Universidad del País Vasco Euskal Herriko Unibertsitatea

BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

**MÁSTER UNIVERSITARIO EN**

**INGENIERÍA DE TELECOMUNICACIÓN**

# TRABAJO FIN DE MÁSTER

## ADAPTACIÓN DEL SISTEMA DE CONVERSIÓN DE TEXTO A VOZ EXTREMO A EXTREMO TACOTRON A OTRAS LENGUAS

**Alumno/Alumna** *García Romillo, Víctor*

**Director/Directora** *Navas Cordón, Eva*

**Departamento** Ingeniería de Comunicaciones

**Curso académico** *2018/2019*

*Bilbao, 12 de Junio de 2019*

## Resumen

[ES] La síntesis de voz ha alcanzado unos niveles de calidad muy elevados en materia de inteligibilidad y naturalidad. Es una herramienta de comunicación muy útil y puede utilizarse para sustituir la voz de personas con dificultades de habla. Los enfoques basados en aprendizaje profundo han mejorado mucho la calidad de las voces sintéticas, pero requieren de grandes bases de datos de habla para su funcionamiento. La gran mayoría de proyectos que pueden encontrarse en la red son realizados para bases de datos en inglés. El presente proyecto emplea el sistema "Text-to-Speech" Tacotron-2 para sintetizar voces en otras lenguas, preservando las características de las mismas.

[EN] Synthetic speech has reached a very high quality in terms of intelligibility and naturalness. It is a very useful communication tool and it can be used to substitute the voice of people with oral impairments. Deep learning approaches have improved the quality of synthetic voices, but they require large speech databases to work. The majority of projects that can be found in the net have been developed using English databases. This project makes use of Tacotron-2, a text-to-speech system, in order to generate voices in other languages, preserving their characteristics.

[EU] Hizketa-sintesiak, ulergarritasun eta naturaltasun arloan, kalitate izugarrizko mailak lortu ditu. Komunikazio inguruan tresna lagungarri bat izan liteke, eta erabili ahal da hitz egiteko arazoak dituzten pertsonen ahotsa ordezteko. Ikaskuntza sakonean oinarritutako planteamenduek asko hobetu dute ahots sintetikoen kalitatea, baina hauek era egokian lan egiteko, hizketa datu-base oso handiak behar dituzte. Sarean aurkitu ahal diren proiektu gehienak ingelesez dauden datu-baseetarako garatu dira. Hemen erakusten den proiektuan "Text-to-Speech" Tacotron-2 sistema erabiltzen da ahotsak beste hizkuntz batzuetan sintetizatzeko, haien ezaugarriak mantenduz.

## Agradecimientos

Este trabajo de fin de Máster no hubiera sido posible sin la ayuda del grupo de investigación Aholab. Quisiera agradecer a mi directora Eva Navas por su paciencia y comprensión, así como por toda la orientación ofrecida a lo largo de este proyecto.

A mis familiares y amigos, quienes me brindaron su incondicional apoyo aún habiendoles obligado a escuchar cientos de audios generados en los experimentos.

Muchas gracias a todo el mundo que directa o indirectamente haya contribuido a que este proyecto salga adelante.

# Índice

# Listado de tablas

# Listado de Ilustraciones

## Listado de acrónimos

CNN: Convolutional Neural Network

DNN: Deep Neural Network

E2E: End-to-End

ISTFT: Inverse Short Time Fourier Transform

LSTM: Long Short-Term Memory

ML: Machine Learning

MOS: Mean Opinion Score

MSE: Mean Square Error

PCA: *Principal Component Analysis*

RNN: Recurrent Neural Network

seq2seq: Sequence to sequence

STFT: Short Time Fourier Transform

TTS: Text-to-Speech

# 1 Introducción

La conversión de texto en habla (*Text To Speech*, TTS) permite transformar automáticamente un texto en su correspondiente forma sonora de la manera más parecida posible a como lo haría un ser humano, es decir permiten general la señal de voz correspondiente a la lectura en voz alta del texto de entrada. La síntesis de voz generada a partir de texto es un tema estudiado desde hace décadas [1] dado su alto número de aplicaciones. Inicialmente empleada para la ayuda de personas con discapacidad visual, hoy en día pueden encontrarse sistemas TTS en cualquier dispositivo de ámbito diario, desde ordenadores a teléfonos inteligentes pasando por asistentes de voz, contestadores automáticos etc...

La creciente implantación de estos sistemas se debe a las mejoras realizadas en los últimos años sobre las tecnologías de síntesis. Estas mejoras abarcan principalmente dos ámbitos: la naturalidad y la inteligibilidad. La naturalidad se trata del grado de semejanza del audio generado con la voz humana, mientras que la inteligibilidad se relaciona con el grado de entendimiento del audio generado. El sintetizador perfecto es natural e inteligible al mismo tiempo.

Dentro de los avances tecnológicos conseguidos los últimos años destacan los producidos por los métodos de aprendizaje profundo, los cuales se han establecido como el nuevo paradigma en los diferentes campos de las tecnologías del habla. En el área de la síntesis de voz el uso de estas técnicas presenta mejoras en cuanto a naturalidad e inteligibilidad, de manera que su aplicación se ha extendido y se ha afianzando tanto en el ámbito de la investigación científica como en la explotación comercial.

A partir de 2017 grandes compañías como Baidu y Google [2] [3] comenzaron a trabajar en sistemas TTS basados en redes neuronales profundas (*Deep Neural Networks*, DNN). Una de las principales aportaciones de trabajar con redes neuronales

se trata de la posibilidad de crear sistemas extremo a extremo (*End-to-End*, E2E). Estos sistemas permiten convertir información de entrada, como por ejemplo texto, en una información de salida de otro tipo (o del mismo) como pudiera ser voz, definiendo lo que sería un sistema completo TTS que hace uso de un único sistema de aprendizaje neuronal.

Uno de los principales sistemas de conversión de texto a voz E2E es el desarrollado por Google, Tacotron-2. Los resultados obtenidos con esta tecnología son sobresalientes, sin embargo, han sido obtenidos únicamente utilizando una base de datos de voz (Corpus) en lengua inglesa. Por ello este proyecto pretende adaptar Tacotron a otras lenguas, como por ejemplo el castellano, considerando las particularidades del mismo, para poder disponer de una herramienta capaz de generar voz de alta calidad en lenguas distintas a la inglesa.

# 2 Contexto

El objetivo de la conversión de texto a voz es producir voces naturales capaces de expresarse en un estilo determinado y de reflejar el acento, estado de ánimo y otras características de los locutores humanos. Con este objetivo en mente, existen dos grandes grupos en los que se pueden categorizar las tecnologías para generar voz, la síntesis concatenativa y la síntesis estadístico-paramétrica. Ambas tecnologías presentan ventajas e inconvenientes; mientras la primera proporciona resultados más naturales, la segunda presenta un uso menor de memoria y además permite generar nuevas voces mediante interpolación y técnicas de adaptación.

Las tecnologías basadas en síntesis concatenativa hacen uso de grandes bases de datos de voz grabada, de las que se extraen unidades (palabras, difonos, trifonos...) que posteriormente se unen de la forma que mejor se adapte al texto objetivo que se quiere reproducir [4] como puede observarse en la Figura 1. Esta metodología es capaz de producir resultados muy naturales, pero para poder reproducir oraciones con una prosodia correcta es necesario almacenar una gran cantidad de grabaciones con distintas entonaciones, acentos y duraciones. Además, la elección de las unidades adecuadas para reproducir el texto deseado requiere de técnicas complejas [5] que exigen gran capacidad de procesado. Otro inconveniente que presenta la síntesis concatenativa está relacionado con los problemas de desajuste en las fronteras entre unidades concatenadas, como las diferencia de fase entre dos unidades empleadas o saltos grandes en la amplitud de ambas muestras, produciendo sonidos a modo de “click” que producen ruidos en la señal sintética [1] y degradan su calidad.

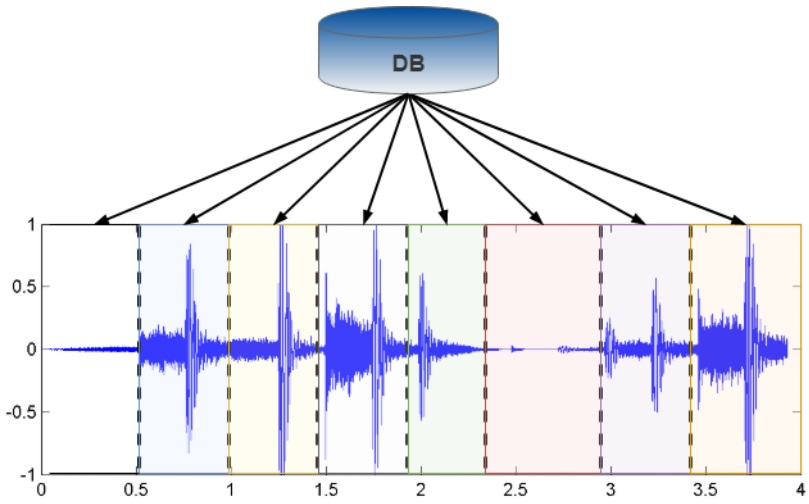



*Figura 1 Síntesis concatenativa de voz*

La síntesis estadístico-paramétrica se basa en la obtención un modelo matemático capaz de relacionar las características acústicas de la voz con aspectos fonético-lingüísticos extraíbles a partir del correspondiente texto. Inicialmente basadas en modelos ocultos de Markov [6] , las técnicas estadístico-paramétricas están tornando hacia el uso de redes neuronales profundas [7] , dado que permiten representar de una forma más acertada las relaciones no lineales entre los parámetros de la voz y la representación simbólica del habla. Se han probado distintas arquitecturas de red con el propósito de obtener voces de mejor calidad, desde redes de alimentación hacia delante (*feed-forward networks*) [8] a redes recurrentes [9] pasando por WaveNets [10] y siendo estas últimas las que mejores resultados ofrecen.

Para poder entender bien qué lugar ocupan las redes neuronales profundas dentro de un sistema de conversión texto a voz es necesario conocer cómo es la arquitectura general de uno. La Figura 2 muestra la arquitectura de bloques clásica de un sistema TTS. El texto es procesado en un bloque inicial denominado Front-End, donde se realizan dos tareas. En primer lugar, se normaliza el texto introducido convirtiendo los símbolos, números y abreviaturas en su equivalente en texto crudo. Por ejemplo, si el texto de entrada es “El 3/3/2019 hablamos con el Sr. Pérez” el texto normalizado

correspondiente sería “El tres de marzo de 2019 hablamos con el señor Pérez”. Posteriormente, el resultado de este pre-procesado es analizado para asignar transcripciones fonéticas de cada palabra y obtener información prosódica. En esta etapa se convierten los caracteres de entrada en una representación simbólica de los fonemas que es necesario pronunciar para leer correctamente ese texto y además se asigna a cada fonema posibles valores de duración, entonación y algunas características lingüísticas como si debe llevar acento, a qué tipo de frase (enunciativa, interrogativa, exclamativa…) pertenece, etc.

El *Back-End* se encarga de sintetizar la voz a partir de la información obtenida en el *Front-End*, produciendo las muestras de señal de voz que corresponden a la representación del texto de entrada producida por el *Front-End*.

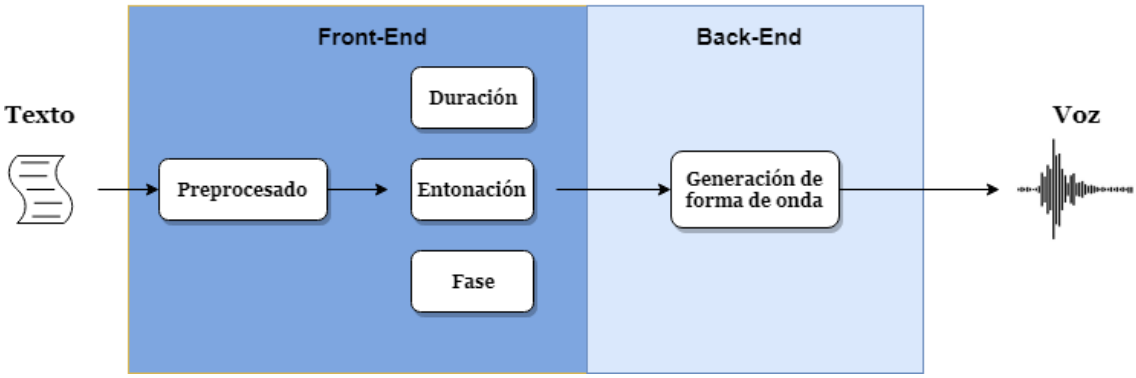



*Figura 2 Arquitectura clásica de bloques de un sistema TTS*

En los primeros trabajos que aplicaron redes neuronales en sistemas TTS, las redes empleadas sustituían únicamente la parte del *Back-End*. Sin embargo, últimamente han aparecido sistemas en los que las redes neuronales sustituyen la cadena completa. Estos sistemas se conocen como sistemas extremo a extremo o *End-to-End*. Deep Voice [11] fue el primero de estos sistemas en el que cada etapa del sistema TTS se implementaba mediante redes neuronales. La calidad de la señal generada no alcanza la que se obtiene al generar la señal con WaveNet y por ello se propusieron mejoras en

las versiones Deep Voice 2 [12] y Deep Voice 3 [13] en las que se puede generar la señal mediante WaveNet. Otro sistema con arquitectura totalmente extremo a extremo para TTS se trata de Tacotron [14] y su versión posterior Tacotron-2 [15] , la que se utiliza en el desarrollo de este proyecto. Tacotron funciona generando espectrogramas a partir del texto. Posteriormente, estos espectrogramas se convierten en voz utilizando el algoritmo Griffin-Lim [16] junto a la transformada de Fourier a corto plazo (STFT) o bien empleando WaveNet.

Existen más tecnologías extremo a extremo, pero por el momento, Tacotron es la que ha obtenido mejores resultados en las evaluaciones subjetivas de calidad y naturalidad. Este sistema ofrece una excelente puntuación empleando la métrica de opinión media (MOS), alcanzando un valor de 4.53, cerca de la puntuación de 4.58 obtenida por grabaciones reales realizadas en estudio [15] .[1] Tacotron tiene dos versiones, siendo la segunda la que ofrece mejores resultados. Este sistema recibe como entrada el texto a sintetizar y produce los espectrogramas de la señal de audio correspondiente mediante la aplicación de varias redes neuronales, tal y como se muestra en la Figura 3. Estos espectrogramas son posteriormente convertidos a forma de onda bien mediante otra red neuronal conocida como WaveNet [10] , bien con el algoritmo de Griffin-Lim [16] .

[1] Anexo I: Escala Mean Opinion Score (MOS)

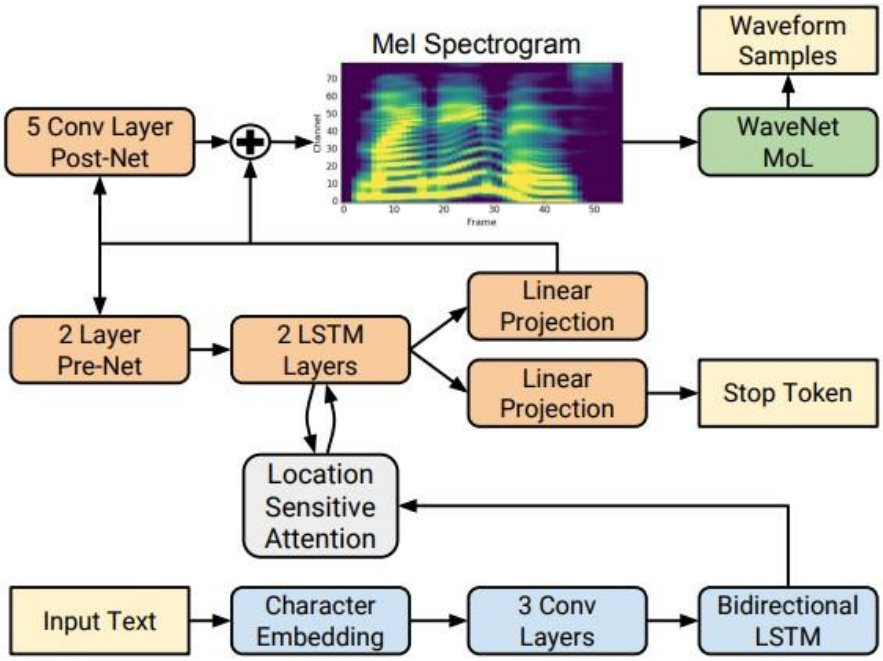



*Figura 3 Arquitectura de bloques del sistema Tacotron 2 [15]*

# 3 Objetivos

El objetivo principal del proyecto se trata de conseguir una voz sintética empleando el modelo Tacotron-2 en una lengua distinta a la inglesa. En primera instancia se realizará para una voz femenina en castellano, pero como objetivo secundario se plantea la creación de una voz sintética masculina en euskera.

Antes de llegar al objetivo final, existen una serie de objetivos parciales. En primer lugar, se replicará el experimento original empleando la base de datos disponible en inglés (LJ-Speech 1.1) sin realizar ninguna modificación sobre la misma. Esto permite por un lado familiarizarse con el modelo y los procedimientos necesarios, y por otro lado comprobar los tiempos necesarios de entrenamiento y síntesis con los recursos computacionales existentes hasta alcanzar un determinado umbral de calidad. El resultado obtenido tras este proceso será un modelo entrenado en las condiciones del laboratorio donde se ha realizado el proyecto, capaz de sintetizar voz en inglés con buena calidad.

El siguiente objetivo parcial se trata de la generación de un código y una metodología para preparar el corpus propio y adaptarlo al modelo como exige el mismo. Tras ello, se entrenará el modelo con distintos parámetros, se diseñará un corpus de texto válido para probar todos los aspectos que se consideren relevantes en la señal sintética y se comparará el resultado de las distintas pruebas, alcanzando así el objetivo final.

# 4 Beneficios

La ejecución de este proyecto reporta una serie de beneficios aplicables en distintos ámbitos. Si bien es cierto que el principal foco del mismo es conseguir mejoras en el ámbito social, la adaptación de Tacotron a distintas lenguas conlleva también beneficios técnicos y económicos.

## 4.1 Beneficios técnicos

En materia técnica destacan principalmente dos beneficios. Por una parte la replicación del experimento original. Desde la publicación del artículo [15] en 2018 se ha intentado ampliamente reproducir los resultados obtenidos, en la mayoría de los casos de forma infructuosa debido a los requisitos computacionales y el volumen de Corpus requerido. Un entorno y un corpus distinto sirven para comprobar la capacidad de adaptación del modelo, permitiendo observar la relación calidad/tiempo en los resultados obtenidos bajo potencias computacionales distintas.

Por otro lado, la adaptación del modelo a otras lenguas puede servir en futuros modelos destinados a la traducción automática neuronal, facilitando la implementación de un sistema E2E en el que se introduzca una voz en una lengua y se obtenga una traducción de la misma en otra lengua distinta.

## 4.2 Beneficios sociales

Los beneficios del proyecto tienen un mayor impacto en el ámbito social. Las personas con alguna discapacidad visual grave pueden hacer uso de un sistema TTS en una lengua distinta del inglés con una calidad alta.

También se benefician del proyecto personas con alguna discapacidad del habla o que deban someterse a una laringectomía total. Esta operación se realiza a personas que han sido diagnosticadas con cáncer de laringe y se trata de la extracción completa de la laringe. Tras esta operación el paciente pierde la voz y tiene que reaprender a hablar

utilizando distintas técnicas que le permiten comunicarse, pero con una voz esofágica o traqueoesofágica completamente distinta a la que tenían antes de someterse a la operación.

La adaptación de Tacotron a otra lengua permitiría al paciente la posibilidad de almacenar su voz antes de la operación para posteriormente reproducirla con unas características similares. De este modo podría reducirse el trastorno adaptativo producido tras la operación al conservar la identidad propia de la voz.

## 4.3 Beneficios económicos

Los resultados del proyecto podrían repercutir económicamente en el sector de los medios y el entretenimiento, debido principalmente a la digitalización de contenido por parte de grandes proveedores como HBO y Netflix. Redes sociales como Youtube, que generan aproximadamente 300 horas de vídeo cada minuto, facilitan el acceso a los consumidores a contenido audiovisual, quienes pueden visionar una cantidad de contenido extensamente superior a la generada exclusivamente por las cadenas de televisión tradicionales.

La exigencia de la audiencia también se incrementa. Los usuarios, en función de la cultura local, tal vez deseen visionar contenidos en su propia lengua, de modo que los mismos deberán ser doblados. La síntesis de voz, por tanto, tendría cabida en el mundo audiovisual, incorporándose como una herramienta más a la hora de generar contenido en diversos idiomas.

Por último, también es necesario contemplar el impacto que el proyecto podría ejercer en el sector de los videojuegos [17] , con un valor de mercado aproximado de 140.000 millones de dólares. El proyecto posibilitaría la creación de voces en diversos idiomas para los distintos personajes que quiera desarrollar la compañía de forma flexible y económica.

# 5 Descripción de requerimientos

## 5.1 Voz femenina y masculina

La adaptación al modelo debe realizarse tanto para una voz femenina como para una voz masculina, indistintamente de las diferencias tonales que puedan existir entre ambas voces. En principio el modelo es capaz de realizar el aprendizaje y captar las diferencias de forma autónoma, pero existen parámetros modificables que pueden afectar al resultado final y que hay que tener en cuenta.

## 5.2 Compatibilidad con castellano y euskera

Se deben generar voces tanto en castellano como en euskera, identificando y respetando las características de cada lengua en materia fonética y semántica. El alcance del proyecto únicamente recoge estas dos lenguas, pero los cambios realizados son aplicables a cualquier otra lengua que esté escrita con el alfabeto latino. Para lenguas con otro alfabeto basta con realizar una modificación sobre la tabla de símbolos empleada incluyendo los caracteres propios de su alfabeto.

## 5.3 Naturalidad e Inteligibilidad

La voz generada debe ser natural e inteligible, debe conservar las características de la voz original con la que se entrena el modelo y ser reconocible. Además debe de presentar una prosodia correcta, respetando las pausas, los acentos y la entonación de la persona que se graba.

# 6 Análisis de riesgos

Los potenciales riesgos que puedan afectar al proyecto están mayoritariamente relacionados con la complejidad del modelo utilizado.

Conceptualmente existe un riesgo medio dada la falta de experiencia propia en cuanto a tecnologías del habla, pero considerando el dominio previo en materia de aprendizaje automático este riesgo es subsanable realizando una fase previa extensa en la que se trabaje la bibliografía pertinente.

Para asegurar la fiabilidad de los resultados obtenidos y adquirir el manejo necesario de Tacotron, el proyecto presenta una dificultad incremental. En primer lugar se obtienen resultados utilizando el modelo sin alteraciones y con el corpus original en inglés con el que se realizó el artículo[15] . Posteriormente se realizan las modificaciones necesarias y se emplea el corpus propio en castellano, comparando los resultados con los obtenidos en la fase previa.

Uno de los principales riesgos que pueden afectar al proyecto es un volumen de corpus insuficiente para el entrenamiento del modelo. Dado que el modelo funciona principalmente para ser monolocutor, se disponen de 6´15 horas de grabación de gran calidad. El equivalente utilizado en lengua inglesa tiene una duración estimada de 24 horas de grabación.

A continuación se detallan de forma específica los principales riesgos presentes en el proyecto junto a su plan de contingencia.

| RIESGO | |
|---|---|
| **Descripción** | Volumen de corpus insuficiente para entrenar el modelo |
| **Impacto** | Alto |
| **Probabilidad** | Media |
| **Plan de contingencia** | El corpus disponible presenta una duración aparentemente buena, pero comparativamente muy inferior a utilizada para lengua inglesa.<br>Para paliar este riesgo existen técnicas como el "*fine-tuning*", que permiten secuenciar distintos corpus durante el entrenamiento del modelo, pudiendo incrementar el número de horas de grabación disponibles para el entrenamiento. |

*Tabla 1 Volumen de corpus insuficiente*

| RIESGO | |
|---|---|
| **Descripción** | Resultados inferiores a los obtenibles en la lengua inglesa |
| **Impacto** | Medio |
| **Probabilidad** | Alta |
| **Plan de contingencia** | Dadas las limitaciones en materia de volumen del corpus y recursos computaciones se asigna una probabilidad alta a la obtención de resultados inferiores a los obtenidos en inglés.<br>Una correcta ejecución de los planes de contingencia asociados a los problemas previamente mencionados minimizará la distancia entre los resultados obtenidos en inglés y los obtenidos en otra lengua. |

*Tabla 2 Resultados inferiores al habla inglesa*

| RIESGO | |
|---|---|
| **Descripción** | Insuficiencia de recursos computacionales necesarios para entrenar el modelo. |
| **Impacto** | Alto |
| **Probabilidad** | Media |
| **Plan de contingencia** | La carencia de recursos computacionales es un riesgo muy grande en el ámbito del aprendizaje automático, puesto que las horas de computación y la potencia de cálculo son los principales fundamentos del mismo, por lo que se asigna un impacto alto a este riesgo.<br>En el entorno de trabajo en el que se desarrolla este proyecto los recursos computacionales son compartidos con distintos proyectos, de modo que la probabilidad del suceso es media. El establecimiento de colas de prioridad para el acceso a los mismos minimiza la probabilidad del suceso. |

*Tabla 3 Insuficiencia de recursos computacionales*

# 7 Análisis de alternativas

A la hora de escoger una implementación de Tacotron, son varias las alternativas que se presentan y entre las cuales hay que elegir para obtener un resultado final óptimo y conforme al alcance el proyecto. En este apartado se presentan las alternativas más representativas entre las que elegir, justificando su elección en base a unos criterios detallados al comienzo del apartado.

## 7.1 Implementación de Tacotron

Tacotron se trata de un proyecto de código abierto, de modo que son varias las personas y entidades que han puesto en marcha implementaciones asentadas en los conceptos expuestos en el artículo publicado en 2018 [15] .

Para la realización de este proyecto concreto, se escoge una implementación basándose en los siguientes criterios:

- Experiencia previa: si bien es cierto que todas las implementaciones están codificadas en Python, las librerías principales que utilizan cada una de ellas son distintas. Este apartado valora la experiencia previa propia con el manejo de las distintas librerías, facilitando la tarea de realizar las modificaciones necesarias.
- Calidad de resultados: este criterio es fundamental y se basa en la calidad de los resultados obtenidos con cada implementación. La valoración es subjetiva puesto que las implementaciones entre las que elegir no disponen de la misma métrica con la que compararse, pero sí es posible encontrar ejemplos de audio sintético producidos con todas ellas con los que poder hacer una comparativa.
- Solución de problemas: los repositorios de cada implementación están abiertos a la resolución de problemas y propuestas de mejora por parte de la comunidad. Este apartado puede resultar trivial, pero dada la naturaleza de los

proyectos de "*Deep learning*" es una característica muy relevante a la hora de realizar un proyecto.

El peso otorgado a cada uno de los criterios es el mismo en este caso, de modo que cada uno de ellos pondera lo mismo de cara a la elección.

### 7.1.1 Alternativas

#### *7.1.1.1 Implementación Rayhane Mamah*

Rayhane Mamah es un ingeniero industrial especializado en técnicas de Deep Learning. Su implementación en GitHub del Tacotron-2 es la mejor valorada por la comunidad y aunque actualmente no reciba actualizaciones podría decirse que ha llegado a un estado funcional y estable.

Esta implementación hace uso de la librería Tensorflow, librería que permite generar y entrenar modelos de aprendizaje automático (*Machine Learning*, ML) de manera sencilla utilizando APIs de alto nivel como *Keras*.

Con 189 consultas resueltas, esta implementación se presenta como la mejor en el ámbito de resolución de problemas, aportando además explicaciones detalladas y un gran número de referencias para facilitar la compresión de su funcionamiento.

#### *7.1.1.2 Implementación NVIDIA*

La compañía californiana NVIDIA también dispone de un repositorio en GitHub del sistema Tacotron 2, junto a diversos proyectos de "*Deep Learning*". Esta implementación es la segunda mejor valorada en *GitHub* y presenta una gran flexibilidad a la hora de sintetizar la voz deseada a partir de los espectrogramas de mel.

El proyecto ha sido realizado utilizando la librería *Pytorch*, una librería reciente que trabaja a más bajo nivel que *Tensorflow* pero con la misma funcionalidad. Además, el proyecto se ejecuta sobre *Jupyter Notebooks*, una plataforma web de programación

con funcionamiento basado en celdas, al igual que otros programas como por ejemplo *Wolfram Mathematica*.

Esta implementación tiene un total de 161 consultas resueltas y unos resultados muy buenos, pero ligeramente inferior a los obtenidos con la implementación de Rayhane.

#### *7.1.1.3 Implementación Ryuichi Yamamoto*

Ryuichi Yamamoto es el desarrollador de la versión de WaveNet mejor valorada existente en *GitHub*. Las implementaciones anteriormente mencionadas hacen uso del código desarrollado por Ryuichi para hacer las funciones de *Back-End*.

Junto a la colaboración de Rayhane, Ryuichi ha desarrollado una versión de Tacotron en la plataforma *Google Collab*, permitiendo utilizar el código de forma sencilla y empleando los recursos computacionales de Google.

Los resultados obtenidos son los mejores de las tres implementaciones escogidas y además facilitan un modelo de WaveNet pre-entrenado con un millón de iteraciones en lengua inglesa. Sin embargo, las características del código hacen que sea el más complejo de modificar para adaptarlo a distintos idiomas de entrada.

### 7.1.2 Elección

A continuación se mostrará una tabla que recoge la decisión final:

| | Rayhane | NVIDIA | Ryuichi |
|---|---|---|---|
| **Experiencia Previa** | 3 | 2 | 1 |
| **Calidad Resultados** | 2 | 1 | 3 |
| **Resolución problemas** | 3 | 2 | 1 |
| **TOTAL** | **8** | **5** | **5** |

*Tabla 4 Selección de alternativas*

Así pues, la implementación empleada es la de Rayhane Mamah.

# 8 Descripción de la solución

Para obtener una versión de Tacotron 2 adaptada a otros idiomas como el castellano, es necesario realizar diversas tareas y en este apartado se describen los aspectos más importantes que es necesario tener en cuenta.

## 8.1 Formato de entrada

El primer paso a la hora de utilizar Tacotron es preparar el corpus para introducirlo en el modelo. Tacotron se entrena con un conjunto de señales de audio y sus respectivas transcripciones ortográficas. Para la realización del proyecto se dispone dos corpus grabados con alta calidad.

Ambos corpus han sido grabados con una tasa de muestreo de 48 kHz y posteriormente diezmada a 16 kHz. El corpus en castellano (Aintzane_es) está compuesto por 3876 archivos de audio con extensión .wav y duración agregada de 6.15 horas. El corpus en euskera (Kiko_eu) está compuesto por 3795 archivos de audio con extensión .wav y duración agregada de 5.34 horas.

Durante la etapa de preparación de los datos se realizan modificaciones tanto a los textos como a los ficheros de audio que componen el corpus, generándose también nuevos recursos necesarios en algunas de las etapas. A continuación, se recogen todas las modificaciones realizadas.

### 8.1.1 Recodificación de los textos y cálculo de parámetros

Para poder facilitar la tarea deben juntarse todos los ficheros de audio en una carpeta y los textos en un único fichero que contenga todas las frases, cada una en una línea. En el caso particular de los corpus con los que se ha trabajado, las frases estaban almacenadas en distintos ficheros de texto, todos ellos con codificación ISO -8859-1. Tacotron trabaja con una codificación UTF-8, de modo que es necesario recodificar las frases con un sencillo fragmento de código como el siguiente:

```
import glob
from tqdm import tqdm

read_files = glob.glob("/home/victor/textos/*.txt")

with open("kikotexts.csv","w") as outfile:
    for f in tqdm(read_files):
        with open(f,"rb") as infile:
            text = infile.read().decode('ISO-8859-1').encode('utf-8')
            tmp = str(text,'utf-8')
            outfile.write(f[-11:-4] + "|" + tmp)
```

Además se deben calcular también una serie de parámetros que deben configurarse en un fichero denominado “hparams.py” tanto para uso propio como para uso del modelo. Estos parámetros son:

- Tasa de muestreo, en este caso 16 kHz
- Tamaño de ventana, se recomienda una ventana de 50 ms
- Desplazamiento, se recomienda un desplazamiento de 12.5 ms

El Anexo III: Fichero hparams.py recoge todos los parámetros calculados y distintos parámetros que afectan al entrenamiento y síntesis de Tacotron.

### 8.1.2 Recorte de silencios

Las señales de audio de entrada pueden presentar silencios de inicio y final que deben ser eliminados. Estos silencios afectan negativamente al entrenamiento, creando una indeterminación para el modelo al respecto del verdadero comienzo de la frase y de su final. La Figura 4 muestra un audio sin preprocesar.

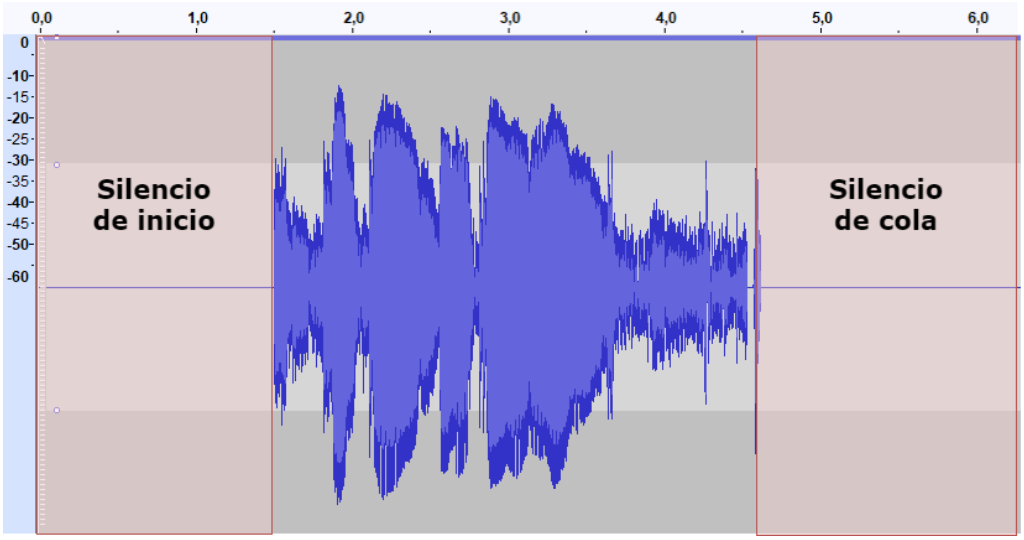



*Figura 4 Audio con silencio de inicio y silencio de cola*

Para realizar el recorte, se establece como nivel de referencia el valor más alto de la señal en el tiempo y un umbral por debajo de ese valor de referencia. La elección del umbral depende del Corpus utilizado. Si la señal no supera dicho umbral al inicio o al término, será considerado como silencio y se recortará, obteniendo un resultado como el mostrado en la Figura 5:

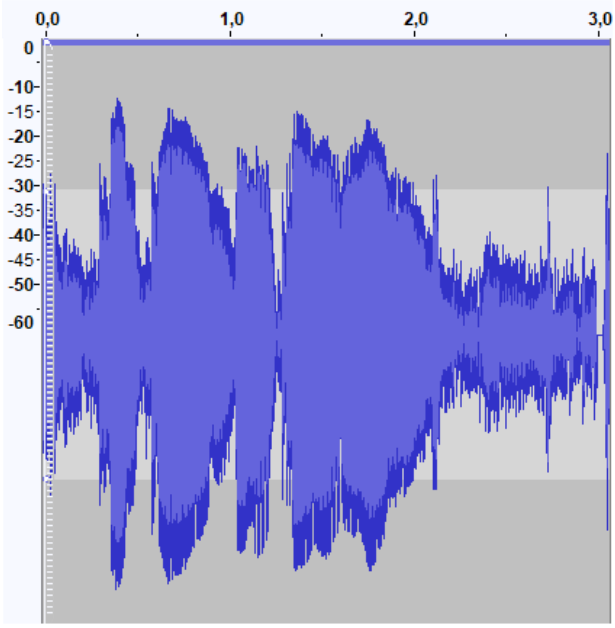



*Figura 5 Señal recortada*

### 8.1.3 Preénfasis

Se denomina preénfasis a la aplicación de un filtro, generalmente lineal, que potencia las frecuencias más altas de una señal. En el caso de Tacotron, resulta útil aplicar un preénfasis ya que la voz presenta en general un espectro con pendiente negativa. Con el filtro de preénfasis se equilibra la energía de la banda alta y baja, amplificando las altas frecuencias. Este paso debe deshacerse tras obtenerse la voz sintetizada aplicando el mismo filtro invertido. La Figura 6 muestra un ejemplo de una señal sin aplicar preénfasis (en azul) y aplicándolo (rojo). Se puede observar cómo las altas frecuencias, a partir de 3kHz son potenciadas por el filtro de preénfasis, mientras que las bajas frecuencias (por debajo de 3kHz) se ven atenuadas.

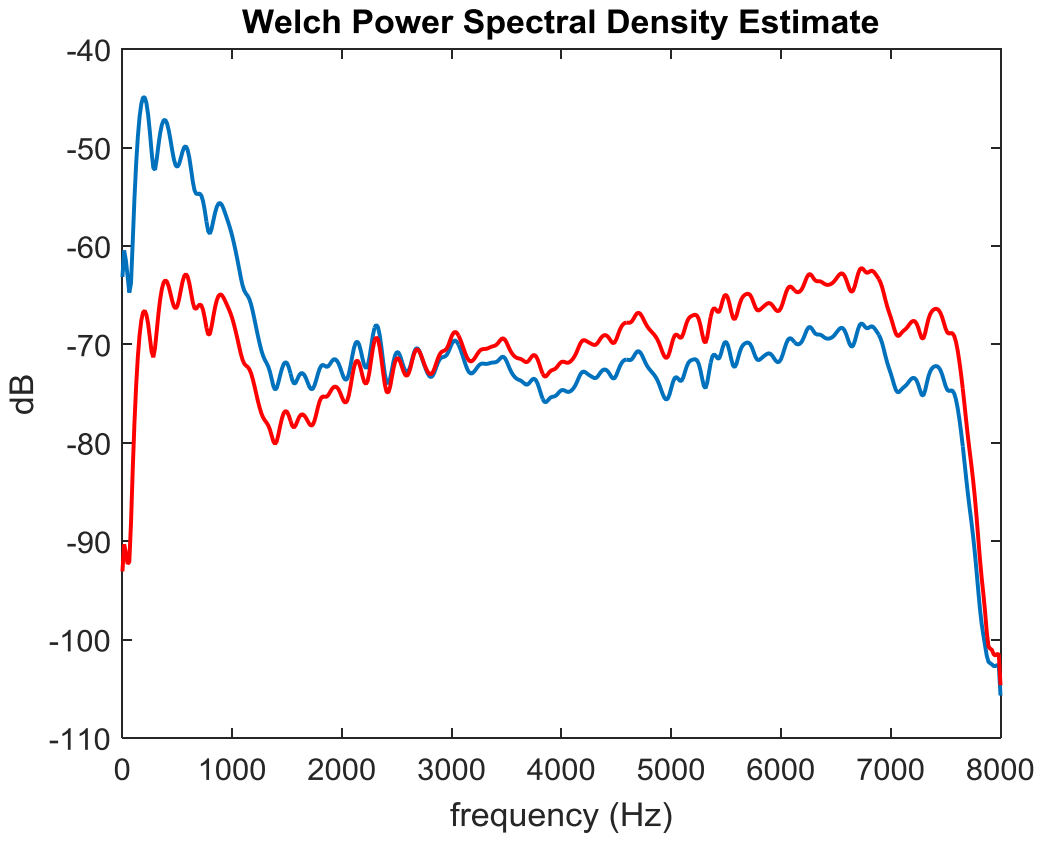



*Figura 6 Señal sin preénfasis (azul) y con preénfasis (rojo)*

### 8.1.4 Re-escalado

Tras la aplicación del preénfasis es necesario realizar un re-escalado de las señales de entrada para que todas las muestras se encuentren en un rango comprendido entre -1 y 1. El motivo por el cual se realiza este re-escalado es para conservar la información relativa al rango dinámico de la señal pero permitir al modelo que interprete palabras y caracteres iguales de forma similar, es decir, bajo una misma escala.

### 8.1.5 Creación de espectrogramas

Finalmente se generan los espectrogramas de todas las señales de audio. Este proceso se realiza empleando la librería *Librosa* y su función para realizar transformadas de Fourier a corto plazo. La Figura 7 muestra la estructura de directorios resultante tras generar todos los recursos necesarios. La carpeta “Audio” contiene todos los ficheros de audio con formato .npy (estructura vectorial en Python). La carpeta “Linear” contiene los espectrogramas en escala lineal de los audios y la carpeta “Mels” contiene los espectrogramas en escala de mel. La escala mel es una escala frecuencial no lineal que tiene en cuenta la respuesta del oído humano, de manera que se tiene mayor resolución en el espectrograma en las zonas en las que el oído es más sensible (bajas frecuencias) a costa de perder resolución frecuencial en las zonas donde el oído tiene menos sensibilidad (altas frecuencias). El fichero “train.txt” contiene cada frase del corpus junto a los *paths* en los que encontrar su correspondiente audio o espectrograma.

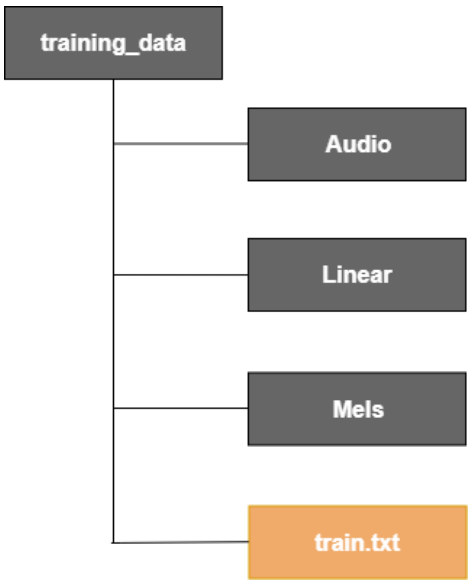



*Figura 7 Estructura de carpetas*

## 8.2 Tacotron

Tacotron 2 se trata de un sistema basado en redes neuronales capaz de sintetizar voz a partir de texto. Su funcionamiento está basado en la combinación de redes neuronales convolucionales (*Convolutional Neural Network*, CNN) y redes neuronales recurrentes (*Recurrent Neural Network*, RNN). El sistema consta de dos partes diferenciadas, como puede apreciarse en la Figura 8:

- Por una parte una red secuencia a secuencia (*sequence-to-sequence*, seq2seq) recurrente con mecanismo de atención, encargada de predecir una secuencia de tramas pertenecientes a un espectrograma de mel partiendo de una secuencia de entrada formada por caracteres.
- Por otro lado una versión modificada de WaveNet, que se encarga de generar muestras de forma de onda en el dominio temporal, condicionadas en base a los espectrogramas de mel predichos en la etapa anterior.

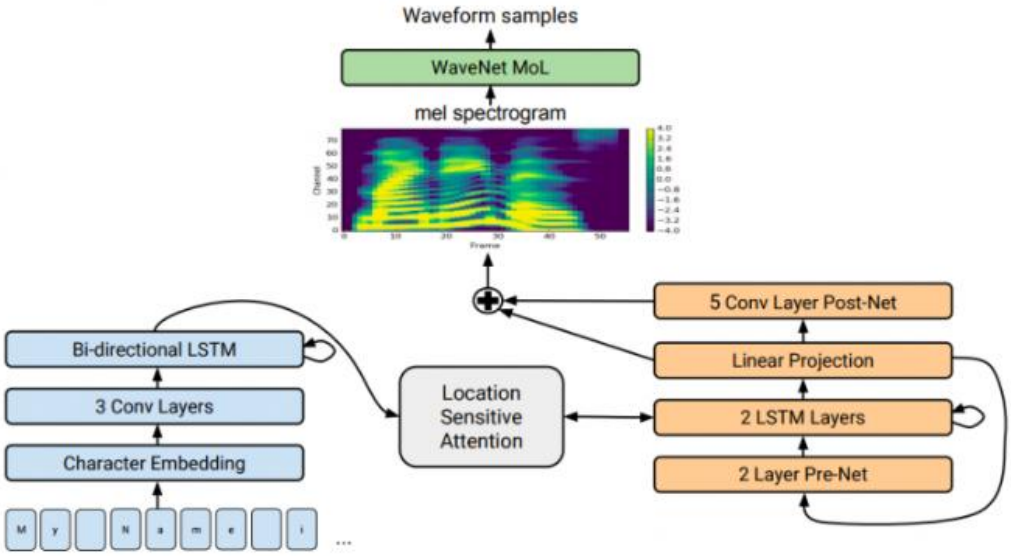



*Figura 8 Arquitectura Tacotron-2. En azul la parte destinada al Encoder y en naranja al Decoder [15]*

Generalmente, cuando se hace uso de redes neuronales, los escenarios que requieren convertir una secuencia de entrada en una de salida, suelen implementar arquitecturas codificador-decodificador (*Encoder-Decoder*) como la que aparece en la Figura 9.

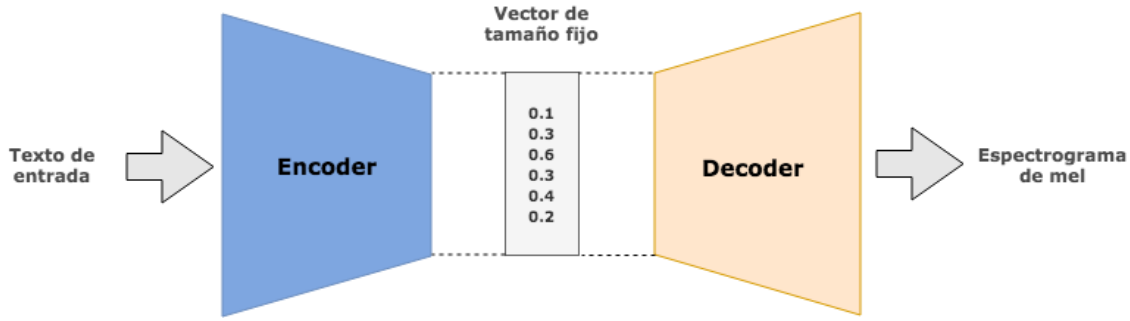



*Figura 9 Arquitectura Encoder-Decoder*

Esta imagen representa una arquitectura *Encoder-Decoder* simple, en la que el *Encoder* comprime la secuencia de entrada en un vector de tamaño fijo y el *Decoder* intenta predecir la secuencia de salida empleando dicho vector. Se suelen utilizar RNNs para ejercer la función de *Encoder* y *Decoder*.

Sin embargo, esta arquitectura presenta complicaciones en el caso de que la secuencia de entrada sea larga, puesto que el *Encoder* tiene que realizar un resumen del contenido de la misma en el vector de tamaño fijo, pudiendo llegar a perderse información en función del tamaño de la secuencia. Este problema es especialmente notable en Tacotron, puesto que los textos introducidos son de tamaño variable y pueden llegar a exceder la capacidad del *Encoder*. Por ello, Tacotron implementa una técnica denominada “mecanismo de atención” [18] .

El propósito del mecanismo de atención es liberar al *Encoder* de la tarea de resumir la secuencia de entrada. Para ello, el *Encoder* pasa a realizar una serie de anotaciones de la misma longitud que la secuencia de entrada, y el *Decoder* aprende a “atender” a las partes más relevantes de dichas anotaciones para generar la secuencia de salida. La Figura 10 permite visualizar de forma sencilla un mecanismo de atención implementado en un escenario de traducción secuencia-a-secuencia.

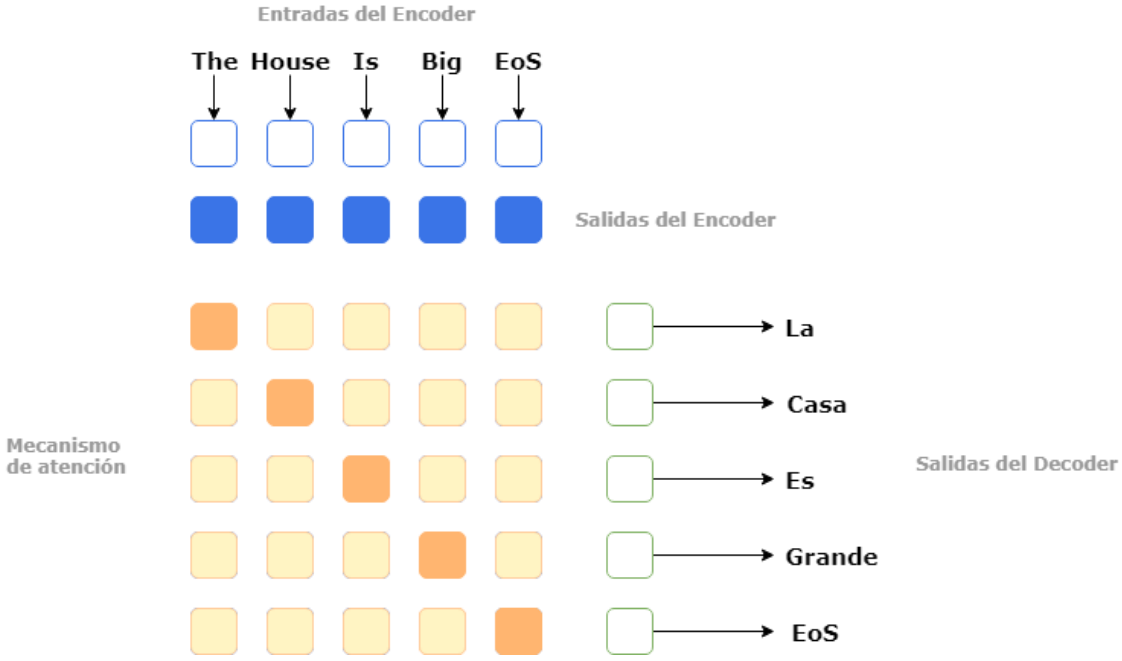



*Figura 10 Mecanismo de atención[19]*

Las explicaciones previas muestran de forma genérica las funciones de un *Encoder* y un *Decoder* con mecanismo de atención, pero existen distintas formas de implementar ambas partes. A continuación se detallan ambos elementos en Tacotron.

### 8.2.1 *Encoder*

En primer lugar la secuencia de caracteres debe ser introducida en el sistema con una representación con la que pueda trabajar. La técnica mediante la cual se obtienen estos caracteres se denomina "*Character embedding*". En este caso los caracteres serán representados según un vector de 512 dimensiones. Para ello, es necesario definir qué caracteres conforman el conjunto total de posibles representaciones. En el caso del Tacotron sin modificar existen 67 caracteres representables, incluyendo mayúsculas, minúsculas y diversos signos como guiones, signos de exclamación, etc…

Para poder adaptar Tacotron al castellano se incorporan 8 caracteres más, las vocales acentuadas, la letra “ñ” y los signos de apertura interrogativo y exclamativo. En el corpus que se ha utilizado no existen vocales acentuadas en mayúsculas, de lo contrario habría que añadirlas.

La Figura 11 muestra una representación de dimensiones reducidas del *character embedding* realizado durante el entrenamiento del modelo. La representación se realiza automáticamente en la herramienta de visualización Tensorboard [20] la cual aplica una técnica denominada análisis de componentes principales (*Principal Component Analysis,* PCA), que permite reducir las dimensiones del *embedding* para visualizarlo en tres dimensiones. Cada carácter queda representado por una esfera en el espacio. La posición relativa de cada punto con respecto al resto tiene un significado interno dentro del modelo, y es desconocido, aunque se pueden deducir ciertas interpretaciones como por ejemplo la cercanía espacial entre las vocales, o diversos grupos que se encuentran cercanos entre sí y presentan una articulación parecida entre ellos.

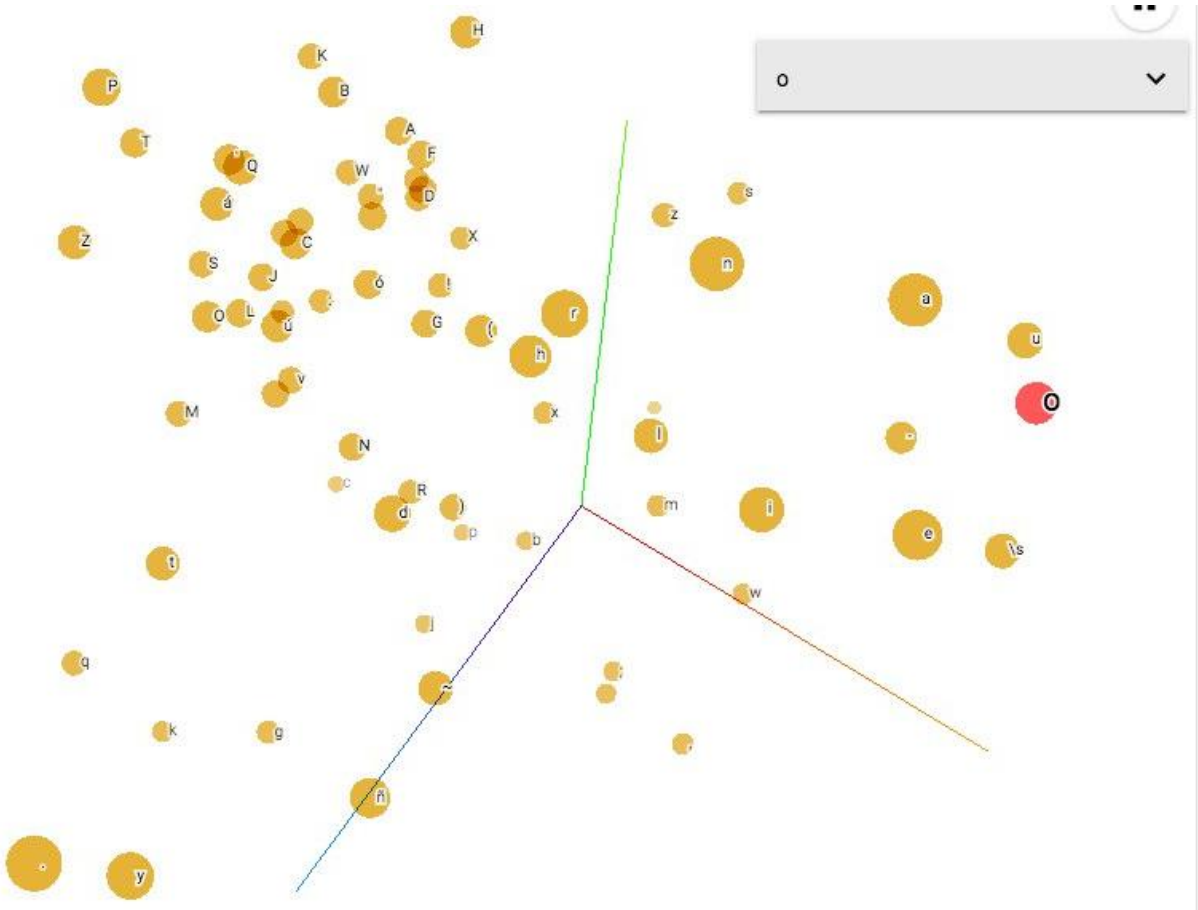



*Figura 11 Representación tridimensional del Character embedding*

La salida del *embedding* pasa posteriormente por 3 redes convolucionales (CNN) que se encargan de identificar y codificar relaciones entre los caracteres de entrada, aunque se encuentren alejados entre sí. Finalmente, la salida de la última CNN se introduce en una red LSTM bidireccional encargada de generar los denominados estados ocultos o anotaciones previamente nombradas, que serán utilizadas por el mecanismo de atención para generar los vectores (vectores contexto) utilizados por el *Decoder* para predecir las tramas de los espectrogramas de mel.

### 8.2.2 Decoder

La Figura 12 muestra cómo implementa Tacotron 2 el *Decoder* junto al mecanismo de atención. Una de las bases sobre las que se asienta la robustez del modelo se trata de la utilización de parámetros obtenidos en etapas anteriores, permitiendo al modelo disponer de una visión global del proceso y permitiéndole avanzar aunque existan secuencias repetidas.

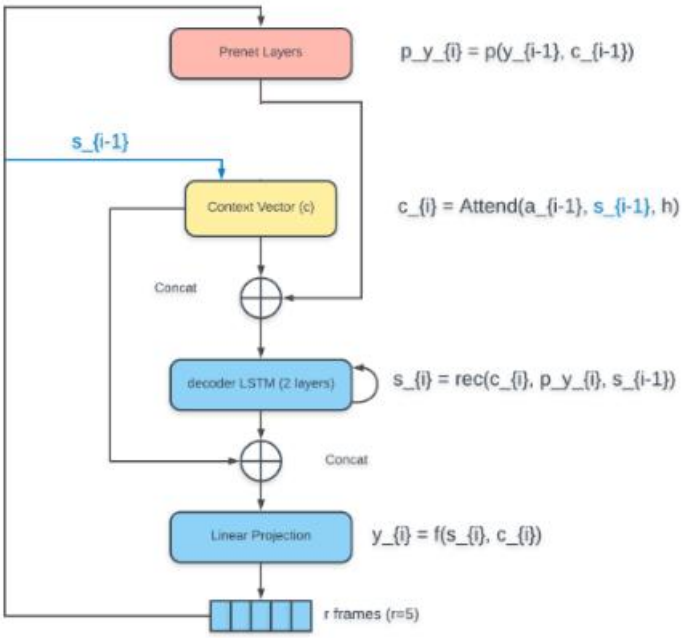



*Figura 12 Decoder implementado por Rayhane Mamah[19]*

La capacidad de implementar información relativa a etapas anteriores la proporcionan principalmente dos capas totalmente conectadas entre sí denominadas Pre-net, en las que se introduce o bien la trama real de la etapa anterior (durante el entrenamiento de la red) o bien la trama predicha en la etapa anterior (durante la síntesis). Esta capa es imprescindible para que el *Decoder* "aprenda a atender" correctamente y realice una predicción correcta de la salida. Esta salida se calcula en una etapa compuesta por

dos redes LSTM y se desarrolla en base a un vector contexto y a las tramas del espectrograma de mel predichas con anterioridad.
Finalmente las tramas predichas atraviesan 5 capas convolucionales denominadas *post-net,* cuya función es calcular un valor residual que sumado a las propias tramas incrementa la calidad del espectrograma final.

## 8.3 Entrenamiento Tacotron

Una vez conocida la arquitectura y el funcionamiento general de Tacotron, se realiza el entrenamiento del modelo con el corpus propio. Para simplificar la explicación de este apartado, únicamente se empleará el corpus en castellano, dado que el procedimiento a realizar con el corpus en euskera es relativamente similar.

De forma muy simplificada, el modelo de Tacotron basa su aprendizaje en obtener un error cuadrático medio (MSE) cada vez más pequeño en los espectrogramas de salida. Para ello, realiza una predicción de un espectrograma a partir de un texto, lo compara con el espectrograma real de la misma frase del corpus y modifica los estados internos de las celdas existentes en cada capa del modelo para obtener un resultado más parecido al objetivo real.

Si se emplease todo el corpus para entrenar el modelo, podría incurrirse en un problema a la hora de generalizar, es decir, el modelo podría estar tan condicionado a las muestras del entrenamiento que al intentar predecir una secuencia distinta a la conocida el resultado sea muy pobre. Este suceso es conocido como sobre-entrenamiento (*Overfitting),* y está relacionado con distintas causas como por ejemplo un modelo demasiado complejo o una base de datos de entrada pequeña.

Para evaluar el proceso, es necesario emplear una parte del corpus con la que el modelo no esté sesgado, es decir, que sea transparente para el modelo, que no la haya "visto" con anterioridad. Es necesario, por lo tanto, reservar una parte de los datos para evaluación, de manera que no sean utilizados durante el proceso de

entrenamiento. Ese fragmento del corpus se conoce como set de evaluación y permite identificar si el proceso de aprendizaje está sufriendo de *Overfitting.*

Para monitorizar el proceso de entrenamiento es necesario observar una serie de gráficas que permiten obtener una idea acerca del estado en el que se encuentra el modelo, si es necesario seguir entrenando, o si debe pararse el entrenamiento y modificar alguna de las variables. Las gráficas que deben controlarse durante el proceso de entrenamiento son las siguientes:

- Gráfica de alineamiento, la cual muestra cómo está aprendiendo el modelo el mecanismo de atención. La gráfica ideal debería mostrar una línea diagonal.
- Gráfica de pérdida durante el entrenamiento, muestra el error cuadrático medio cometido en cada iteración del bucle de aprendizaje, suele presentar una forma exponencial decreciente.
- Gráfica *"stop_token_loss".* Esta es una característica distintiva entre Tacotron 2 y su predecesor. Los espectrogramas de Mel predichos por la primera versión de Tacotron presentaban una duración fija, mientras que Tacotron 2 dispone de este mecanismo, que se presenta como una opción dinámica de parada en síntesis para identificar cuándo parar de decodificar, en lugar de establecer un límite fijo.

El 15.2 más adelanteAnexo II: Gráficas de entrenamiento Tacotron 2. recoge estas gráficas durante el entrenamiento realizado con el corpus propio. El tiempo transcurrido hasta un entrenamiento completo del modelo con el corpus en castellano es de 4 días y de 5 días con el corpus en euskera.

## 8.4 Síntesis Tacotron

Una vez el entrenamiento finaliza, se puede realizar la síntesis de las distintas frases escogidas para comprobar el funcionamiento del modelo. De acuerdo con los

Objetivos establecidos, las frases escogidas deben permitir evaluar los siguientes aspectos:

- Distinción de mayúsculas y minúsculas
- Capacidad de reproducir la acentuación correctamente
- Identificación de signos interrogativos y exclamativos
- Capacidad de reproducir la letra “ñ”
- Respuesta ante faltas ortográficas
- Reproducción de pausas (comas, punto y coma…)

Así pues, se ha diseñado un corpus textual para pruebas teniendo en cuenta los requerimientos anteriores. Las frases escogidas que contienen todas las características previamente mencionadas son las siguientes:

- Ésta es una frase de prueba
- ¿Funcionan los signos de interrogación?
- Mañana se empañará el cristal, tendremos que desempañarlo con empeño.
- No podré ir a pescar porque se me ha olvidado la caña, mañana la traigo.
- Desayuné zumo, leche y zereales.
- Anoche perdí la yave del vuzón.
- ¿Qué sucede si falta una palabra en frase?
- Todos los hombres mueren, pero no todos han vivido.
- Hasta el infinito y más allá.
- Hazlo o no lo hagas, pero no lo intentes.
- ¿De qué sirve confesarme si no me arrepiento?
- Todos esos momentos se perderán en el tiempo como lágrimas en la lluvia.
- No existen preguntas sin respuesta, solo preguntas mal formuladas.
- La mejor jugada del diablo fue convencer al mundo de que no existía.
- En ocasiones veo muertos.

- En ocasiones, veo muertos.
- Tres tristes tigres tragan trigo en un trigal.
- Mantén cerca a tus amigos, pero aún más cerca a tus enemigos.
- Siempre digo: ve el lado bueno de las cosas.
- Siempre digo; ve el lado bueno de las cosas.
- ¡Muchas gracias por todo!
- ¡MUCHAS GRACIAS POR TODO!

El último paso para completar el proyecto se trata de generar la voz sintética a partir de los espectrogramas predichos. Para ello, se puede hacer uso de dos herramientas. Por un lado, el Algoritmo de Griffin-Lim [16] aplicable directamente sobre los resultados obtenidos. Por otro lado existe la posibilidad de emplear WaveNet [10] una red neuronal condicionada con espectrogramas de mel para generar la voz sintética. A continuación se incluye una breve explicación de ambas herramientas.

## 8.5 Algoritmo de Griffin-Lim

Para reconstruir una señal en el dominio temporal es necesario conocer la información relativa a su magnitud y su fase. Los espectrogramas obtenidos únicamente presentan información relativa a la magnitud de la señal, pero no se dispone de la información relativa a la fase. Para obtenerla, el algoritmo de Griffin-Lim propone los siguientes pasos [21] :

- Crear una matriz compleja, introduciendo en la parte real los datos referentes a la magnitud real obtenida en los espectrogramas predichos y rellenando la parte imaginaria con ruido uniforme.
- Aplicar una transformada inversa de Fourier de corto plazo (ISTFT) sobre dicha matriz, obteniendo una señal en el dominio temporal
- Realizar una transformada de Fourier de corto plazo (STFT) sobre la señal obtenida. De esta forma, se obtiene un poco de información de la fase. El ruido

que inicialmente se había introducido en la parte imaginaria de la matriz compleja generada pasa a contener algo de información.

- Sobre la nueva matriz obtenida, se sustituye la parte real por la información original.
- Se vuelve al paso 2 y se itera sucesivamente hasta lograr un resultado satisfactorio.

Cada iteración presenta una información de fase cada vez mejor. En la realización de este proyecto se utilizan 60 iteraciones, asegurando la convergencia del algoritmo. En caso de que el proceso sea demasiado lento podría disminuirse esta cifra a 30 iteraciones.

## 8.6 WaveNet

*WaveNet* es un modelo generativo de audio basado en la arquitectura de otro modelo denominado *PixelCNN* [22], capaz de modelar la distribución de probabilidad conjunta entre miles de variables.

La probabilidad conjunta de una forma de onda $(\mathrm{p}(\mathrm{x}))$ se puede factorizar como el producto de distintas probabilidades condicionales como muestra la siguiente ecuación:

$$\mathrm{p}(\mathrm{x}) = \prod_{t=1}^{T} (x_t |\, x_1, \dots, x_{T-1})$$

donde $x_t$ representa una muestra de audio en un instante t y se ve condicionada por todas las muestras anteriores.

Para modelar la distribución de probabilidad condicional se utiliza una pila de capas convolucionales. Las convoluciones realizadas son causales, de modo que la salida del sistema en un instante "t" jamás dependerá de instantes futuros "$t+t_0$". Para incrementar la cantidad de información que perciben las neuronas que componen las

capas convolucionales del sistema, se utiliza una convolución dilatada, como la que aparece en la Figura 13.

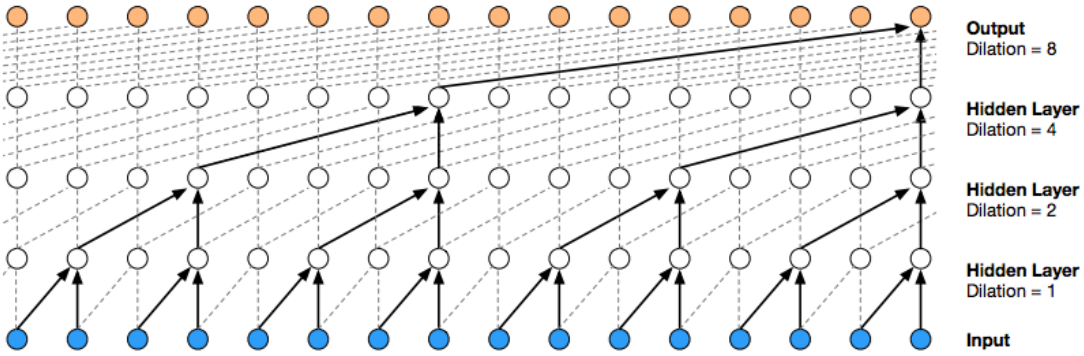



*Figura 13 Pila de convoluciones causales dilatadas*

Si se incorpora una entrada adicional (h), WaveNet puede modelar la probabilidad conjunta $\mathrm{p(x \mid h)}$ como:

$$\mathrm{p(x \mid h)} = \prod_{t=1}^{T} (x_t | x_1, \dots, x_{T-1}, \mathrm{h})$$

Condicionando el modelo con esa nueva entrada se pueden generar voces naturales, con las características del habla de la persona que genera los datos de entrada. Existen dos tipos de condicionamiento de entrada; un condicionamiento de entrada global y un condicionamiento de entrada local. El condicionamiento global influye sobre la salida de manera uniforme en el tiempo, es decir, afecta a todas las muestras de la señal generada como por ejemplo un condicionamiento de locutor (*speaker embedding*). El condicionamiento de entrada local es información variable en el tiempo, como puede ser un espectrograma de mel, cuyos valores cambian de trama a trama.

Para que las distintas capas sean capaces de aprender la probabilidad condicionada, es necesario entrenar el modelo. En el caso del presente proyecto, el modelo puede ser entrenado empleando los espectrogramas de mel reales de los distintos audios

grabados (*Ground-Truth*) o empleando los espectrogramas de mel predichos por Tacotron (*Ground-Truth-Aligned*).

El artículo [15] establece que los resultados obtenidos entrenando con los espectrogramas de mel predichos son superiores a los obtenidos con los espectrogramas reales. El motivo es que en tiempo de síntesis WaveNet debe generar la voz sintética empleando como entrada los espectrogramas predichos. Si la red se entrena con una serie de espectrogramas predichos, aprende los errores que haya en ellos y es capaz de corregirlos a la hora de realizar la síntesis.

# 9 Descripción de los resultados

El siguiente enlace recogen muestras de las señales de audio más importantes generadas durante el proceso https://aholab.ehu.eus/users/victor/. Como puede observarse, se ha logrado el objetivo de generar voces sintéticas femeninas y masculinas, tanto en castellano como en euskera.

De las dos técnicas empleadas para obtener la señal de audio a partir de los espectrogramas predichos, únicamente Griffin-Lim ha proporcionado resultados satisfactorios. La falta de precisión en los resultados obtenidos utilizando WaveNet se debe a falta de tiempo de entrenamiento.

Las voces generadas conservan la prosodia y las características de timbre original de las personas que han grabado los corpus con los que se entrenan los modelos. Las frases son inteligibles y aunque muestren unas pequeñas vibraciones, puede afirmarse que son bastante naturales.

El modelo en castellano ha aprendido correctamente las características propias del idioma, es capaz de reconocer la letra “ñ” y las acentuaciones, y además responde positivamente a las pausas introducidas en el texto, como por ejemplo las comas. En las frases generadas no se aprecia diferencia entre mayúsculas y minúsculas puesto que el corpus de entrada no contiene texto en mayúsculas.

La Figura 14 muestra el espectrograma predicho de una de las frases que componen el conjunto de frases de prueba. En primer lugar se aprecia un silencio bastante amplio al comienzo del espectrograma, silencio que se repite a lo largo de todas las frases de prueba. El motivo de este silencio inicial se debe a un mal recorte de los silencios de inicio en los audios con los que se entrena el modelo, causado por una elección incorrecta del umbral con el que se detectan el silencio. Aparentemente, este silencio no modifica la calidad del audio generado, pero sí que genera cierta inestabilidad a la

hora de sintetizar, resultando en una mala síntesis en algunas de las frases. La solución del problema no se ha llevado a cabo por falta de tiempo, pero pasaría por escoger un valor de referencia con el que sí se recorten las señales, repitiendo el entrenamiento de Tacotron con las nuevas señales obtenidas.

Otra característica que puede observarse es un suavizado de la señal, especialmente en las componentes frecuenciales altas. Este problema es propio de Tacotron-2 y únicamente se puede mejorar con más corpus y más horas de entrenamiento, pero no se puede eliminar por completo.

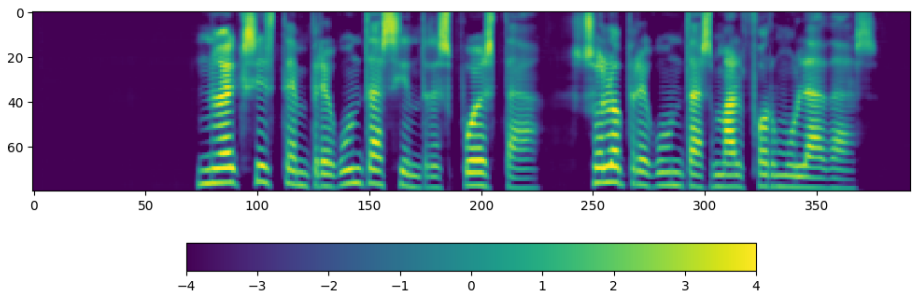
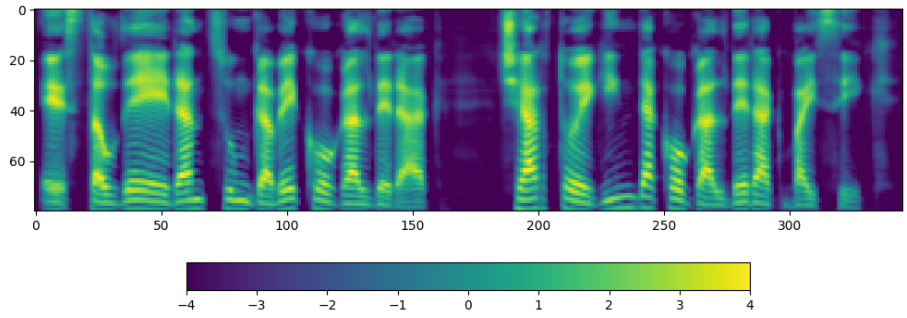



*Figura 14 Espectrograma predicho con modelo entrenado en castellano*

Con respecto al modelo entrenado en euskera, se identificó el problema de recorte de la señal y en este caso las señales de audio utilizadas en el entrenamiento sí han sido correctamente recortadas, resultando en un modelo mucho más estable capaz de sintetizar todas las frases correctamente. El error entre los espectrogramas predichos y los reales es mayor durante el entrenamiento, al modelo le cuesta más aprender del corpus, generando una señal un poco más ruidosa. El motivo de este incremento en el error se debe a un volumen de corpus ligeramente inferior al empleado en castellano.

La Figura 15 muestra el espectrograma predicho de una de las frases en euskera. En ella se observa cómo la señal comienza directamente sin ningún silencio al inicio, y el previamente mencionado suavizado.




Ez daude erantzun gabeko galderak,
txarto egindako galderak baizik.

*Figura 15 Espectrograma predicho con modelo entrenado en euskera*

# 10 Descripción de fases y tareas

El presente proyecto está compuesto de 4 fases bien diferenciadas. Cada fase tiene asociada una lista de tareas que se irán realizando de forma secuencial hasta finalizar por completo dicha fase. Para verificar el progreso realizado se establecen reuniones bisemanales, en las que se detalla el avance y se propone el trabajo a realizar durante las 2 semanas próximas.

**F-1: Formación previa y documentación**

Esta se trata de una de las fases más importantes. Durante el desarrollo de la misma se adquieren los conocimientos necesarios acerca de la tecnología empleada. También se incluye en esta fase la elección del modelo utilizado para la síntesis de voz. Dada la importancia de esta tarea se considera oportuno invertir aproximadamente un mes antes de comenzar con la puesta en marcha del proyecto:

**T1.1 Lectura de bibliografía**

**T1.2 Selección de modelo a implementar**

**T1.3 Instalación de dependencias**

**F-2: Replicación de resultados en inglés**

Con propósito de adquirir experiencia en el modelo a utilizar durante el proyecto, se realiza una fase de replicación del artículo[15] . Durante esta fase se adquiere conocimiento sobre las características de los datos a introducir al sistema, los parámetros modificables junto a su impacto sobre el resultado final y la supervisión durante las tareas de entrenamiento y síntesis:

**T2.1 Entrenamiento Tacotron**

**T2.2 Síntesis Tacotron**

**T2.3 Entrenamiento WaveNet**

**T2.4 Síntesis WaveNet**

**F-3: Implementación del Corpus propio**

Una vez adquirida la experiencia necesaria en la fase 2, se deben repetir todos los pasos realizados utilizando el Corpus propio, con todas las complicaciones que puedan surgir. Por ello, la fase 3 presenta unas tareas idénticas a la fase previa, pero incluye una tarea inicial muy importante que consiste en la adaptación del corpus al modelo:

**T3.1 Adaptación del Corpus al modelo**

**T3.2 Entrenamiento Tacotron**

**T3.3 Síntesis Tacotron**

**T3.4 Entrenamiento WaveNet**

**T3.5 Síntesis WaveNet**

**F-4: Gestión del proyecto**

Durante el desarrollo del proyecto se establece una tarea genérica destinada a cubrir la documentación que será necesario redactar para cubrir los avances que vayan realizándose. Esta tarea se desarrollará durante todo el proyecto y se extenderá brevemente al final:

**T4.1 Documentación del proyecto**

A continuación se definen los tiempos estimados de finalización de cada tarea:

| Tarea | Tiempo estimado | Tarea | Tiempo estimado |
|---|---|---|---|
| **T1.1** | 14 días | **T3.1** | 14 días |
| **T1.2** | 7 días | **T3.2** | 11 días |
| **T1.3** | 5 días | **T3.3** | 4 días |
| **T2.1** | 16 días | **T3.4** | 14 días |
| **T2.2** | 2 días | **T3.5** | 5 días |
| **T2.3** | 20 días | **T4.1** | 115 días |
| **T2.4** | 3 días | | |

*Tabla 5 Duración de tareas del proyecto*

En la siguiente página se adjunta el diagrama Gantt (Figura 16) que resume el tiempo y las fechas asociadas a cada tarea, dispuesto en horizontal.

# 11 Diagrama GANTT

## DIAGRAMA GANTT

| Nombre de la tarea | Fecha incio | Fecha fin | Duración | Porcentaje |
|---|---|---|---|---|
| **Formación previa y preparación** | | | | |
| Lectura de bibliografía | 1/02 | 15/02 | 14 | 100% |
| Selección de modelo a implementar | 15/02 | 22/02 | 7 | 100% |
| Instalación de dependencias | 22/02 | 27/02 | 5 | 100% |
| **Replicación de resultados en inglés** | | | | |
| Entrenamiento Tacotron | 27/02 | 13/03 | 16 | 100% |
| Síntesis Tacotron | 13/03 | 15/03 | 2 | 100% |
| Entrenamiento WaveNet | 18/03 | 8/04 | 20 | 100% |
| Síntesis WaveNet | 9/04 | 12/04 | 3 | 100% |
| **Implementación Corpus propio** | | | | |
| Adaptación del Corpus al modelo | 15/04 | 29/04 | 14 | 100% |
| Entrenamiento Tacotron | 29/04 | 10/05 | 11 | 100% |
| Síntesis Tacotron | 13/05 | 17/05 | 4 | 100% |
| Entrenamiento WaveNet | 17/05 | 31/05 | 14 | 100% |
| Síntesis WaveNet | 31/05 | 5/06 | 5 | 100% |
| **Gestión del proyecto** | | | | |
| Documentación | 15/04 | 5/06 | 50 | 100% |

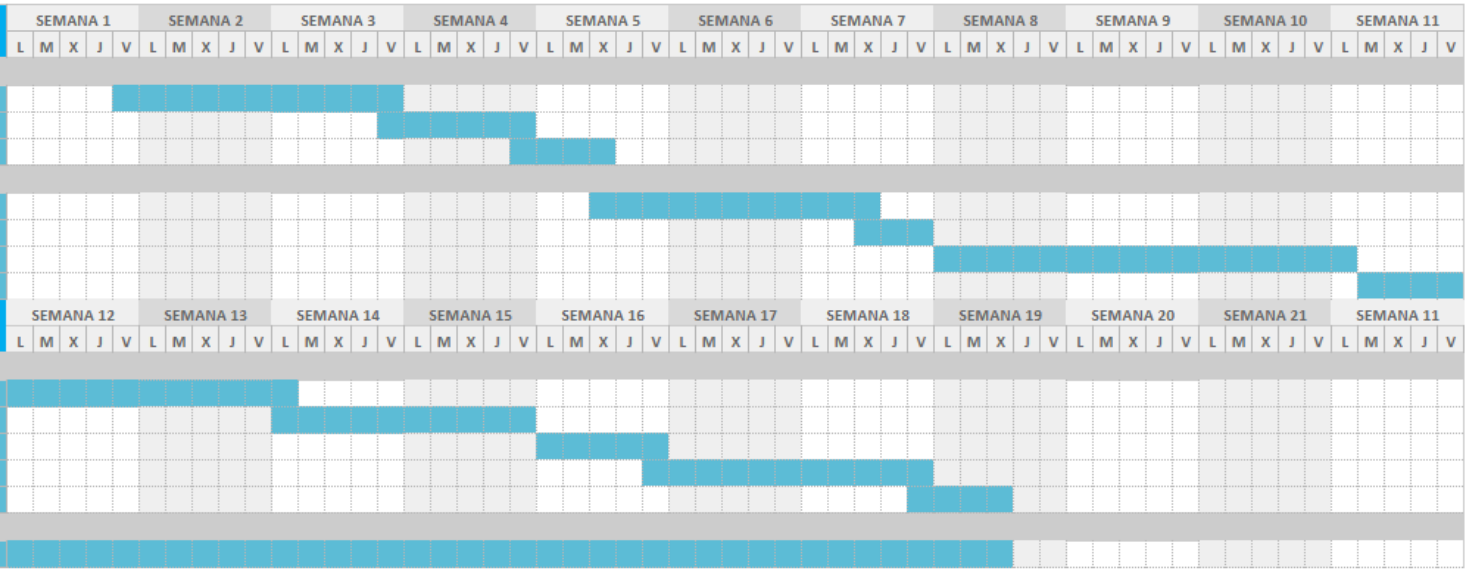



*Figura 16 Diagrama Gantt*

# 12 Aspectos económicos

A continuación se incluye el descargo de costes del proyecto. En primera instancia se presentarán los recursos empleados y posteriormente se hará una descripción detallada de los costes destinados a cada concepto, separando recursos humanos de materiales y terminando con el coste completo del proyecto.

## 12.1 Recursos empleados

En cuanto a los recursos humanos empleados en el proyecto, se incluye la participación de un Ingeniero de Telecomunicaciones junior junto a una Directora de proyecto.

Los recursos materiales se tratan principalmente de recursos de audio y recursos computacionales necesarios para entrenar y sintetizar los modelos utilizados. Se ha utilizado material de audio para la escucha de las señales de voz generada. En cuanto a los recursos computacionales, la presencia de unidades de procesamiento gráficas potentes es indispensable. Las tarjetas gráficas empleadas se tratan de 3 Gigabyte GeForce GTX Titan X con un coste unitario de 1070 €. Además, es necesario añadir el coste del procesador y las distintas memorias que componen el servidor.

Dado que todos los programas empleados en el desarrollo del proyecto son gratuitos y de código libre, la única licencia que es necesario desembolsar es la de “Microsoft Office”, empleada para la redacción de toda la documentación.

Finalmente a todo ello se incluyen los gastos generados en material fungible, como material de oficina, recursos para la elaboración de informes, etc.

## 12.2 Recursos humanos

Dada la escasa necesidad de personal del proyecto este apartado será breve, sin embargo, el importe será el más representativo de cara al descargo de costes final:

| Cargo | Coste horario (€/h) | Tiempo empleado (h) | Subtotal (€) |
|---|---|---|---|
| Ing.Telecom Junior | 13 | 600 | 7800 |
| Dir. de proyecto | 25 | 60 | 1500 |
| Subtotal Recursos Humanos | | | 9300 € |

*Tabla 6 Coste Recursos Humanos*

## 12.3 Recursos materiales

A continuación se detalla el descargo de costes de todos los recursos materiales:

| Concepto | Coste de adquisición (€) | Vida útil (años) | Tiempo de uso (meses) | Amortización (€) |
|---|---|---|---|---|
| GTX Titan X (x3) | 3210 | 5 | 4 | 214.0 |
| I7-6850K CPU | 300 | 5 | 4 | 20.0 |
| F. Alimentación | 300 | 5 | 4 | 20.0 |
| P.Base | 250 | 5 | 4 | 16.6 |
| Corsair LPX (x2) | 200 | 5 | 4 | 13.3 |
| Almacenamiento | 200 | 5 | 4 | 13.3 |
| Equipo de audio | 160 | 5 | 4 | 10.6 |
| Licencia Office | 90 | 5 | 4 | 6.0 |
| Subtotal Recursos Materiales | | | | 313.8 € |

*Tabla 7 Coste Recursos Materiales*

## 12.4 Gastos

En esta categoría se incluyen todos los conceptos no aprovechables a otros proyectos y destinados únicamente a la realización de este. Como puede apreciarse, los gastos no han sido muy elevados comparando con el resto de los recursos utilizados:

| Concepto | Subtotal (€) |
|---|---|
| **Material de oficina** | 30 |
| **Subtotal Gastos** | 30 € |

*Tabla 8 Gastos*

## 12.5 Presupuesto final

Por último, se incluyen los costes totales del proyecto:

| Concepto | Total (€) |
|---|---|
| **Recursos Humanos** | 9300 |
| **Recursos Materiales** | 313.8 |
| **Gastos** | 30 |
| **Total** | **9643.8 €** |

*Tabla 9 Presupuesto final*

Así pues, el coste total del proyecto asciende a 9643.8€.

# 13 Conclusiones

El objetivo de este proyecto consistía en sintetizar voces humanas empleando la tecnología desarrollada por Google, Tacotron-2. Las voces debían ser tanto de mujer como de hombre, en castellano y en euskera, y además debían ser inteligibles y naturales. Para ello, era necesario realizar las adaptaciones necesarias tanto sobre la implementación de Tacotron utilizada como sobre el corpus utilizado.

Tras finalizar el proyecto, puede afirmarse que los objetivos establecidos al inicio del mismo se han visto satisfechos. Las voces sintéticas generadas reproducen correctamente la prosodia de los hablantes y las características distintivas que presentan el castellano y el euskera con respecto a otras lenguas.

A pesar de no haber obtenido un resultado positivo empleando WaveNet, se ha identificado el problema, plasmando la solución del mismo en la presente memoria. Un problema común a los proyectos basados en aprendizaje profundo es la falta de tiempo de entrenamiento así como la falta de volumen de datos con el que entrenar los modelos. La obtención de más datos de entrada es compleja, sin embargo, no se descarta la posibilidad de dar continuidad al proyecto para solucionar los problemas relacionados con la falta de tiempo de entrenamiento.

En materia personal, el interés hacia las técnicas de aprendizaje autónomo era alto, habiendo trabajado previamente en proyectos relacionados. Dicho interés se ha visto incrementado durante la realización de este trabajo. En el ámbito de las tecnologías del habla, los conocimientos previos al respecto no eran muy elevados, pero las fases previas del proyecto permitieron obtener la instrucción necesaria para el desarrollo del trabajo, generando también un interés muy elevado hacia las mismas.

# 14 Fuentes de información

[1] Taylor, P. (2009). Text-to-speech synthesis. Text-to-Speech Synthesis (Vol. 9780521899). Cambridge: Cambridge University Press. https://doi.org/10.1017/CBO9780511816338

[2] Google AI Blog: Tacotron 2: Generating Human-like Speech from Text. (n.d.). Retrieved June 9, 2019, from https://ai.googleblog.com/2017/12/tacotron-2-generating-human-like-speech.html

[3] Baidu. (2016). Baidu Research. Retrieved June 9, 2019, from http://research.baidu.com/Blog/index-view?id=91

[4] Schwarz, D. (2005). CURRENT RESEARCH IN CONCATENATIVE SOUND SYNTHESIS. In Proceedings of the International Computer Music Conference (pp. 9–12). Retrieved from https://www.semanticscholar.org/paper/Current-Research-in-concatenative-sound-synthesis-Schwarz/16c426ea5bbf28ddcb666176a7d0a179a0c1b35a

[5] Campbell, N., & Black, A. W. (1997). Prosody and the Selection of Source Units for Concatenative Synthesis. In Progress in Speech Synthesis (pp. 279–292). New York, NY: Springer New York. https://doi.org/10.1007/978-1-4612-1894-4_22

[6] Tokuda, K., Nankaku, Y., Toda, T., Zen, H., Yamagishi, J., & Oura, K. (2013). Speech synthesis based on hidden Markov models. Proceedings of the IEEE, 101(5), 1234–1252. https://doi.org/10.1109/JPROC.2013.2251852

[7] Ze, H., Senior, A., & Schuster, M. (2013). Statistical parametric speech synthesis using deep neural networks. In ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings (pp. 7962–7966). https://doi.org/10.1109/ICASSP.2013.6639215

[8] Wu, Z., Watts, O., & King, S. (2016). Merlin: An Open Source Neural Network Speech Synthesis System. In 9th ISCA Speech Synthesis Workshop (Vol. 2016, pp. 202–207). https://doi.org/10.21437/ssw.2016-33

[9] Fan, Y., Qian, Y., Xie, F., & Soong, F. K. (2014). TTS synthesis with bidirectional LSTM based Recurrent Neural Networks. In Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH (pp. 1964–1968). Retrieved from https://www.isca-speech.org/archive/archive_papers/interspeech_2014/i14_1964.pdf

[10] Oord, A. van den, Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., … Kavukcuoglu, K. (2016). WaveNet: A Generative Model for Raw Audio. Retrieved from http://arxiv.org/abs/1609.03499

[11] Sercan Arık, Mike Chrzanowskii, Adam Coates, Gregory Diamos, Andrew Gibiansky, Yongguo Kang, Xian Li, John Miller, Andrew Ng, Jonathan Raiman, Shubho Sengupta, M. S. (2013). Deep Voice: Real-time Neural Text-to-Speech Sercan. Computers and Mathematics with Applications, 65(10), 1471–1482. https://doi.org/10.1016/j.camwa.2012.12.009

[12] Arik, S., Diamos, G., Gibiansky, A., Miller, J., Peng, K., Ping, W., … Zhou, Y. (2017). Deep Voice 2: Multi-Speaker Neural Text-to-Speech. Retrieved from http://arxiv.org/abs/1705.08947

[13] Ping, W., Peng, K., Gibiansky, A., Arik, S. O., Kannan, A., Narang, S., … Miller, J. (2017). Deep Voice 3: Scaling Text-to-Speech with Convolutional Sequence Learning. Retrieved from http://arxiv.org/abs/1710.07654

[14] Wang, Y., Skerry-Ryan, R. J., Stanton, D., Wu, Y., Weiss, R. J., Jaitly, N., … Le, Q. (2017). Tacotron: Towards end-To-end speech synthesis. In Proceedings of the Annual

Conference of the International Speech Communication Association, INTERSPEECH (Vol. 2017-Augus, pp. 4006–4010). https://doi.org/10.21437/Interspeech.2017-1452

[15] Shen, J., Pang, R., Weiss, R. J., Schuster, M., Jaitly, N., Yang, Z., … Wu, Y. (2018). Natural TTS Synthesis by Conditioning WaveNet on MEL Spectrogram Predictions. In ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings (Vol. 2018-April, pp. 4779–4783). https://doi.org/10.1109/ICASSP.2018.8461368

[16] Griffin, D. W., & Lim, J. S. (1984). Signal Estimation from Modified Short-Time Fourier Transform. IEEE Transactions on Acoustics, Speech, and Signal Processing, 32(2), 236–243. https://doi.org/10.1109/TASSP.1984.1164317

[17] Takahashi, D. (2018). Newzoo: Games market expected to hit $180.1 billion in revenues in 2021 | VentureBeat. Retrieved June 9, 2019, from https://venturebeat.com/2018/04/30/newzoo-global-games-expected-to-hit-180-1-billion-in-revenues-2021/

[18] Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural Machine Translation by Jointly Learning to Align and Translate. Retrieved from https://arxiv.org/pdf/1409.0473.pdf

[19] Spectrogram Feature prediction network · Rayhane-mamah/Tacotron-2 Wiki. (n.d.). Retrieved June 9, 2019, from https://github.com/Rayhane-mamah/Tacotron-2/wiki/Spectrogram-Feature-prediction-network

[20] Google. (n.d.). TensorBoard: Visualizing Learning | TensorFlow Core | TensorFlow. Retrieved June 10, 2019, from https://www.tensorflow.org/guide/summaries_and_tensorboard

[21] What are intuitive explanations of the Griffin Lim Algorithm? - Quora. (n.d.). Retrieved June 11, 2019, from https://www.quora.com/What-are-intuitive-explanations-of-the-Griffin-Lim-Algorithm

[22] Van den Oord, Aaron, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, and Alex Graves. "Conditional image generation with pixelcnn decoders." In *Advances in neural information processing systems*, pp. 4790-4798. 2016.

# 15 Anexos

## 15.1 Anexo I: Escala Mean Opinion Score (MOS)

La escala MOS es utilizada para medir la calidad de un estímulo o un sistema, principalmente en el campo de las telecomunicaciones y la *"Quality of Experience" (QoE).* Se trata de una media aritmética entre los valores existentes en una escala predefinida que un sujeto asigna a un sistema o estímulo basado en su opinión acerca de la calidad del mismo.

La escala suele estar comprendida entre el 1 y el 5 siendo 1 la calidad más pobre y 5 la más alta, de acorde a la siguiente tabla:

| Calificación | Etiqueta |
|---|---|
| **5** | Excelente |
| **4** | Buena |
| **3** | Decente |
| **2** | Pobre |
| **1** | Mala |

*Tabla 10 Escala MOS*

Realizando la prueba sobre distintos sujetos, la calidad de un sistema o estímulo sería:

$$MOS = \frac{\sum_{n=1}^{N} R_n}{N}$$

Donde R son las calificaciones que cada sujeto individualmente asigna al sistema y N el número total de sujetos que participan en la evaluación.

Inicialmente la escala MOS se inició impulsada por la industria telefónica para asignar una puntuación subjetiva sobre la calidad de las llamadas. Esta metodología se ha

seguido usando ampliamente en la industria telefónica hasta el punto de ser estandarizada por la ITU-T según la recomendación P.800, la cual especifica las condiciones en las que se debe realizar la evaluación, desde el volumen de escucha en dB hasta las condiciones acústicas de la sala.

## 15.2 Anexo II: Gráficas de entrenamiento Tacotron 2.

Este anexo recoge las gráficas que han sido supervisadas durante el entrenamiento del modelo utilizando ambos corpus. Para no colapsar el apartado con las gráficas generadas en todas las iteraciones, únicamente se mostrarán las más representativas.

En la Figura 17 se muestran las gráficas más importantes a supervisar durante el proceso. La herramienta Tensorboard [20] permite realizar gráficas a partir de "*logs*" generados durante el entrenamiento. Arriba a la derecha se muestra el error cometido en cada iteración. Durante el entrenamiento se debe buscar punto de inflexión en el que esta gráfica deja de descender, ya que a partir de ese punto puede considerarse que el modelo está dejando de aprender. Abajo a la derecha se muestra el aprendizaje a la hora de generar el "*token*" que emplea el Decoder para identificar cuándo parar de decodificar. Es preciso que esta gráfica se encuentre estable entorno al cero al detener el entrenamiento.

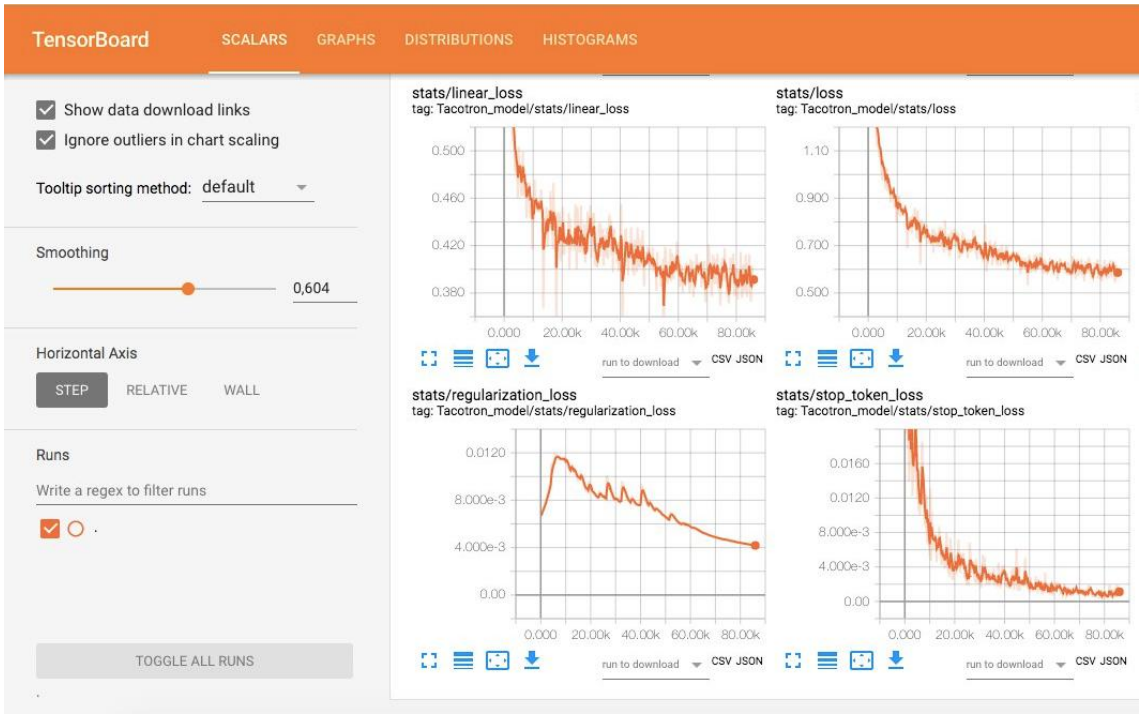



*Figura 17 Panel de Tensorboard con entrenamiento en castellano*

La Figura 18 muestra la alineación Encoder-Decoder sobre la última iteración realizada. La alineación se trata de cómo de bien escoge el Decoder a qué vectores del Encoder debe prestar atención para generar la salida. Una alineación perfecta genera una línea diagonal.

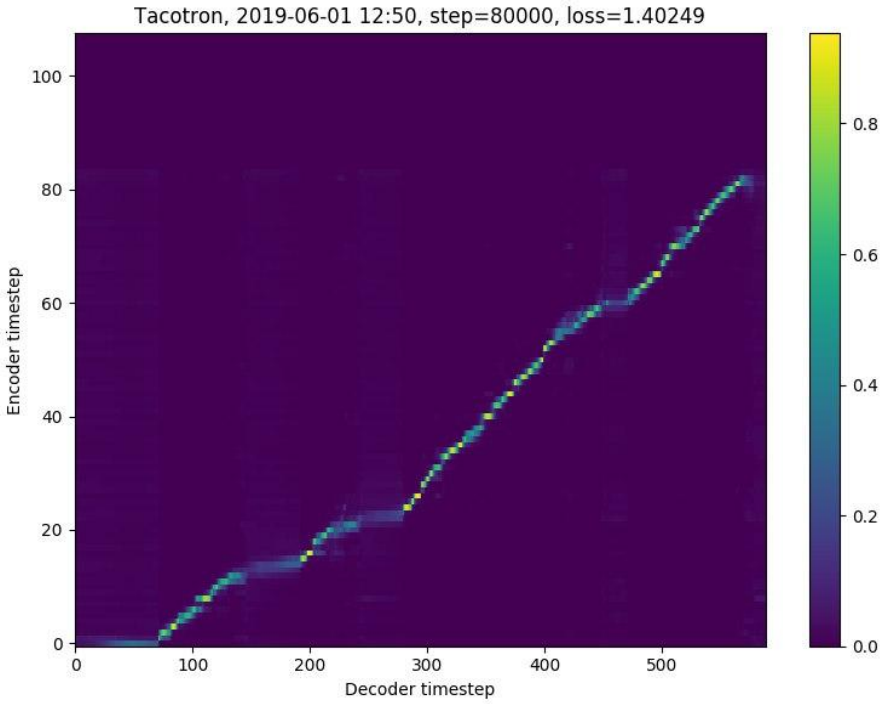
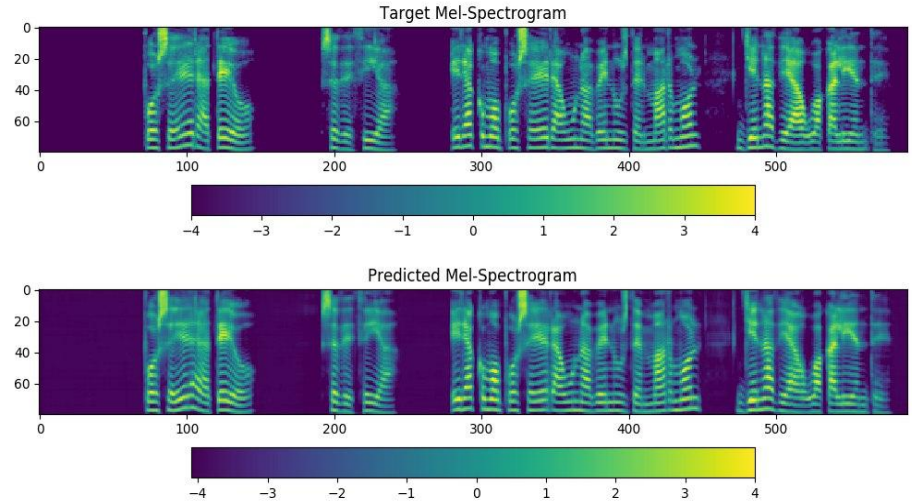



*Figura 18 Gráfica de alineación*

El resultado, como puede apreciarse, es bueno pero podría conseguirse una alineación mejor realizando un preprocesado que elimine mejor los silencios de inicio de las frases.

La Figura 19 muestra el espectrograma de mel real y el predicho por Tacotron sobre la misma frase mostrada en la gráfica de alineación. Realizando una comparación entre ambas gráficas, puede observarse como la alineación del modelo falla en los silencios.




Tacotron, 2019-06-01 12:50, step=80000, loss=1.40249

*Figura 19 Espectrograma real y predicho*

## 15.3 Anexo III: Fichero hparams.py

A continuación se recogen parte de los parámetros más importantes y que han sido modificados para adaptar el modelo al corpus y condiciones de computación disponibles. La lista completa de parámetros a modificar se encuentra en el repositorio de Rayhane Mamah.

```
import numpy as np
import tensorflow as tf

hparams = tf.contrib.training.HParams(

      cleaners='basic_cleaners',
      tacotron_num_gpus = 1,
      WaveNet_num_gpus = 1,
      split_on_cpu = True,
      num_mels = 80,
      num_freq = 513,
      rescale = True,
      rescaling_max = 0.999,

      #Mel spectrogram
      n_fft = 1024,
      hop_size = 200,
      win_size = 800,
      sample_rate = 16000,
      frame_shift_ms = None,
      magnitude_power = 2.,

      trim_silence = True,
      trim_fft_size = 2048,
      trim_hop_size = 512,
      trim_top_db = 40,

      signal_normalization = True,
      normalize_for_WaveNet = True,
      clip_for_WaveNet = True,

      preemphasize = True,
      preemphasis = 0.97,

      min_level_db = -100,
      ref_level_db = 20,
      fmin = 95,
      fmax = 7600,

      #Input parameters
      embedding_dim = 512,

      #WaveNet
```

```
        input_type= "raw",
        quantize_channels=2**16,

        #Model Losses parameters
        cdf_loss = True,

        #Upsampling parameters
        cin_channels = 80,
        upsample_activation = 'Relu',
        upsample_scales = [10, 20],

        #Tacotron Training
        tacotron_random_seed = 5339,
        tacotron_data_random_state = 1234,

        #Learning rate schedule
        tacotron_decay_learning_rate = True,
        tacotron_start_decay = 40000,
        tacotron_decay_steps = 18000,
        tacotron_decay_rate = 0.5,
        tacotron_initial_learning_rate = 1e-3,
        tacotron_final_learning_rate = 1e-4,

        #Speaker adaptation parameters
        tacotron_fine_tuning = False,

        #WaveNet Training
        WaveNet_random_seed = 5339,
        WaveNet_data_random_state = 1234,

        #train/test split ratios, mini-batches sizes
        WaveNet_batch_size = 8,
        WaveNet_synthesis_batch_size = 10 * 2,

        #Learning rate schedule
        WaveNet_lr_schedule = 'exponential',
        WaveNet_learning_rate = 1e-3,
        WaveNet_decay_rate = 0.5,
        WaveNet_decay_steps = 200000,

        #Tacotron-2 integration parameters
        train_with_GTA = True,
        )
```