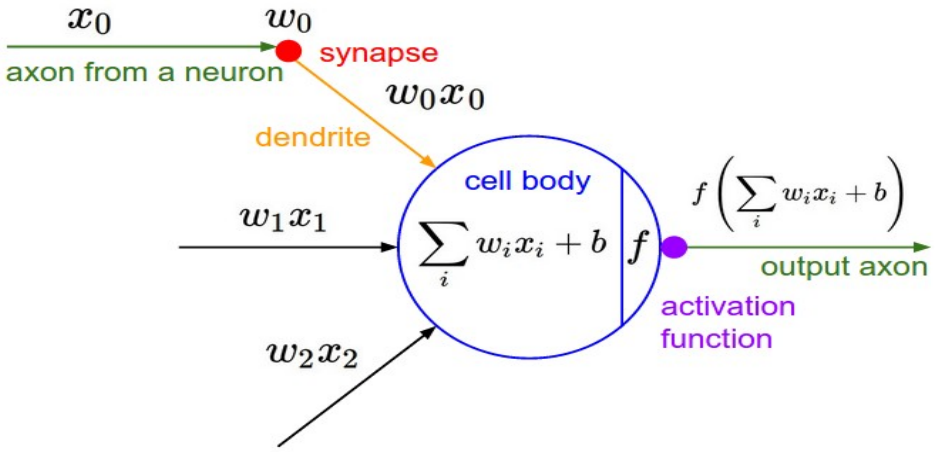


Figura 2.3: Esquema de una neurona artificial. Fuente: cs231n

Conceptos básicos

Arquitectura Las neuronas se agrupa en unidades que llamaremos capas. Cada capa, se caracteriza por tener varios parámetros siendo algunos de éstos, el número de neuronas por capa, el tipo de conexión entre ella y con el resto de capas, y el numero de conexiones entre capas. La figura 2.4 ilustra la estructura básica de una red neuronal. En ella podemos encontrar la capa de entrada (rojo), dos capas ocultas (azul) y la capa de salida (verde). La capa de entrada contiene las entradas a la red neuronal, en este caso tendría tres nodos de entrada. Las capas ocultas son aquellas que se encuentran entre la capa de entrada y la capa de salida. Se caracterizan porque

las neuronas de este tipo de capas no tienen contacto con el mundo exterior a diferencia de la capa de entrada y de salida, de ahí su nombre. La capa de salida contendría la señal estimada por la red. Normalmente a la hora de contar el número de capas de una red no se tiene en cuenta la capa de entrada. En nuestro ejemplo, tendríamos una red de 3 capas. Las capas de una red pueden expresarse matemáticamente como un vector d^n donde n es la capa y su dimensión varía en función del número de neuronas que posee dicha capa. Los pesos entre cada capa puede representarse de forma matricial W_{ij} donde cada elemento de dicha matriz representa las conexiones entre neuronas.

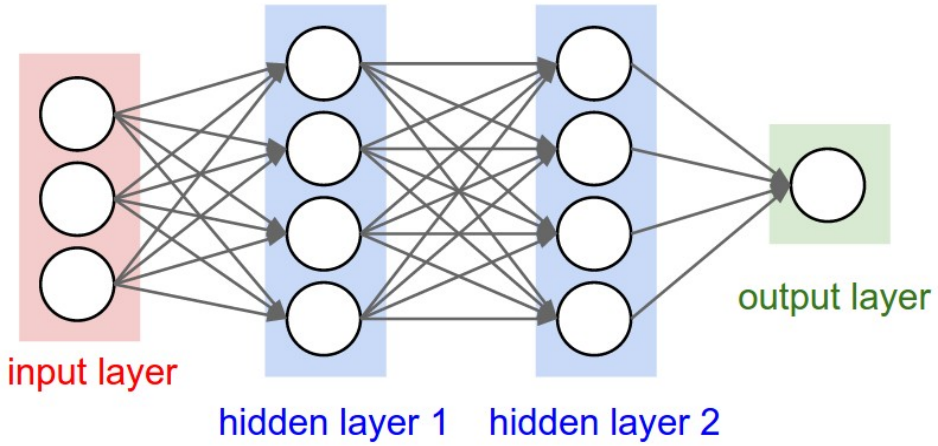


Figura 2.4: Esquema de una red neuronal. Fuente: cs231n

Sesgo o Bias Es un escalar que se añade a cada capa dentro de una red neuronal, podemos considerarla como una entrada extra a cada neurona de una capa. Su principal función es desplazar hacia la izquierda o hacia la derecha en el eje x la función de activación.

Aprendizaje Se trata de un proceso mediante el cual se van aplicando un conjunto de entradas a la red neuronal donde se van ajustando adecuadamente e internamente todos y cada uno de los pesos asociados a cada rama para poder obtener la salida deseada. Para llevar al cabo el aprendizaje de una red neuronal es necesario dividir el proceso es dos fases: la fase de entrenamiento y la fase prueba [15]. Para ello se divide la base de datos que utilizaremos como entrada en dos partes: una para la fase de entrenamiento y otra para la fase de prueba. Normalmente se suele dar un 80 % aproximadamente a la fase de entrenamiento y un 20 % a la fase de prueba. Sin embargo en algunas ocasiones se puede dar un mayor porcentaje a la fase de prueba que a la fase de entrenamiento con el fin de obtener unos resultados más fiables.

Fase de entrenamiento Una vez escogido el tipo de neurona a utilizar y el tipo de arquitectura de la red, se van introduciendo los datos en la red neuronal de forma que el valor de los pesos se vayan ajustando hasta conseguir un resultado óptimo. Este proceso puede dividirse básicamente en dos tipos: aprendizaje supervisado y aprendizaje no supervisado. En el primero se le presenta a la red un conjunto de patrones de entrada junto con la salida esperada. Los pesos se van modificando de manera proporcional al error que se produce entre la salida real de la red y la salida esperada. El el caso del segundo, se presenta a la red un conjunto de patrones de entrada. No hay información disponible sobre la salida esperada. El proceso de entrenamiento en este caso deberá ajustar sus pesos en base a la correlación existente entre los datos de entrada [16].

Fase prueba En ésta parte se presentan a la red neuronal los datos escogidos para la parte de prueba que son desconocidos por la red. Si el entrenamiento se ha realizado de forma correcta y los valores de los pesos son los óptimos, la red tendría que ser capaz y analizar los datos de forma correcta debido a su aprendizaje.

Métodos de entrenamiento

Algoritmo Back Propagation Se trata de un algoritmo de aprendizaje supervisado que junto con la diferencia entre la señal estimada y la señal deseada utiliza las derivadas parciales respecto de cada uno de sus parámetros. El primer paso de este algoritmo es la inicialización de los parámetros de la red, concretamente el valor de los pesos y de los bías de cada neurona generalmente de manera aleatoria. Acto seguido, dado un patrón p de entrada $X^p : x_1^p.....x_N^p$ éste se transmite a través de los pesos w_{ij} desde la capa de entrada hacia la capa oculta [17]. Cada neurona de la primera capa oculta suma sus entradas multiplicados por sus respectivos pesos y el bias para posteriormente aplicarle una función de activación obteniendo la salida de cada neurona de la primera capa oculta. Estos pasos se realizan de forma secuencial por cada capa oculta hasta llegar a la capa de salida. Una vez obtenida los valores correspondientes con la salida estimada, se calcula el error comparando la salida deseada con la conocida previamente. Uno de los métodos mas comunes a la hora de obtener dicho error es usando el denominado error cuadrático medio (MSE) calculándose de la siguiente manera:

$$E^p = \frac{1}{2}\sum_{k=1}^{M}(d_k^p - y_k^p)^2 \tag{2.4}$$

donde d_k^p es la salida deseada para la neurona de salida k ante el patrón p. Posteriormente se obtiene la medida del error general mediante la suma de todos los errores de cada patrón.

La base del algoritmo de propagación hacia atrás para la modificación de los pesos es la técnica conocida como gradiente decreciente. De esta forma se reduce el error ajustando los pesos en la dirección:

$$\Delta_w J = \sum_{p=1}^{P} \frac{\partial E^p}{\partial w_{ji}} \tag{2.5}$$

A nivel práctico, la forma de modificar los pesos de forma iterativa consiste en aplicar la regla de la cadena a la expresión del gradiente y añadir una tasa de aprendizaje [17]:

$$w_{n+1} = w_n - f(\alpha, \Delta_w J(w)) \tag{2.6}$$

donde w_n y $f(\alpha, \Delta_w J(w))$ es el valor de los pesos en el instante actual y el término de actualización de parámetros respectivamente. Los parámetros de esta actualización α y $\Delta_w J(w)$ son la tasa de aprendizaje y el gradiente de la función coste con respecto a los pesos w.

2.2.5. Función de activación

Puede definirse como una función que se aplica a la salida de la neurona. Básicamente mapea la salida de una neurona entre un rango determinado, generalmente entre 0 y 1 o -1 y 1 aunque depende de la función de activación. Las funciones de aplicación pueden dividirse básicamente en dos tipos: lineales y no lineales.

La función de activación lineal como puede verse en la figura 2.5 la salida de esta función no está acotada entre ningún rango. Matemáticamente viene expresada por $f(x) = ax$. Una de las limitaciones de la función de activación lineal es a la hora del entrenamiento, al calcular la derivada en el algoritmo back propagation, el resultado es una constante y no depende de la variable independiente. Esto se traduce en que si se produce un error de predicción, los cambios realizados en la propagación hacia atrás, son constantes y no dependen de los cambios producidos en la entrada.

La funciones de activación no lineales a diferencia de las lineales posee un rango limitado en su salida. Este tipo de funciones son las mas utilizadas a la hora del diseño de redes neuronales. Entre las funciones de activación no lineal más comunes podemos encontrar las funciones **sigmoid**, **relu** y **tanh**.

La función **sigmoid** se define matemáticamente como:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.7}$$

Como se puede apreciar en la figura 2.7 los valores comprendidos entre -2 y 2 en el eje de abscisas varían de forma brusca, lo que se traduce en un pequeño cambio en la

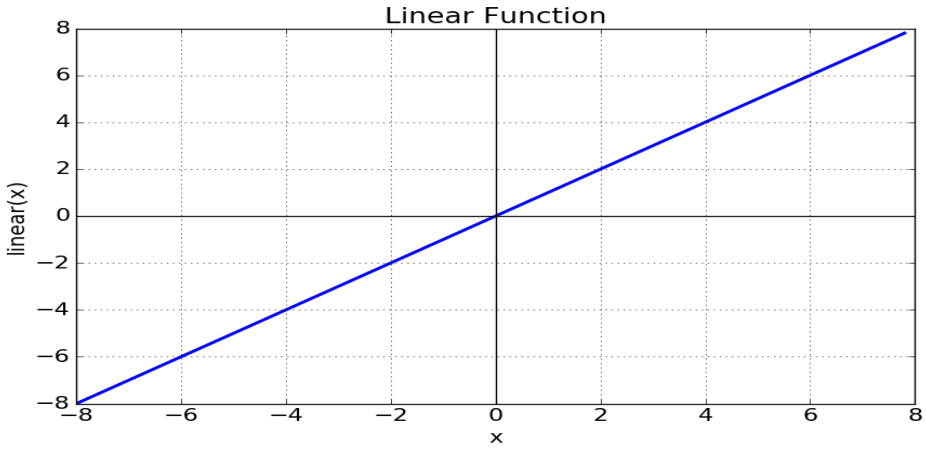


Figura 2.5: Función de activación lineal

entrada dentro de este rango produce un cambio significativo en la salida de la función de activación. Otra ventaja de esta función es que posee un rango de salida comprendido entre 0 y 1 en comparación con la lineal que posee un rango de menos infinito a mas infinito. La función **tanh** es análoga a la función **sigmoid** con la diferencia de que tiene un rango de salida comprendido entre -1 y 1.

La función **relu** matemáticamente definida como:

$$f(x) = max(0, x) \tag{2.8}$$

obtenemos un valor de x si x es positivo y 0 si son negativos como puede verse en la figura 2.6. Una de las ventajas de esta función de activación es que reduce el problema del desvanecimiento de gradiente que explicaremos en apartados posteriores.

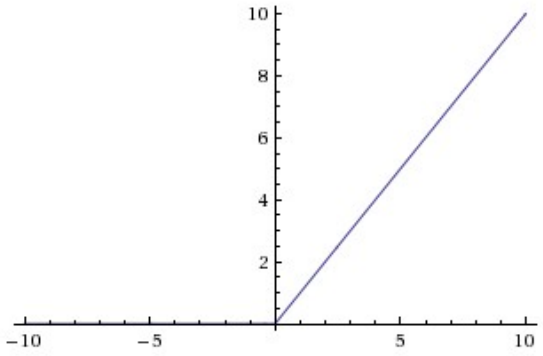


Figura 2.6: Función relu

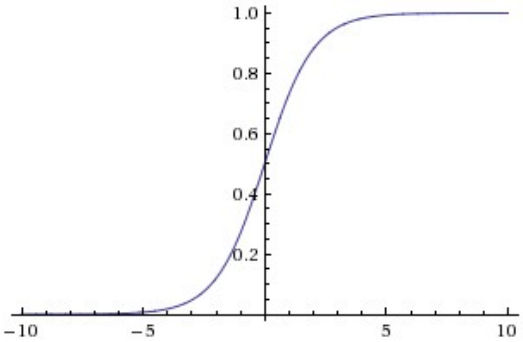


Figura 2.7: Funcion sigmoid

Redes recurrentes

Uno de los problemas que se puede tener a la hora de trabajar con redes neuronales densas es que no guardan ningún tipo de información sobre eventos secuencialmente distantes cuando se trabaja con secuencias. Cada ejemplo que se le muestra a la entrada

de la red no posee relación con el resto. Las redes neuronales recurrentes no poseen este tipo de problema ya que tienen la propiedad de modelar dependencias temporales de los datos introducidos. Como se ilustra en la figura 2.8 cada capa de una red neuronal recurrente posee dos entradas. La primera sería la entrada de los datos a la capa como en las redes densas y la segunda es una realimentación de la salida obtenida en un instante de tiempo anterior x_{t-1}. Matemáticamente se puede describir de la siguiente forma:

$$a_t = g(W_{aa}a_{t-1} + W_{ax}x_t + b_a) \tag{2.9}$$

$$h_t = g(W_{ha}a_t + b_h), \tag{2.10}$$

En la ecuación (2.9), a_t, W_{aa}, W_{ax} y b_a son la salida de la neurona en el instante actual, la matriz de pesos correspondiente a la realimentación de la capa, matriz de pesos correspondiente a la entrada de la capa y el bias asociado a la capa recurrente. En la ecuación (2.10), h_t, W_{ha}, y b_y corresponden con la capa de salida de la red, la matriz de pesos entre la capa recurrente y la de salida y el bias asociado a la capa de salida respectivamente.

Uno de los mayores problemas que poseen las redes neuronales recurrentes es el denominado desvanecimiento de gradiente.

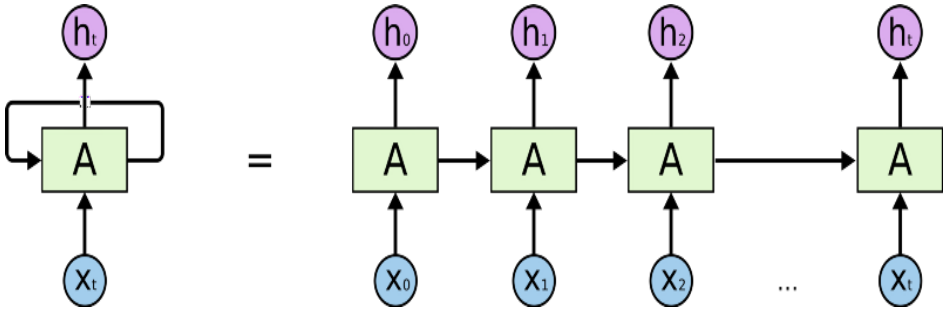


Figura 2.8: Arquitectura de una red neuronal recurrente de una capa. Fuente: https://towardsdatascience.com

Desvanecimiento de gradiente El desvanecimiento de gradiente se produce a la hora de realizar el algoritmo de propagación hacia atrás calculando el error con respecto al valor de los pesos donde el gradiente tiende a ser más pequeño conforme vamos avanzando hacia atrás en la red. En otras palabras, las neuronas en las primeras capas aprenden mucho más lento comparado con las neuronas de las últimas capas y por lo

tanto las primeras capas necesitan un mayor tiempo de entrenamiento. Las primeras capas de nuestra red son especialmente importantes porque aprenden y detectan los patrones básicos de dicha red. Por lo tanto, si las primeras capas están bien entrenadas nos darán una mayor precisión en las capas posteriores [18].

Redes Gated Recurrent Unit Las redes GRU son una versión mejorada de las redes recurrentes estándar. Una de las principales ventajas de estas redes es que resuelven el problema del desvanecimiento de gradiente mencionado en el apartado anterior. Para ello utilizan lo que se llama una puerta de reinicio y una puerta de actualización siendo éstas dos vectores cuya función es decidir qué información debe pasar a la salida.

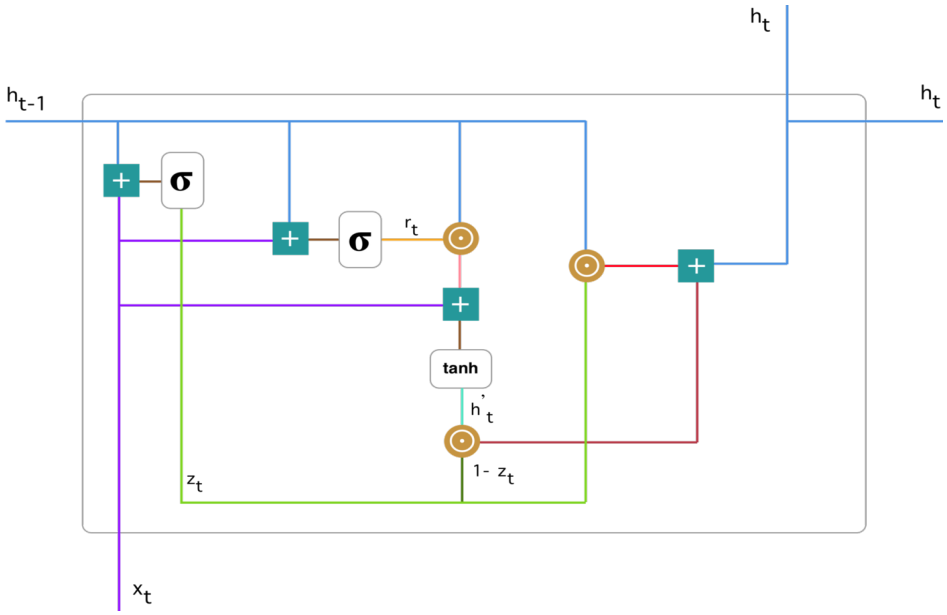


Figura 2.9: Esquema de una celda GRU. Fuente: https://towardsdatascience.com

En la figura 2.9 se puede observar las partes de la que está formada una celda GRU. La primera parte (arriba a la izquierda) observamos la puerta de actualización la cual, puede expresarse mediante la siguiente fórmula:

$$z_t = \sigma(W^{(z)}x_t + U^{(z)}h_{t-1}) \tag{2.11}$$

donde la entrada x_t multiplicada por su respectivo peso $W(z)$. Lo mismo sucede con h_{t-1} que mantiene la información de la unidad anterior y se multiplica por su respectivo peso $U(z)$. Finalmente se suman ambos resultados y se la aplica la función de activación sigmoid. La principal función de esta puerta es seleccionar la cantidad de información de la unidad anterior en los instantes anteriores.

La segunda parte (segundo sumador empezando por la izquierda) podemos encontrar la puerta de reinicio la cuál, básicamente decide cuanta información de los instantes previos va a ser olvidada. La expresión de esta puerta posee la misma estructura que de la puerta anterior:

$$r_t = \sigma(W^{(r)}x_t + U^{(r)}h_{t-1}) \tag{2.12}$$

Finalmente las puertas afectan a la salida final de la siguiente manera. Primero se introduce un nuevo contenido de memoria que utilizará la puerta de reinicio para almacenar la información relevante del pasado calculándose de la siguiente manera:

$$h'_t = tanh(W^{(r)}x_t + r_t \odot Uh_{t-1}) \tag{2.13}$$

De esta forma se determinará lo que hay que eliminar de los estados anteriores. Aplicado a nuestro caso, si al principio de la canción solo suenan instrumentos y pasados unos segundos aparece la voz mezclada con los instrumentos, nosotros solo necesitaríamos esta última parte por lo que la red tendría que asignar al vector r_t un valor muy próximo a cero. Finalmente, es necesario calcular el vector h_t que contiene la información de la unidad actual y la transmita a la red. Para ello, utilizamos la puerta de actualización la cual, determina qué información actual de la unidad y de los pasos anteriores.

2.3. Separación de voz y música mediante redes neuronales

2.3.1. Arquitectura de la red

La entrada a la red neuronal como hemos mencionado en capítulos anteriores, es el módulo del espectro obtenido al realizar la STFT. Las salidas, y_{1_t} y y_{2_t}, y las salidas estimadas, $\widehat{y}_{1_t}$ y $\widehat{y}_{2_t}$, de la red son el modulo del espectro de las diferentes fuentes. La figura 2.10 muestra un ejemplo de esta arquitectura. Además, para suavizar los resultados de la separación empleamos una máscara de tiempo - frecuencia. Dadas las entradas, x_t, desde la mezcla, obtenemos las salidas estimadas y_{1_t} y y_{1_t} a través de la red. La máscara tiempo frecuencia m_t se define de la siguiente manera:

$$m_t(f) = \frac{|\widehat{y}_{2_t}|}{|\widehat{y}_{1_t} + \widehat{y}_{2_t}|}, \tag{2.14}$$

donde $f \in \{1, ..., F\}$ representa las diferentes frecuencias. Una vez que se realiza la máscara m_t, ésta se aplica al módulo del espectro z_t de la señal mezclada (vocal+música) para obtener la estimación de la separación del espectro $\widehat{s}_{1_t}$ y $\widehat{s}_{1_t}$ que

corresponden con las fuentes 1 y 2 de la siguiente forma:

$$\widehat{s}_{1t}(f) = m_t(f)z_t(f), \tag{2.15}$$

$$\widehat{s}_{2t}(f) = (1 - m_t(f))z_t(f) \tag{2.16}$$

donde $f \in \{1, ..., F\}$ representa las diferentes frecuencias.

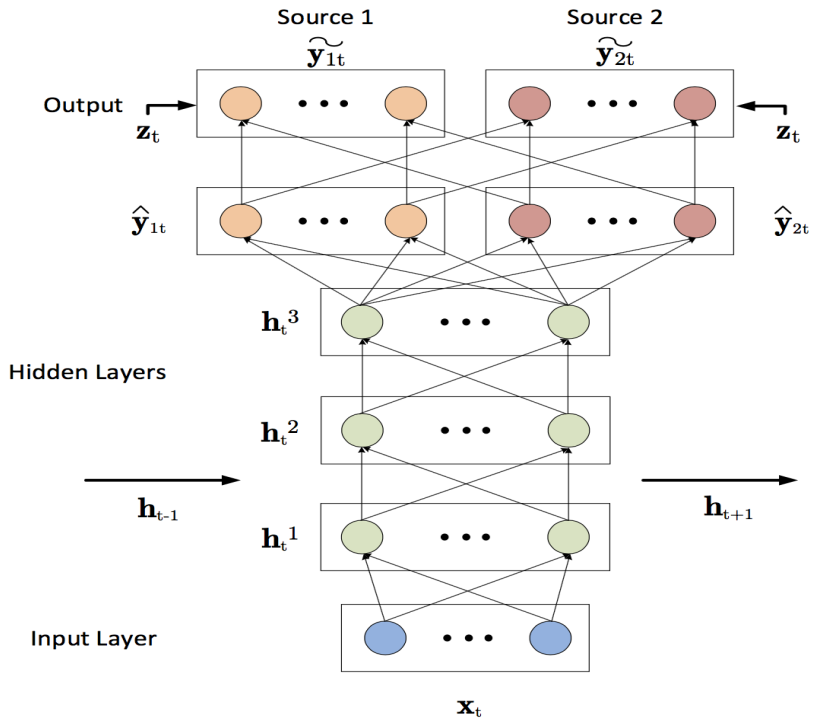


Figura 2.10: Arquitectura de la red neuronal propuesta

La máscara tiempo frecuencia puede verse también como una capa en la red neuronal. En lugar de entrenar la red y aplicar la máscara a los resultados de forma separada, podemos unirlo y entrenar los modelos con la máscara. Añadimos entonces una capa extra a la salida original de la red de la siguiente manera:

$$\tilde{y}_{1t}(f) = \frac{|\widehat{y}_{2_t}|}{|\widehat{y}_{1_t} + \widehat{y}_{2_t}|} \odot z_t, \tag{2.17}$$

$$\tilde{s}_{2t}(f) = \frac{|\widehat{y}_{2_t}|}{|\widehat{y}_{1_t} + \widehat{y}_{2_t}|} \odot z_t, \tag{2.18}$$

donde el operador $\odot$ es el producto muestra a muestra (producto Hadamard). De esta manera podemos optimizar la red con la función de enmascaramiento conjuntamente.

Dadas las salidas estimadas $\widehat{y}_{1_t}$ y $\widehat{y}_{2_t}$ de las fuentes originales y_{1_t} y y_{2_t} se han optimizado los parámetros de dicha red utilizado el criterio de mínimo error cuadrático definido de la manera siguiente.

$$J_{MSE} = \|\widehat{y}_{1t} - y_{1_t}\|_2^2 + \|\widehat{y}_{2t} - y_{2_t}\|_2^2 \tag{2.19}$$

Capítulo 3

Marco experimental

3.1. Base de datos

Para llevar al cabo el estudio del comportamiento de los sistemas propuestos en este trabajo es necesario introducir diferentes señales al sistema y analizar sus respectivas salidas. Hoy en día existen una gran variedad de bases de datos de audio específicas que son utilizadas en numerosos proyectos de procesado de audio.

La base de datos MIR-1K [19] utilizada está formada por 1000 extractos de canción de duración entre 4 y 13 segundos extraída de 110 canciones pop de karaoke chino realizado mayormente por amateurs. La frecuencia de muestreo de cada extracto es de 16000 Hz. La música y la voz están grabadas en el canal izquierdo y derecho respectivamente. Esta base de datos ha sido utilizada en numerosos trabajos de separación [20, 21], y va a ser usada como base de datos en este trabajo. A la hora de escuchar los diferentes extractos de la base de datos se ha observado la aparición de voz en el canal izquierdo cuando sólo tendría que haber música pura. Por lo tanto, esto puede afectar a la hora de la evaluación en los parámetros de medida.

Otra alternativa con características similares a la base de datos MIR-1K sería Ikala [22] la cual, está compuesta por 252 extractos de 30 segundos muestreados de 206 canciones de iKala grabadas de forma que la música y la voz se encuentran en los canales izquierdo y derecho respectivamente.

3.2. Experimentos

Para el estudio del comportamiento de nuestra arquitectura base se han modificado diversos parámetros. En cuanto a la representación espectral como entrada a nuestra red se ha optado por usar 1024 puntos de la transformada de Fourier localizada con un solape del 25 %. Se ha establecido un tiempo de entrenamiento máximo de 400 épocas. Se ha realizado un barrido en número de neuronas y en número de capas ocultas

realizando las simulaciones con diferentes tipos de activación. El primer experimento se ha realizado con una capa oculta de tipo densa variando su número de neuronas en 128, 256, 512, 1000 y 2000. Además se ha realizado simulaciones con diferentes tipos de activación como ***relu***, ***sigmoid*** y **lineal** tanto en la capa oculta como en la capa extra añadida. El segundo experimento ha consistido en utilizar dos capas ocultas de tipo denso con las misma variaciones que en la primera simulación. En el tercer experimento se ha incrementado el número de capas ocultas densas a tres manteniendo los mismos parámetros a barrer. Finalmente, en el último experimento de capas densas se ha añadido una capa más, siendo cuatro capas ocultas densas en total realizando las mismas configuraciones. Posteriormente en los siguiente experimentos se han utilizado capas ocultas de tipo GRU. En el primer experimento se ha utilizado una capa de este tipo con diferentes números de neuronas realizando un barrido en 128, 256, 512 ,1000 y 2000. En el segundo experimento se ha incrementado el número de capas a dos con el mismo barrido de parámetros. En el tercer experimento hemos utilizado tres capas ocultas de tipo GRU con la misma configuración que en los ejemplos anteriores y función de activación de la capa extra **sigmoid**.

3.3. Medidas de prestaciones

Para evaluar las prestaciones de la calidad de separación de fuente de los algoritmos, se suelen usar comúnmente una serie de métricas definidas por Vicent et en [4]. Estas métricas se basan en dada una señal estimada $\widehat{s}(t)$ de una fuente $s_i(t)$ que se puede expresear como suma de cuatro componentes:

$$\widehat{s}(t) = s_{target} + e_{interf} + e_{noise} + e_{artif} \tag{3.1}$$

donde s_{target} es una la señal de la fuente $s_i(t)$ con una permitida distorsión, e_{interf}, e_{noise} y e_{artif} son los errores producidos por interferencias de otras fuentes, del ruido y los algoritmos de separación de la señal respectivamente.

A partir de ésta descomposición de $s_i(t)$ se definen tres parámetros computando los niveles de energía de cada uno de ellos expresados en decibelios (dB):

Source to Distorsion Ratio (SDR) calcula cuánta distorsión existe en el sonido predicho. Un valor más alto de SDR indica que hay menos ruidos de distorsión en general, que incluyen la interferencia, ruido y errores de artefactos.

$$SDR := 10 \log \frac{\|s_{target}\|^2}{\|e_{interf} + e_{noise} + e_{artif}\|^2} \tag{3.2}$$

Source to Interferences Ratio (SIR) compara directamente cómo de bien, los ruidos procedentes de sonidos fuentes han sido separados de s_{target} Un valor más alto de SIR indica que el sonido de la señal de la fuente se distingue bien de los ruidos de interferencia que deseamos separar.

$$SIR := 10 \log \frac{\|s_{target}\|^2}{\|e_{interf}\|^2} \tag{3.3}$$

Source to Artifacts Ratio (SAR) calcula el efecto de los errores de artefactos en la salida de sonido predicha. Los artefactos incluyen distorsiones de las fuentes, o artefactos "burbling". Los valores más altos de SAR indican que los efectos de los errores de artefactos son mínimos.

$$SAR := 10 \log \frac{\|s_{target} + e_{interf} + e_{noise}\|^2}{\|e_{artif}\|^2} \tag{3.4}$$

En nuestros experimentos utilizaremos el toolbox específico para este tipo de medidas [23]. Cada uno de estos parámetros serán funciones que como argumentos de entrada tendrán señal estimada y la original, en nuestro caso la señales serán las voces y como salida los parámetros descritos anteriormente.

Además realizaremos una normalización del SDR (NSDR) definido como:

$$NSDR(\widehat{V}, V, X) = SDR(\widehat{V}, V) - SDR(X, V) \tag{3.5}$$

Donde $\widehat{V}$, V y X son respectivamente, la voz estimada, la voz original y la mezcla. Como en nuestro caso vamos a trabajar con varias tramas de pequeño tamaño, globalizamos los parámetros sumando los parámetros nombrados anteriormente obtenidos en cada una de las tramas multiplicados por sus respectivas longitudes dividido entre la longitud total obteniendo así los términos GNSDR, GSIR y GSAR. Cuanto mayor sean estos parámetros de medida mayor será la calidad de la separación.

3.4. Herramientas utilizadas

Google Colaboratory: es una herramienta de investigación para la educación y la investigación de aprendizaje automático desarrollada por Google. Se trata de un entorno de portátil Jupyter que no requiere configuración para su uso en programas básicos. Ésta herramienta nos proporciona como hardware, la tarjeta gráfica Tesla K80 como GPU y un Intel Xeon como CPU para poder realizar nuestros experimentos de formas más eficiente. Además tiene la ventaja de que se pueden ejecutar comandos de Linux por lo que no es necesario escribir todo el código en las notas Jupyter. Sin embargo, tiene como desventaja, la limitación del tiempo de cómputo máximo

seguido de 12 horas. En este trabajo se ha programado una nota júpiter con comandos Linux de manera que accediendo a Google Drive, podamos acceder a nuestro proyecto programado en Python y así poder realizar los experimentos para posteriormente, guardar los resultados en Google Drive.

Tensorflow: es una biblioteca de aprendizaje automático de código abierto para investigación y producción de aprendizaje automático desarrollado por Google TensorFlow. Ofrece API para principiantes y expertos para desarrollar en computadoras de escritorio, móviles, web y en la nube. Esta biblioteca además es compatible con Google Colaboratory, permitiendo trabajar con el hardware ofrecido por esta aplicación.

Cluster de computación de voz Vivolab: es la infraestructura científica utilizada por los investigadores de Vivolab en el área de procesado de señal debido a la gran capacidad de cómputo que éste posee. En este trabajo se ha accedido a él para poder realizar un mayor número de simulaciones sin la limitación de 12 horas que Google Colaboratory impone.

Python: Python es un lenguaje de programación interpretado, orientado a objetos y de alto nivel con semántica dinámica. Sus estructuras de datos integradas de alto nivel, combinadas con la tipificación dinámica y el enlace dinámico, lo hacen muy atractivo para el desarrollo rápido de aplicaciones, así como para su uso como lenguaje de scripting o pegamento para conectar componentes existentes entre sí. La sintaxis simple y fácil de aprender de Python hace hincapié en la legibilidad y, por lo tanto, reduce el costo del mantenimiento del programa. Python admite módulos y paquetes, lo que fomenta la modularidad del programa y la reutilización del código [24]. En este trabajo se ha utilizado en la parte de las redes neuronales debido a su extendido uso en esta área y a la gran compatibilidad con Tensorflow y Google Colaboratory. En concreto, para modelar las diferentes configuraciones se ha utilizado y modificado un programa base [25] programado en este lenguaje.

Matrix Laboratory: también conocido como MATLAB [26], es una herramienta de software matemático que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio llamado M. Entre sus prestaciones básicas se hallan: la manipulación de matrices, la representación de datos y funciones, la implementación de algoritmos, la creación de interfaces de usuario (GUI) y la comunicación con programas en otros lenguajes y con otros dispositivos hardware. Se ha elegido esta herramienta a

la hora de estudiar los algoritmos de procesado de señal clásico debido a su familiaridad de uso durante todo el grado.

Capítulo 4

Resultados

Utilizando la base de datos MIR-1K en los diferentes métodos de separación de fuente explicados en apartados anteriores y evaluándolos con las métricas propuestas en este trabajo se han obtenido y evaluado los siguientes resultados.

4.1. Robust Principal Component Análisis

En este experimento se ha aplicado el algoritmo RPCA a cada una de las canciones extraídas de la base de datos MIR-1K obteniendo los diferentes valores NSDR, SIR, y SAR para posteriormente mediante un promediado calcular las métricas globales GNSDR, GSIR y GSAR. Además se ha realizado dos distinciones: en la primera se ha aplicado el algoritmo sin ninguna mascara binaría mientras que en el segundo se ha optado por dicha máscara. Con el fin de obtener una referencia objetiva sobre los valores de las métricas, se ha realizado una lectura de diferentes proyectos que utilizan las mismas métricas de evaluación y se ha llegado a la conclusión de que unos valores aceptables de GSIR y GSAR superarían los 5 dB aproximadamente mientras que los valores de GNSDR suelen estar por encima de los 2 dB dependiendo del algortimo utilizado [27].

Los valores finales obtenidos al aplicar Robust Principal Component Análisis vienen reflejados en la figura 4.2. En ella podemos observar un gráfico de barras dividido en dos secciones. En la sección de la izquierda representa las métricas evaluadas del algoritmo RPCA utilizando una máscara binaria y en la sección de la derecha, las mismas métricas pero prescindiendo de ésta.

En el caso de no utilización de máscara binaria observamos un valor de GSAR bastante más alto en comparación con el resultado obtenido aplicando la máscara y con respecto a los valores GNSDR y GSIR. Esto puede ser debido a que los errores producidos por los algoritmos de procesado son bastante bajos y a la hora de utilizar la máscara binaria puede producir una mayor tasa de error mayor produciéndose una

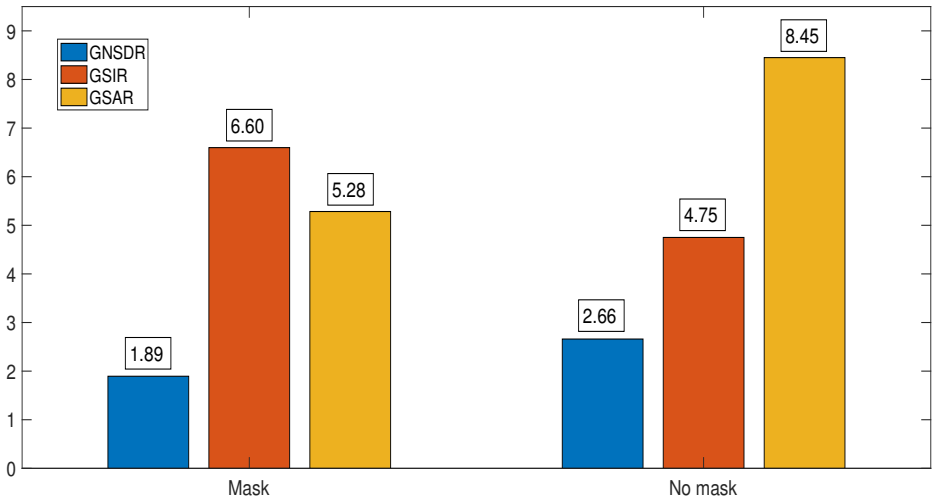


Figura 4.1: Resultados obtenidos tras RPCA

disminución del valor de dicho parámetro.

El parámetro GSIR posee un valor más alto a la hora de la utilización de la máscara. Esta máscara puede favorecer el aislamiento entre frecuencias y por lo tanto un aumento del GSIR. Sin embargo, se puede apreciar cómo el valor de GNSDR siendo el parámetro que más refleja la calidad de la separación, es más alto a la hora de prescindir de la máscara binaria. Aún así, nos aporta un valor de GNSDR bastante bajo en comparación con algunos experimentos posteriores.

Subjetivamente, al escuchar la voz y la música por separado puede apreciarse que se ha separado parte de la voz quedando aún parte de ésta en la parte instrumental. Esta apreciación puede también observase en la figura 4.1 en la parte de la derecha como en frecuencias bajas consigue un mayor aislamiento de la voz, pero a medida que subimos en frecuencia observamos zonas con menor intensidad de señal que en la voz original lo que se traduce en un peor aislamiento.

4.2. REpeating Pattern Extraction Technique

En este experimento se ha aplicado el algoritmo REPET a cada una de las canciones extraídas de la base de datos MIR-1K obteniendo los diferentes valores NSDR, SIR, y SAR para posteriormente mediante un promediado calcular las métricas globales GNSDR, GSIR y GSAR.

La tabla 4.1 muestra los resultados obtenidos al utilizar éste método donde se puede apreciar unos niveles bajos de los tres componentes de medida. Si lo comparamos con el método RPCA, se puede observar cómo se obtienen unos valores parecidos en GNSDR

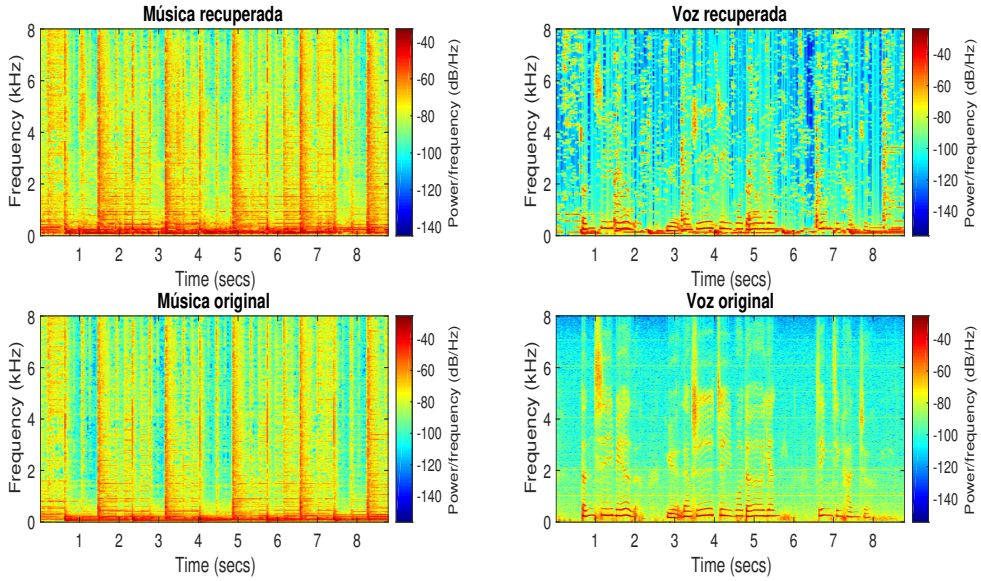


Figura 4.2: Espectrograma de los resultados obtenidos usando RPCA comparándolos con los originales

pero valores más bajos de GSIR y GSAR. Esta disminución, puede deberse a que en la fase 2 del algoritmo, se elimina parte de la información al realizar la réplica dando lugar a un mayor error tanto en la parte de la componente de e_{interf} y e_{artif} nombrados en la ecuación 3.1 lo que produce un GSIR y un GSAR más pequeños. Es decir, un menor aislamiento. Si observamos la figura 4.3 de los espectrogramas de la música original y la música separada, podríamos encontrar que hay más elementos de alta frecuencia en la música separada que en la música original debido que la voz en mezcla de audio no pudo ser eliminada por completo. A la hora de analizar subjetivamente, al escuchar la voz y la música por separado puede apreciarse que se ha separado parte de la voz quedando aún parte de ésta en la parte instrumental siendo un resultado similar al conseguido con RPCA.

GNSDR	GSIR	GSAR
2.49	2.32	2.48

Tabla 4.1: Resultados obtenidos tras aplicar REPET.

4.3. Redes neuronales

En la realización de estos experimentos se han seleccionado 175 extractos de la base de datos MIR-1K para el entrenamiento de la red. Concretamente, los extractos 'amy' y 'abjones' cantados por un hombre y una mujer respectivamente. Como test se han utilizado las 825 restantes. Se ha dedicado una mayor parte de datos al test con la

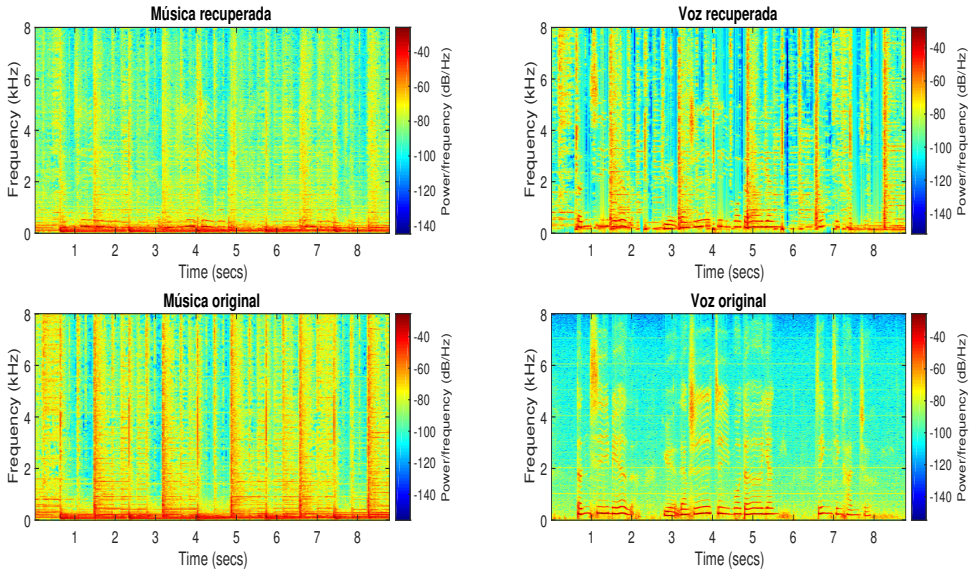


Figura 4.3: Espectrograma de los resultados obtenidos usando REPET comparándolos con los originales

finalidad de dar una mayor validez estadística a los resultados de acuerdo a [27]. La duración de cada experimento contando entrenamiento mas evaluación ha sido de 3 horas de media y se han realizado un total de 120 experimentos dando un total de 360 horas.

4.3.1. Una capa densa

Como se puede observar en la tabla 4.2 con la configuración de 128 neuronas obtenemos unos parámetros de GNSDR y GSIR bastante bajos. De hecho, menor que en los modelos estudiados previamente. En cambio, obtenemos unos valores de GSAR bastante altos lo que se traducen en una menor interferencia de artefactos. Entre las diferentes combinaciones de activación de las capas observamos un mayor comportamiento escogiendo la función de activación tanto en la capa oculta como en la capa de la máscara tiempo frecuencia como **sigmoid**. Conforme vamos aumentando el número de neuronas por capa observamos un incremento de los valores GNSDR y GSIR, especialmente vemos un cambio muy abrupto cuando pasamos de 128 a 256 neuronas. A partir de 1000 neuronas obtenemos el mayor valor posible con esta configuración siendo considerablemente superior a los métodos estudiados previamente.

4.3.2. Dos capas densas

Al aumentar el número de capas ocultas a dos observamos en la tabla 4.3 un aumento considerable de GNSDR Y GSIR en 4 y 8 dB respectivamente, con la

Tabla 4.2: Resultados de GNSDR, GSIR y GSAR obtenidos en la configuración de 1 capa densa D(n,m,t,v) donde D se refiere a capa densa, n es el número de capas ocultas, m el número de neuronas, t activación de la capa oculta y v activación de la capa extra

	D(1,128,L,R)	D(1,128,L,S)	D(1,128,S,R)	D(1,128,S,S)	D(1,128,R,R)	D(1,128,R,S)
GNSDR	-0.4	1.12	-0.445	**1.23**	-0.76	1.17
GSIR	0.65	1.23	0.44	**1.35**	0.2	1.28
GSAR	10.24	11.28	11.01	**11.32**	9.97	10.32
	D(1,256,L,R)	D(1,256,L,S)	D(1,256,S,R)	D(1,256,S,S)	D(1,256,R,R)	D(1,256,R,S)
GNSDR	4.55	5.72	3.55	**5.93**	3.98	5.85
GSIR	8.51	11.17	5.71	**10.45**	7.39	11.40
GSAR	8.48	8.52	9.53	**9.37**	7.89	8.60
	D(1,512,L,R)	D(1,512,L,S)	D(1,512,S,R)	D(1,512,S,S)	D(1,512,R,R)	D(1,512,R,S)
GNSDR	4.56	5.50	4.17	**5.95**	4.61	5.93
GSIR	8.44	11.05	6.50	**10.83**	7.76	11.63
GSAR	8.29	8.27	10.04	**9.02**	9.22	8.58
	D(1,1000,L,R)	D(1,1000,L,S)	D(1,1000,S,R)	D(1,1000,S,S)	D(1,1000,R,R)	D(1,1000,R,S)
GNSDR	4.62	5.53	3.46	**6.07**	4.27	5.91
GSIR	8.67	11.19	5.49	**10.87**	6.99	11.73
GSAR	8.38	8.17	9.62	**9.16**	9.47	8.45
	D(1,2000,L,R)	D(1,2000,L,S)	D(1,2000,S,R)	D(1,2000,S,S)	D(1,2000,R,R)	D(1,2000,R,S)
GNSDR	4.59	5.44	3.40	**5.92**	4.23	5.89
GSIR	8.53	10.50	5.39	**9.99**	6.82	11.28
GSAR	8.21	8.16	9.40	**9.10**	9.18	8.29

configuración de 128 neuronas en una sola capa. A diferencia de la configuración anterior observamos que los mayores resultados se dan escogiendo la capa oculta como **relu** y la capa de la frecuencia tiempo frecuencia como **sigmoid**. Sin embargo, las prestaciones con la configuración de **sigmoid** en capa oculta y **relu** empeoran drásticamente. Conforme aumentamos el número de neuronas de las dos capas vemos un incremento de los parámetros GNSDR y GSIR hasta llegar a 1000 neuronas donde si duplicamos el numero de neuronas de las dos capas empezamos a decrementar las prestaciones. En cuanto al GSAR observamos una disminución de aproximadamente 2 dB con respecto a la configuración anterior y vemos también cómo se va reduciendo conforme aumenta el número de capas.

4.3.3. Tres capas densas

Con tres capas ocultas obtenemos el mejor resultado con la configuración de capas ocultas lineales y capa extra **sigmoid** con 128 neuronas como se muestra en la tabla 4.4. Sin embargo para el resto de configuraciones con 128 neuronas se produce un decremento leve de las prestaciones del sistema. En cuanto incrementamos el número de neuronas por capa oculta mejora levemente las prestaciones para la mayoría de las configuraciones. Concretamente obtenemos el mejor resultado hasta ahora de GNSDR, GSIR y GSAR de 6.11dB, 12.30dB y 8.45dB respectivamente con 1000 neuronas.

Tabla 4.3: Resultados de GNSDR, GSIR y GSAR obtenidos en la configuración de 2 capas densas D(n,m,t,v) donde D se refiere a capa densa, n es el número de capas ocultas, m el número de neuronas, t activación de la capa oculta y v activación de la capa extra.

	D(2,128,L,R)	**D(2,128,L,S)**	**D(2,128,S,R)**	**D(2,128,S,S)**	**D(2,128,R,R)**	**D(2,128,R,S)**
GNSDR	4.70	5.32	2.64	4.93	4.78	**5.42**
GSIR	8.92	10.7	3.55	8.62	8.15	**10.77**
GSAR	8.31	8.19	12.72	9.06	8.97	**8.33**
	D(2,256,L,R)	**D(2,256,L,S)**	**D(2,256,S,R)**	D(2,256,S,S)	**D(2,256,R,R)**	**D(2,256,R,S)**
GNSDR	4.47	5.66	1.79	5.08	4.37	**5.78**
GSIR	8.46	11.25	2.50	8.61	7.63	**11.88**
GSAR	8.32	8.36	12.40	9.43	8.90	**8.19**
	D(2,512,L,R)	**D(2,512,L,S)**	**D(2,512,S,R)**	D(2,512,S,S)	**D(2,512,R,R)**	**D(2,512,R,S)**
GNSDR	4.45	5.22	1.58	5.41	3.76	**5.69**
GSIR	8.58	11.24	2.23	9.52	6.51	**11.61**
GSAR	8.13	7.81	11.98	9.10	8.90	**8.19**
	D(2,1000,L,R)	**D(2,1000,L,S)**	**D(2,1000,S,R)**	D(2,1000,S,S)	**D(2,1000,R,R)**	**D(2,1000,R,S)**
GNSDR	4.28	5.50	0.03	5.65	3.92	**6.10**
GSIR	8.67	11.20	0.82	10.45	6.51	**12.23**
GSAR	7.80	8.12	11.76	8.83	9.22	**8.45**
	D(2,2000,L,R)	**D(2,2000,L,S)**	**D(2,2000,S,R)**	D(2,2000,S,S)	**D(2,2000,R,R)**	**D(2,2000,R,S)**
GNSDR	4.17	4.38	0.10	5.52	3.75	**5.99**
GSIR	8.47	10.80	0.48	9.54	6.30	**11.55**
GSAR	7.59	8.10	12.35	8.04	9.01	**8.60**

Tabla 4.4: Resultados de GNSDR, GSIR y GSAR obtenidos en la configuración de 3 capas densas D(n,m,t,v) donde D se refiere a capa densa, n es el número de capas ocultas, m el número de neuronas, t activación de la capa oculta y v activación de la capa extra

	D(3,128,L,R)	**D(3,128,L,S)**	**D(3,128,S,R)**	**D(3,128,S,S)**	**D(3,128,R,R)**	**D(3,128,R,S)**
GNSDR	4.71	5.61	-1.53	3.84	4.57	**5.38**
GSIR	8.72	10.74	-0.68	6.65	8.13	**11.10**
GSAR	8.42	8.61	10.68	9.17	8.85	**7.99**
	D(3,256,L,R)	**D(3,256,L,S)**	**D(3,256,S,R)**	D(3,256,S,S)	**D(3,256,R,R)**	**D(3,256,R,S)**
GNSDR	4.74	5.54	1.64	1.95	4.66	**5.88**
GSIR	9.32	11.22	2.54	3.12	8.01	**11.72**
GSAR	8.70	8.29	13.43	12.47	8.91	**8.36**
	D(3,512,L,R)	**D(3,512,L,S)**	**D(3,512,S,R)**	D(3,512,S,S)	**D(3,512,R,R)**	**D(3,512,R,S)**
GNSDR	4.66	4.65	1.39	5.39	4.88	**5.89**
GSIR	9.19	9.19	2.18	9.62	8.50	**11.90**
GSAR	8.06	8.06	13.61	9.53	8.90	**8.35**
	D(3,1000,L,R)	**D(3,1000,L,S)**	**D(3,1000,S,R)**	D(3,1000,S,S)	**D(3,1000,R,R)**	**D(3,1000,R,S)**
GNSDR	4.80	5.52	1.41	5.75	4.73	**6.11**
GSIR	10.60	11.35	2.20	11.49	8.43	**12.30**
GSAR	8.15	8.16	13.58	8.38	8.37	**8.45**
	D(3,2000,L,R)	**D(3,2000,L,S)**	**D(3,2000,S,R)**	D(3,2000,S,S)	**D(3,2000,R,R)**	**D(3,2000,R,S)**
GNSDR	4.72	5.22	1.38	5.54	4.53	**5.95**
GSIR	10.02	11.28	2.13	10.45	8.75	**11.87**
GSAR	8.09	7.97	12.76	8.69	8.11	**8.32**

4.3.4. Cuatro capas densas

Tras añadir una última capa de tipo densa a la red observamos en la tabla 4.5 un empeoramiento con respecto a la configuración de tres capas ocultas de manera general. Seguimos obteniendo las mayores métricas utilizando funciones de activación de **relu** en las capas ocultas y **sigmoid** en la capa extra produciéndose la óptima con 1000 neuronas. Además la evolución de los parámetros GNSDR,GSIR y GSAR siguen aproximadamente la misma evolución que en las configuraciones anteriores.

Tabla 4.5: Resultados de GNSDR, GSIR y GSAR obtenidos en la configuración de 4 capas densas D(n,m,t,v) donde D se refiere a capa densa, n es el número de capas ocultas, m el número de neuronas, t activación de la capa oculta y v activación de la capa extra

	D(4,128,L,R)	D(4,128,L,S)	D(4,128,S,R)	D(4,128,S,S)	D(4,128,R,R)	D(4,128,R,S)
GNSDR	4.46	3.72	3.87	4.51	4.53	**5.57**
GSIR	8.39	6.46	6.75	8.19	8.24	**11.46**
GSAR	8.30	9.37	9.28	8.79	8.74	**8.05**
	D(4,256,L,R)	D(4,256,L,S)	D(4,256,S,R)	D(4,256,S,S)	D(4,256,R,R)	D(4,256,R,S)
GNSDR	4.71	4.39	1.53	4.75	4.58	**5.87**
GSIR	9.01	8.28	2.44	9.49	8.09	**11.93**
GSAR	8.58	8.15	13.49	8.38	8.93	**8.33**
	D(4,512,L,R)	D(4,512,L,S)	D(4,512,S,R)	D(4,512,S,S)	D(4,512,R,R)	D(4,512,R,S)
GNSDR	4.60	5.60	1.29	4.25	4.69	**5.69**
GSIR	9.22	9.32	2.13	9.38	8.52	**11.12**
GSAR	8.15	8.12	12.76	8.21	8.85	**8.04**
	D(4,1000,L,R)	D(4,1000,L,S)	D(4,1000,S,R)	D(4,1000,S,S)	D(4,1000,R,R)	D(4,1000,R,S)
GNSDR	4.68	5.48	2.63	5.43	4.34	**5.79**
GSIR	9.92	10.17	1.47	10.24	8.12	**12.29**
GSAR	7.99	8.29	10.37	8.52	8.09	**8.37**
	D(4,2000,L,R)	D(4,2000,L,S)	D(4,2000,S,R)	D(4,2000,S,S)	D(4,2000,R,R)	D(4,2000,R,S)
GNSDR	4.56	5.37	1.38	4.20	4.22	**5.93**
GSIR	9.44	11.28	2.10	9.73	8.10	**11.66**
GSAR	7.92	7.97	11.36	8.15	7.97	**8.31**

4.3.5. Redes GRU

Los resultados obtenidos pueden verse reflejados en la tabla 4.6. En ella observamos como con una sola capa oculta obtenemos unos valores de GNSDR 6.20 y 6.40 usando 512 y 1000 neuronas respectivamente consiguiendo el mejor resultado en este trabajo además de obtenerse unos valores de GSIR y GSAR bastante elevados. Ésto corresponde teóricamente a lo estudiado ya que como hemos mencionado en apartados anteriores las neuronas de tipo GRU poseen información de partes anteriores de la canción que pueden ser relevantes a la hora de la identificación y separación de la voz. Esta gran mejora con respecto a los métodos de procesado clásico puede observarse en la figura 4.4. En ella puede apreciarse un decremento de intensidad del espectro (azul) lo que se traduce en un mayor aislamiento aunque en zonas como en el segundo dos aparece una componente de frecuencias en en la voz original no estaba. A la hora de escuchar está canción se produce una pequeña distorsión en la parte mencionada anteriormente.

No obstante, conforme vamos aumentando el número de capas vamos reduciendo de manera muy reducida los valores de GNSDR, GSIR y GSAR siendo éstos aún bastante aceptables.

Tabla 4.6: Resultados de GNSDR, GSIR y GSAR obtenidos en la configuración de G(n,m,v) donde G se refiere a capa GRU, n es el número de capas ocultas, m el número de neuronas, y v activación de la capa extra

	G(1,128,S)	G(1,256,S)	G(1,512,S)	G(1,1000,S)	G(1,2000,S)
GNSDR	5.57	5.93	6.20	**6.40**	6.21
GSIR	10.51	11.15	11.90	**11.58**	11.60
GSAR	9.07	8.82	8.80	**8.90**	8.70
	G(2,128,S)	**G(2,256,S)**	**G(2,512,S)**	**G(2,1000,S)**	**G(2,2000,S)**
GNSDR	5.96	5.84	5.90	5.72	5.62
GSIR	11.11	11.63	12.05	11.58	11.78
GSAR	8.81	8.43	8.27	8.50	8.14
	G(3,128,S)	**G(3,256,S)**	**G(3,512,S)**	**G(3,1000,S)**	**G(3,2000,S)**
GNSDR	4.74	5.63	5.77	5.65	5.53
GSIR	8.86	11.28	11.96	11.49	11.32
GSAR	8.45	8.40	8.30	8.38	8.22

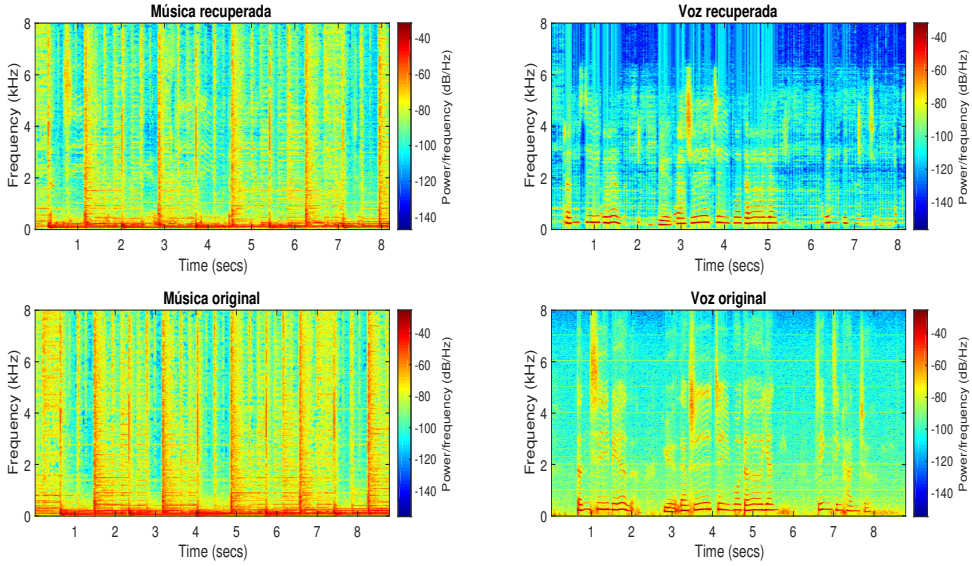


Figura 4.4: Espectrograma de los resultados obtenidos usando redes neuronales con la mejor configuración GRU con 1 capa y 1000 neuronas comparándolos con los originales

Capítulo 5

Conclusión y líneas futuras

En este trabajo se ha realizado un estudio detallado de diferentes métodos de separación de fuente, concretamente la separación de la voz de una canción en grabación mono.

El algoritmo RPCA utiliza la descomposición matricial en una matriz de rango bajo y una matriz dispersa aprovechando donde en la primera matriz se encontraría la parte instrumental y en la segunda la parte vocal. Los valores obtenidos por las métricas son bajos en comparación con otros métodos utilizados pero tiene como ventaja la rápida convergencia del algoritmo y realización de la separación de la canción.

El algoritmo REPET depende de la identificación del patrón de repetición en la música para separar la voz del cantante de la música. En consecuencia, este algoritmo es altamente sensible al periodo de repetición de la música. Identificar el patrón de repetición preciso es el núcleo del algoritmo. Mientras podamos obtener el periodo de repetición de una mezcla, podríamos filtrar efectivamente la voz cantante de la mezcla de audio. La desventaja de este algoritmo es que este algoritmo todavía asigna algo de música en señal de voz separada debido a que solo las partes que tienen una alta repetición de patrones de la música se separan. Sin embargo, aunque este algoritmo no es capaz de extraer perfectamente la voz original, ésta es todavía audible.

En cuanto a la utilización de redes neuronales se ha demostrado la gran potencia que tienen en el ámbito de separación de fuentes produciendo unos resultados considerablemente más altos que los algoritmos de procesado de señal clásico. Sin embargo a la hora de realizar y estudiar las simulaciones con diferentes tipos de configuraciones se requiere una gran cantidad de tiempo además de un hardware adecuando que acelere el proceso de entrenamiento de los experimentos y una gran cantidad de datos para entrenar. Además como hemos observado en nuestro trabajo, en algunas ocasiones es difícil de interpretar los comportamientos producidos por dicha red. Una forma de solventar este problema podría ser incrementar la cantidad de datos de entrenamiento.

Vistos los resultados obtenidos en este trabajo, existen numerosas posibilidades a la hora de aprovechar este trabajo para ampliar este proyecto en un trabajo final de máster o doctorado.

Debido a la falta de base de datos, no se ha podido realizar un enfoque de separación de audio a un tipo de música determinado. Concretamente se investigaría o se generaría una base de datos especializada en música electrónica para poder trabajar con ese tipo de música.

La primera ampliación consistiría en conseguir unos valores de GNSDR, GSIR y GSAR bastante más elevados en comparación a los de este trabajo con el fin de obtener una calidad que pueda utilizarse a la hora de manipular audio de estudio.

Una posibilidad sería modificar la implementación de los algoritmos base RPCA y REPET con el fin de obtener una mejora de los valores obtenidos en este trabajo. Además se podría investigar sobre otros métodos de procesamiento de señal clásico. Un ejemplo sería intentar separar la parte harmónica de la parte percusiva de una canción y una vez obtenidas las señales separadas, realizar la separación de la parte vocal e instrumental tanto de la parte harmónica y parte percusiva para posteriormente sumar la parte vocal harmónica con la parte vocal percusiva y la parte instrumental percusiva con la parte instrumental harmónica

En cuanto al área de mayor mejora sería utilizando nuevas configuraciones de redes neuronales. Esto barrería desde el incremento considerable del numero de neuronas y capas ocultas. Además de las redes neuronales propuestas en este proyecto existen otro tipo de redes que funcionan muy bien en proyectos de estas características como por ejemplo las redes recurrentes bilineales, redes Long Short Term Memory (LSTM) o redes convolucionales.

Finalmente una vez obtenida una calidad de separación suficientemente alta como para trabajar en estudios de música se procedería a la implementación de un efecto de remezcla que añadiera o quitara progresivamente la voz de una canción en tiempo real para en un futuro implementarlo físicamente en algún dispositivo electrónico de música.

Capítulo 6

Bibliografía

[1] Arthur H. Benade. *Fundamentals of Musical Acoustics*. Dover Books on Music, 1976.

[2] A. S. Bregman. *Auditory scene analysis.* MIT Press: Cambridge, MA, 1976.

[3] D. L. Wang and G. J. Brown. Computational auditory scene analysis: Principles, algorithms and applications. *IEEE Press/Wiley-Interscience*, 2006.

[4] Emmanuel Vicent, Rémi Gribonval, and Cédric Févotte. Performance measurement in blind audio source separation. *IEEE Transactions on Audio, Speech and Language Processing, Institute of Electrical and Electronics Engineers pp.1462-1469*, 2006.

[5] P.B. Mane. Separation of singing voice from music background. *International Journal of Computer Applications pp.975-8887*, 2015.

[6] N. Sonntag. *Proakis*. McGraw-Hill Education, 2016.

[7] Julius O. Smith III. Spectral audio signal processing. *ISBN 978-0-9745607-3-1*, 2011.

[8] Po-Sen Huang, Scott Deeann Chen, Paris Smaragdis, and Mark Hasegawa-Johnson. Singing-voice separation from monaural recordings using robust principal component analysis. *University of Illinois at Urbana-Champaign Department of Electrical and Computer Engineering 405 North Mathews Avenue, Urbana, IL 61801 USA*, 2012.

[9] Zafar Rafii, Antoine Liutkus, and Bryan Pardo. Repet for background/foreground separation in audio. *Blind Source Separation, Springer, Berlin, Heidelberg*, 2014.

[10] Zafar RAFII and Bryan PARDO. A simple music/voice separation method based on the extraction of the repeating musical structure. *36th International Conference on Acoustics, Speech and Signal Processing*, 2011.

[11] Georgi Mladenov Valeri. Gevaert, Wouter Tsenov. Neural networks used for speech recognition. *Journal of Automatic Control. 20. 10.2298/JAC1001001G.*, 2010.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *Microsoft Research*, 2010.

[13] Yoon Kim. Deep residual learning for image recognition. *New York University*, 2014.

[14] Stanford University. Cs231n convolutional neural networks for visual recognition. http://cs231n.github.io/neural-networks-1/, citado Noviembre 2018.

[15] Ron Kohavi; Foster Provost. "glossary of terms"machine learning. 1998.

[16] Luis Federico Vertona. Entrenamiento de redes neuronales basado en algoritmos evolutivos. Tesis doctoral, Universidad de Buenos Aires, 2004.

[17] Universidad Carlos III Madrid. Introducción a la redes neuronales aplicadas. http://halweb.uc3m.es/esp/Personal/personas/jmmarin/esp/DM/tema3dm.pdf, citado Noviembre 2018.

[18] Anish Singh Walia. El problema del desvanecimiento de gradiente. https://medium.com/@anishsingh20/the-vanishing-gradient-problem-48ae7f501257, citado Noviembre 2018.

[19] Chao-Ling Hsu and Prof. Jyh-Shing Roger Jang. Base de datos mir-1k. https://sites.google.com/site/unvoicedsoundseparation/mir-1k, citado Noviembre 2018.

[20] Dominique Fourer and Geoffroy Peeters. Single-channel blind source separation for singing voice detection: A comparative study. 2018.

[21] Dominique Fourer and Geoffroy Peeters. Deep learning for monaural source separation. 2018.

[22] Tak-Shing Chan, Tzu-Chun Yeh, Zhe-Cheng Fan, Hung-Wei Chen, Li Su, Yi-Hsuan Yang, and Roger Jang. Vocal activity informed singing voice separation

with the ikala dataset. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015.

[23] Chao-Ling Hsu and Prof. Jyh-Shing Roger Jang. Toolbox bss-eval. http://bass-db.gforge.inria.fr/bss$_e$*val*, *citadoNoviembre*2018.

[24] Guido van Rossum. Lenguaje de programación python. https://www.python.org/doc/essays/blurb/, citado Noviembre 2018.

[25] Dabi Ahn. Deep neural network for music source separation in tensorflow. https://github.com/andabi/music-source-separation/blob/master/README.md, citado Noviembre 2018.

[26] Cleve Moler. Lenguaje de programación matlab. https://es.mathworks.com/, citado Noviembre 2018.

[27] Po-Sen Huang, Minje Kim, Mark Hasegawa-Johnson, and Paris Smaragdis. Singing-voice separation from monaural recordings using deep recurrent neural networks. *IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 23, no. 12, pp. 2136–2147*, 2014.

[28] S. Z. Ramírez, K. Pérez. Self conscious robots in induction heating home appliances. *IEEE transactions on anthropomorphic robots*, 2018.

Lista de abreviaturas

RPCA Robust principal Component Analysis

REPET REpeating Pattern Extraction Technique

ASA Auditive Scene Analysis

CASA Computacional Auditive Scene Analysis

BSS Blind Source Separation

STFT Short Time Fourier Transform

ISTFT Inverse Short Time Fourier Transform

FFT Fast Fourier Transform

ALM Augmented Lagrange Multiplier

SDR Signal Distorsion Ratio

NSDR Normalized Signal Distorsion Ratio

GNSDR Global Normalized Signal Distorsion Ratio

SIR Source Interference Ratio

GSIR Global Source Interference Ratio

SAR Source Artifacts Ratio

GSAR Global Source Artifacts Ratio

Lista de Figuras

Lista de Tablas