Universidad de La Habana
Facultad de Matemática y Computación

Sintetizador de Voz en Español con Voces Cubanas

Autor:
Sandra Martos Llanes

Tutores:

Lic. Katy Castillo Rosado
Dr.Cs. Flavio Reyes Díaz

Trabajo de Diploma
presentado en opción al título de
Licenciado en Ciencia de la Computación

Fecha

github.com/username/repo

Dedicación

Agradecimientos

Agradecimientos

Opinión del tutor

Opiniones de los tutores

Resumen

Resumen en español

Abstract

Resumen en inglés

Índice general

Índice de figuras

Ejemplos de código

Introducción

La era de la informática conversacional está cambiando la forma en la que los usuarios interactuan con sus dispositivos: el Asistente de Google busca en Internet y lee las instrucciones sobre cómo preparar un pastel, Siri guía en la búsqueda de un lugar desconocido, las líneas automatizadas de servicio al cliente operan sin necesidad de esperas o botones. Uno de los primeros problemas que alguien se plantearía es la comunicación con los dispositivos, y es que no es posible determinar de qué forma va a interactuar una persona con estos. Se pueden hacer aproximaciones y suposiciones, pero cada persona puede decidir operar de una forma diferente, y sería imposible que un actor de voz grabase infinitas respuestas a las ramas de conversación que pueden surgir de una simple pregunta. En este punto entra la síntesis de voz, si en vez de unas grabaciones, el dispositivo pudiese sintetizar una voz humana, podría responder y hablar con la persona, entablando una especie de conversación.

La síntesis de voz es la producción artificial del habla humana. Se han diseñado diferentes sistemas para este propósito, llamados sintetizadores de voz y pueden ser implementados tanto en hardware como en software. Un sistema de conversión texto a voz (TTS, por sus siglas en inglés, *text to speech*) o sintetizador de voz, recibe como datos de entrada una frase escrita y como resultado produce una voz audible que reproduce la frase de entrada . Este sistema de conversión, se compone básicamente por dos componentes: un modelo que predice generalmente en forma de espectograma, la mejor pronunciación posible de cualquier texto dado, y un codificador de voz que, a partir del espectograma anterior produce ondas sonoras de voz.

El proceso de convertir texto a voz requiere un conocimiento detallado en diferentes campos de la ciencia. Si se quisiera construir un sistema TTS desde cero, se tendrían que estudiar los siguientes temas:

- Lingüística, el estudio científico del lenguaje. Para sintetizar un habla coherente, los sistemas TTS necesitan reconocer cómo un hablante humano pronuncia el lenguaje escrito; esto requiere conocimientos de lingüística hasta el nivel del

fonema[1]. Para lograr un TTS verdaderamente realista, el sistema también necesita predecir la prosodia apropiada,que incluye elementos del habla más allá del fonema, como acentos, pausas y entonación.

- Procesamiento digital de señales de voz: en el contexto de los sistemas TTS se utilizan diferentes representaciones de características que describen la señal del habla, lo que hace posible entrenar diversos modelos para generar una nueva voz.

- Inteligencia artificial, especialmente el aprendizaje profundo, un tipo de aprendizaje automático que utiliza una arquitectura informática llamada red neuronal profunda(DNN, del inglés Deep Neuronal Network). Una DNN es un modelo computacional inspirado en el cerebro humano,se conforma por redes complejas de procesadores, cada uno de los cuales realiza una serie de operaciones antes de enviar su salida a otro procesador. Una DNN aprende la mejor vía de procesamiento para lograr resultados deseados. Este modelo tiene una gran potencia informática, lo que lo hace ideal para manejar la gran cantidad de variables necesarias para la síntesis de voz de alta calidad.

Tipos de tecnologías TTS

Hasta la actualidad se han desarrollado variadas tecnologías TTS, las cuales operan de maneras distintas. Entre las más dominantes se encuentran:

1. Síntesis de formantes[1][2] y síntesis articulatoria[3]:
   Los primeros sistemas TTS empleaban tecnologías basadas en reglas, como la síntesis de formantes y la síntesis articulatoria, que lograron resultados similares a través de estrategias ligeramente diferentes. A partir de una grabación realizada a un hablante, se extrajeron características acústicas de este: formantes, cualidades definitorias de los sonidos del habla, en síntesis de formantes; y forma de articulación (nasal, oclusiva,vocal,etc.) en síntesis articulatoria. Luego, se programarían reglas que recrearan estos parámetros con una señal de audio digital. Este TTS era bastante robótico; y estos enfoques necesariamente abstraen gran parte de la variación que se encontrará en el habla humana, aspectos como la variación de tono, porque solo permiten a los programadores escribir reglas para unos pocos parámetros a la vez.

2. Síntesis de difonos:
   El próximo gran desarrollo en el campo TTS se llama síntesis de difonos, se inició en la década de 1970 y todavía era de uso popular durante los últimos

[1]las unidades de sonido que combinadas producen el habla, como el sonido /t/ en tierra

años del siglo XX. La síntesis de difonos crea un habla de máquina mediante la combinación de difonos, combinaciones de fonemas de una sola unidad, y las transiciones de un fonema al siguiente; es decir, no solo la /t/ en la palabra tierra sino la /t/ más la mitad del siguiente sonido /i/.

Los sistemas TTS basados en síntesis de difonos incluyen también modelos que predicen la duración y el tono de cada difono para una entrada dada, primero se conectan las señales de difono y luego se procesa esta señal para corregir el tono y la duración. El resultado es un discurso sintético con un sonido más natural que el que crea la síntesis de formantes, pero aún está lejos de ser perfecto, y tiene pocas ventajas sobre cualquier otro acercamiento.

3. Síntesis de selección de unidades:
   La síntesis de selección de unidades constituye un enfoque ideal para los motores TTS de bajo impacto en la actualidad. Cuando la síntesis de difonos añadió duración y el tono apropiado a través de un segundo sistema de procesamiento, la síntesis de selección de unidad omite ese paso: se inicia con una gran base de datos grabados del habla, alrededor de 20 horas o más, y se seleccionan los fragmentos de sonido que ya tienen la duración y el tono deseado. La síntesis de selección de unidades proporciona un habla similar a la humana sin mucha modificación de la señal, pero sigue siendo identificablemente artificial. Probablemente el audio de salida de la mejor selección de unidades, sea indistinguible de las voces humanas reales, especialmente en contextos con sistemas TTS. Sin embargo, una mayor naturalidad requiere de bases de datos de selección de unidades muy grandes, en algunos sistemas llegando a ser de gigabytes de datos grabados, representando docenas de horas de voz.

4. Síntesis neuronal:
   La tecnología de las redes neuronales profundas(DNN) es la que impulsa los avances actuales en el campo TTS, y es clave para la obtención de resultados mucho más realistas. Al igual que sus predecesores, el TTS neuronal comienza con grabaciones de voz; la otra entrada es texto, el guión escrito que su locutor de origen utilizó para crear esas grabaciones. Esas entradas alimentan una red neuronal profunda y se aprende el mejor mapeo posible entre un bit de texto y las características acústicas asociadas.

   Una vez que el modelo esté entrenado, podrá predecir sonido realista para nuevos textos: con un modelo TTS neuronal entrenado, junto con un codificador de voz entrenado con los mismos datos, el sistema puede producir un habla que es notablemente similar a la del locutor de origen cuando se expone a prácticamente cualquier texto nuevo. Esa similitud entre la fuente y la salida es la razón por la que el TTS neuronal a veces se denomina “clonación de voz”.

Hay todo un grupo de métodos de procesamiento de señales que pueden ser utilizados para alterar la voz sintética resultante y no se asemeje al locutor fuente. En la actualidad, las principales investigaciones se enfocan en lograr voces sintéticas con una calidad de audio cada vez más realista.

Entre las aplicaciones que hacen uso del TTS, se encuentran:

- Sistemas de respuesta de voz interactiva conversacional (IVR), como en los centros de llamadas de servicio al cliente.
- Aplicaciones de comercio de voz, como comprar en un dispositivo Amazon Alexa.
- Herramientas de navegación y guía por voz, como aplicaciones de mapas GPS.
- Dispositivos domésticos inteligentes y otras herramientas de Internet de las cosas (IoT) habilitadas por voz.
- Asistentes virtuales independientes, como Siri de Apple.
- Soluciones de publicidad y marketing experiencial, como anuncios de voz interactivos en servicios de transmisión de música.
- Desarrollo de videojuegos.
- Videos de marketing y formación de la empresa que permiten a los creadores cambiar las voces en off sin identificar al locutor.

Problemática

Existe un conjunto de sistemas TTS que se enmarcan bajo una licencia de software libre:

- Festival y Festvox
- Plataforma MaryTTS
- Sistema TTS
- SV2TTS
- Mozilla-TTS
- COQUI -TTS

A partir del estudio parcial de las plataformas de código abierto utilizadas para el desarrollo de conversores de texto a voz, fue posible comprobar que la mayoría se encuentran basadas en la síntesis neuronal teniendo en cuenta que alcanza mejores resultados. Estas plataformas brindan modelos previamente entrenados para idiomas específicos como inglés, francés, alemán, etc. Muy pocas presentan un modelo en español, y las que lo hacen solo poseen uno, con acento de voz española o voz neutra.

DATYS, es una empresa de desarrollo de software, que como parte de sus soluciones requiere un sintetizador de voz en español con acento propio de nuestro país; esto formará parte de un proyecto que consiste en el desarrollo del primer asistente virtual cubano.

Objetivo

Objetivo General

Desarrollar un sintetizador de texto a voz con voces cubanas.

Objetivos específicos

1. Analizar en profundidad los métodos y plataformas existentes para la síntesis de voz, principalmente las que trabajan sobre la síntesis neuronal, y seleccionar el más adecuado a utilizar para las aplicaciones de DATYS.

2. Diseñar y conformar una base de datos en español, con con voces cubanas para el entrenamiento de los modelos y muestras de voz que se desea generar.

3. Reentrenar el modelo basado en síntesis neuronal seleccionado, para su ajuste al estilo de voz cubana.

4. Evaluar el modelo entrenado y analizar los resultados obtenidos.

Capítulo 1

Sistemas para la síntesis texto a voz

Las redes neuronales han sido las encargadas de los avances actuales en el campo TTS, por tanto esta será la línea seguida en la investigación.

Para lograr la transformación de texto a voz, se siguen dos enfoques distintos. El primero es un sistema de dos estapas: una combinación de dos redes neuronales, una para convertir de texto a espectograma de mel, y luego otra que transforma el espectograma en onda sonora; y el segundo es un modelo de extremo a extremo.

1.1. Sistemas de dos etapas

El paradigma predominante en la conversión de texto a voz es la síntesis en dos etapas, es decir, primero, producir espectrogramas a escala de mel a partir del texto y, luego las ondas de sonido reales con un modelo de codificador de voz(*VoCoder*. La representación acústica de bajo nivel, espectograma de mel, es la utilizada como nexo entre las dos componentes.

1.1.1. Modelos TTS

Tacotron

Tacotron es un modelo TTS de tipo secuencia a secuencia con un paradigma de atención. Este modelo toma caracteres como entrada y devuelve un espectograma sin

procesar usando técnicas para mejorar un modelo seq2seq[1] vainilla[2]. Dado un par <texto,audio>, Tacotron puede ser entrenado desde cero con una inicialización aleatoria, y no requiere alineación a nivel de fonema.

La columna vertebral de Tacotron es un modelo seq2seq con atención, que toma caracteres como entrada, y devuelve el correspondiente espectograma sin procesar, para luego pasarlo al modelo o algoritmo que sinteiza la voz. En el centro de todo esto se encuentra un codificador,un decodificador basado en atención y una red de post procesamiento.

Tacotron se basa en cuadros, o *frames*, por lo que la inferencia es sustancialmente más rápida que los métodos autorregresivos a nivel de muestra. A diferencia de otras tecnologías TTS más antiguas, Tacotron no necesita características lingüísticas diseñadas a mano ni componentes complejos como un alineador de Modelo Markov oculto(HMM). Este modelo realiza una normalización de texto simple[4].

Tacotron 2

Tacotron 2 es similar al anteriormente mencionado Tacotron; es una red recurrente de predicción de características, de tipo secuencia a secuencia con atención, que mapea incrustaciones(embeddings) de caracteres en espectrogramas a escala de mel.

Un espectrograma de frecuencia de mel está relacionado con el espectograma de frecuencia lineal, es decir, la magnitud de la transformada de Fourier de tiempo corto (STFT). Se obtiene aplicando una transformada no lineal al eje de frecuencia de la STFT, inspirado en respuestas calificadas por el sistema auditivo humano, y resume el contenido de frecuencia con menos dimensiones.

El uso de una escala de frecuencia auditiva de este tipo tiene el efecto de enfatizar detalles en frecuencias más bajas, que son fundamentales para la inteligibilidad del habla, al mismo tiempo que se resta importancia a los detalles de alta frecuencia, que están dominados por ráfagas de ruido y generalmente no necesitan ser modelados con alta fidelidad. Debido a estas propiedades, las características derivadas de la escala de mel se han utilizado como representación base para el reconocimiento de voz durante muchas décadas.

Para Tacotron2 los espectrogramas de mel se calculan a través de un transformada de Fourier de tiempo corto (STFT) utilizando un tamaño de cuadro de 50 ms, 12,5 ms de salto de cuadro y una función de ventana de Hann. Se transforma la magnitud

[1]Seq2Seq se basa en el paradigma codificador-decodificador. El codificador codifica la secuencia de entrada, mientras que el decodificador produce la secuencia de destino. codificador

[2]En informática, vainilla es el término utilizado cuando el *software* o los algoritmos, no se emplean a partir de su forma original.

de la STFT a la escala de mel usando un banco de filtros de 80 canales mel que abarca de 125 Hz a 7,6 kHz, seguido de una compresión de registro de rango dinámico. Antes de la compresión de registros, las magnitudes de salida del banco de filtros se recortan a un valor mínimo de 0,01 para limitar el rango dinámico en el dominio logarítmico.[5]

La red del modelo en cuestión está compuesta por un codificador y un decodificador con atención. El codificador convierte una secuencia de caracteres en una representación oculta que alimenta al decodificador para predecir un espectrograma.

Los caracteres de entrada se representan utilizando una incrustación de caracteres 512-dimensional, que se pasan a través de una pila de 3 capas convolucionales, cada una de las cuales contiene 512 filtros con organización 5×1, es decir, donde cada filtro abarca 5 caracteres, seguido de la normalización por lotes y activaciones de ReLU[5]. Como en Tacotron, estas capas convolucionales modelan el contexto a largo plazo en la secuencia de caracteres de entrada. La salida de la capa convolucional final se pasa a una sola capa bidireccional LSTM que contiene 512 unidades (256 en cada dirección) para generar las características codificadas.

La salida del codificador es consumida por una red de atención que resume la secuencia codificada completa como un vector de contexto de longitud fija para cada paso de salida del decodificador. Se usa la atención sensible a la ubicación de [6], que extiende el mecanismo de atención aditiva [7] para usar pesos de atención acumulativos de anteriores pasos de tiempo del decodificador como una funcionalidad adicional. Esto anima al modelo a seguir adelante consistentemente a través de la entrada, mitigando los posibles modos de falla donde algunas subsecuencias son repetidas o ignoradas por el decodificador. Las probabilidades de atención se calculan después de proyectar las entradas y funciones de localización a representaciones ocultas de 128 dimensiones. Las funcionalidades de localización se calculan utilizando 32 filtros de convolución 1-D de longitud 31[5].

El decodificador es una red neuronal autorregresiva recurrente que predice un espectrograma de mel a partir de la secuencia de entrada codificada un fotograma a la vez. La predicción del paso de tiempo anterior es primero pasado a través de una pequeña red previa que contiene 2 capas completamente conectadas de 256 unidades ReLU ocultas. La pre-red actuando como un cuello de botella de información es esencial para el aprendizaje de la atención.

La salida de la pre-red y el vector de contexto de atención se concatenan y pasan a través de una pila de 2 capas *Long Short-Term Memory(LSTM)* unidireccionales con 1024 unidades. La concatenación de la salida LSTM y el vector de contexto de atención se proyecta a través de una transformación lineal para predecir el cuadro de espectrograma objetivo. Finalmente, se pasa el espectrograma de mel predicho a

través de una post-red convolucional de 5 capas que predice un residuo a agregar a la predicción para mejorar la reconstrucción general. Cada capa post-net está compuesta por 512 filtros con forma 5×1 con lote normalización, seguida de activaciones de tanh en todas las capas excepto en la final[5].

En paralelo a la predicción de cuadros de espectrograma, la concatenación de salida del decodificador LSTM y el contexto de atención se proyectan a un escalar y pasa a través de una activación sigmoidea para predecir la probabilidad de que la secuencia de salida se haya completado. La predicción de este "token de parada" se usa durante la inferencia para permitir que el modelo determine dinámicamente cuándo terminar la generación en lugar de generar siempre por una duración fija. Específicamente, la generación se completa en el primer fotograma para el que esta probabilidad supera un umbral de 0,5.

Las capas convolucionales en la red se regularizan usando abandono(dropout) con probabilidad 0.5, y las capas LSTM son regularizadas usando desconexión(zoneout) con probabilidad 0.1. Para introducir la variación de salida en tiempo de inferencia, se aplica dropout con probabilidad 0.5 solo a capas en la pre-red del decodificador autorregresivo[5].

En contraste con el Tacotron original, este modelo utiliza una construcción más simple, usando LSTM vainilla y capas convolucionales en el codificador y decodificador en lugar de pilas CBHG y capas recurrentes GRU. No se usa un "factor de reducción", es decir, cada paso del decodificador corresponde a un único cuadro de espectrograma.

Este sistema puede ser entrenado directamente desde un conjunto de datos sin depender de una compleja ingeniería de características, y logra calidad de sonido de última generación cercana a la del habla humana natural. Los resultados de Tacotron 2, constituyen un paso de avance sobre Tacotron y otros sistemas previos, sin embargo dejan aún espacio para mejoras.

Deep Voice 1, 2, 3

Deep Voice de Baidu[8][9] sentó las bases para los avances posteriores en la síntesis de voz de extremo a extremo. Consta de 4 redes neuronales diferentes que juntas forman un extremo de la canalización: un modelo de segmentación que localiza los límites entre fonemas,un modelo que convierte grafemas en fonemas, un modelo para predecir la duración de los fonemas y las frecuencias fundamentales, y un modelo para sintetizar el audio final.

Deep Voice 2 [9] se presentó como una mejora de la arquitectura original de Deep Voice. Si bien la canalización principal era bastante similar, cada modelo se creó desde cero para mejorar su rendimiento. Otra gran mejora fue la adición de compatibilidad

con varios hablantes.

Deep Voice 3 [9][10] es un rediseño completo de las versiones anteriores. Aquí se tiene un solo modelo en lugar de cuatro diferentes. Más específicamente, los autores propusieron una arquitectura de caracter a espectrograma completamente convolucional que es ideal para el cálculo paralelo. A diferencia de los modelos basados en RNN. También se experimentó con diferentes métodos de síntesis de forma de onda con WaveNet logrando los mejores resultados una vez más.

Modelos TTS con Transformers

Los transformadores(*transformers*), están dominando el campo del lenguaje natural desde hace un tiempo, por lo que era inevitable que ingresaran gradualmente al campo TTS. Los modelos basados en transformadores tienen como objetivo abordar dos problemas de los métodos TTS anteriores, como Tacotron2:

- Baja eficiencia durante el entrenamiento y la inferencia.
- Dificultad para modelar dependencias largas usando redes neuronales recurentes(RNN, por sus siglas en inglés).

La primera arquitectura basada en transformadores se introdujo en 2018 y reemplazó las RNN con mecanismos de atención de múltiples cabezales que se pueden entrenar en paralelo.

FastSpeech

FastSpeech, una novedosa red de avance basada en Transformer para generar espectrogramas de mel en paralelo para TTS; toma como entrada una secuencia de texto(fonema) y genera espectrogramas de mel de forma no autorregresiva. Adopta una red feed-forward basada en la autoatención[3] en Transformer y convolución de 1D.

El modelo resuelve problemas existentes en modelos TTS antiguos de la siguiente forma:

- A través de la generación de espectrogramas de mel paralelos, FastSpeech acelera enormemente el proceso de síntesis.
- El predictor de duración de fonemas asegura alineaciones estrictas entre un fonema y sus espectrogramas, lo que es muy diferente de las alineaciones de atención automáticas y suaves en los modelos autorregresivos. Por lo tanto,

[3]La autoatención permite a una red neuronal entender una palabra en el contexto de las palabras que la rodean.

FastSpeech evita los problemas de propagación de errores y alineaciones de atención incorrectas, lo que reduce la proporción de palabras omitidas y palabras repetidas.

- El regulador de longitud puede ajustar fácilmente la velocidad de la voz alargando o acortando la duración del fonema para determinar la duración de los espectrogramas de mel generados, y también puede controlar parte de la prosodia añadiendo pausas entre fonemas adyacentes.

La arquitectura para Fast Speech es una estructura de avance basada en la autoatención en Transformer y la convolución de 1D; se nombra esta estructura como Feed-Forward Transformer (FFT). Feed-Forward Transformer apila múltiples bloques FFT para la transformación de fonema a espectrograma de mel, con N bloques en el lado del fonema y N bloques en el lado del espectrograma de mel, con un regulador de longitud en el medio para cerrar la brecha de longitud entre el fonema y la secuencia del espectrograma de mel.

Posee un regulador de longitud que se utiliza para resolver el problema de la discordancia de longitud entre el fonema y la secuencia del espectrograma en el transformador de avance, así como para controlar la velocidad de la voz y parte de la prosodia. Finalmente un predictor de duración que genera un escalar, que es exactamente la duración prevista del fonema.

El entrenamiento de FastSpeech y de gran mayoría de los modelos TTS se realiza sobre un conjunto de datos que contiene clips de audios con sus correspondientes transcripciones de texto. Específicamente para FastSpeech, se divide al azar el conjunto de datos en 3 conjuntos: muestras para entrenamiento, muestras para validación y muestras para las pruebas.

El modelo FastSpeech puede casi coincidir con el modelo autoregresivo Transformer TTS en términos de calidad de voz, acelera la generación de espectrograma mel por 270x y la síntesis de voz de extremo a extremo por 38x, casi se elimina el problema de saltar y repetir palabras, y puede ajustar la velocidad de voz (0.5x-1.5x) sin problemas[11].

FastPitch

FastPitch es un modelo TTS feed-forward completamente paralelo basado en FastSpeech, condicionado por contornos de frecuencia fundamentales. El modelo predice contornos de tono durante la inferencia. Al alterar estas predicciones, el discurso generado puede ser más expresivo, coincidir mejor con la semántica del enunciado y, al final, ser más atractivo para el oyente.

Los modelos paralelos pueden sintetizar órdenes de magnitud de espectrogramas de mel más rápido que los autorregresivos, ya sea basándose en alineaciones externas o alineándose ellos mismos. El condicionamiento en la frecuencia fundamental también mejora la convergencia y elimina la necesidad de destilar el conocimiento de los objetivos del espectrograma de mel utilizados en FastSpeech.

La arquitectura del modelo, se basa en FastSpeech y se compone principalmente de dos pilas de transformadores alimentados hacia adelante, feed-forward(FFTr) . El primero opera en la resolución de los tokens de entrada, el segundo en la resolución de los cuadros de salida. [12].

Para el entrenamiento y la experimentación los parámetros del modelo siguen principalmente FastSpeech.

Glow-TTS

A pesar de la ventaja, los modelos TTS paralelos no se pueden entrenar sin la guía de modelos TTS autorregresivos como alineadores externos.

Glow-TTS[13] es un modelo generativo basado en flujo para TTS paralelo que no requiere de ningún alineador externo. Al combinar las propiedades de los flujos y la programación dinámica, el el modelo busca la alineación monótona más probable entre el texto y la representación latente del habla en sí misma. La arquitectura del modelo consiste en un codificador que sigue la estructura del codificador de *Transformer TTS* con pequeñas modificaciones, un predictor de duración, que tiene una estructura y configuración como la de FastSpeech, y finalmente un decodificador basado en flujo, que es la parte fundamental del modelo.

Se demostró que hacer cumplir alineaciones monótonas fuerte permite un texto a voz robusto, que se generaliza a largas pronunciaciones, y el empleo de flujos generativos permite síntesis de voz rápida, diversa y controlable. Glow-TTS puede generar espectrogramas mel 15,7 veces más rápido que el modelo TTS autorregresivo, Tacotron 2, mientras obtiene un rendimiento con calidad de voz comparable. Según la literatura, el modelo se puede extender fácilmente a una configuración de múltiples hablantes.

1.1.2. VoCoders

Los VoCoders neuronales basados en redes neuronales profundas pueden generar voces similares a las humanas, en lugar de utilizar las tradicionales métodos que contienen artefactos audibles[14][15][16].

La línea principal de la investigación se basa en los modelos TTS, como los antes expuestos, sin embargo como no es posible sintetizar voz sin un VoCoder, y luego de

varias pruebas realizadas, se concluye que los más adecuados para el objetivo principal son los siguientes:

HIFI-GAN

Varios trabajos recientes sobre la síntesis del habla han empleado redes generativas adversariales(GAN, por sus siglas en inglés) para producir formas de onda sin procesar. Aunque estos métodos mejoran la eficiencia de muestreo y uso de memoria, su calidad de muestra aún no ha alcanzado el de los modelos generativos autorregresivos y basados en flujo. HiFi-GAN[17] es un modelo que logra una síntesis de voz eficiente y de alta fidelidad.

Como el audio del habla consta de señales sinusoidales con varios períodos, se comprobó que modelar patrones periódicos de un audio es crucial para mejorar la calidad de la muestra.

Además se muestra la adaptabilidad de HiFi-GAN a la síntesis de voz de extremo a extremo. Para terminar, una versión pequeña de HiFi-GAN genera en CPU muestras 13,4 veces más rápido en tiempo real con calidad comparable a una contraparte autorregresiva.

UnivNet

La mayoría de los codificadores de voz neuronales emplean espectrogramas de mel de banda limitada para generar formas de onda. UnivNet[18], es un codificador neural de voz que sintetiza formas de onda de alta fidelidad en tiempo real.

Usando espectrogramas de mel de banda completa como entrada, se espera generar señales de alta resolución agregando un discriminador que emplea espectrogramas de múltiples resoluciones como entrada. En una evaluación de un conjunto de datos que contiene información sobre cientos de ponentes, UnivNet obtuvo los mejores resultados positivos, objetivos y subjetivos, entre los modelos que competían. Estos resultados, incluida la mejor puntuación subjetiva en la conversión texto a voz, demuestran el potencial para una rápida adaptación a nuevos hablantes sin necesidad de entrenamiento desde cero.

WaveGrad

WaveGrad[19] es un modelo condicional para la generación de formas de onda que estima los gradientes de la densidad de datos. El modelo se basa en trabajos previos sobre emparejamiento de puntuaciones y modelos probabilísticos de difusión. Parte de una señal Gaussiana de ruido blanco e iterativamente refina la señal a través de un muestreador basado en gradientes, condicionado en el espectrograma de mel. WaveGrad ofrece una forma natural de intercambiar velocidad de referencia por calidad

de la muestra ajustando el número de pasos de refinamiento, y cierra la brecha entre los modelos autorregresivos y no autorregresivos en términos de calidad de audio. El modelo puede generar muestras de audio de alta fidelidad usando como tan solo seis iteraciones. Los experimentos revelan que WaveGrad genera señales de audio de alta fidelidad, superando las líneas de base adversariales no autorregresivas y emparejando un fuertemente la línea de base autorregresiva basada en la probabilidad, utilizando menos operaciones secuenciales.

1.2. Modelos de extremo a extremo

1.2.1. YourTTS

1.2.2. VITS

Variational Inference with adversarial learning for end-to-end Text-to-Speech (VITS)[20] es un método TTS de extremo a extremo en paralelo. Usando un Autocodificador Variacional se conectan los dos módulos de sistemas TTS: modelo acústico y VoCoder, a través de variables latentes para permitir el aprendizaje de extremo a extremo.

Para mejorar el poder expresivo del método con el fin de sintetizar formas de onda de voz de alta calidad, se aplican flujos de normalización a la distribución condicional previa y entrenamiento adversarial en el dominio de formas de ondas.

El modelo VITS se describe principalmente en tres etapas: una formulación condicional de Autocodificador variacional; estimación de alineación derivada de la inferencia variacional; y entrenamiento adversarial para mejorar la calidad de la síntesis.

La arquitectura general del modelo consiste en un codificador posterior, un codificador anterior, un decodificador, un discriminante, y predictor de duración estocástica. El codificador posterior y discriminante solo se usan para entrenamiento, no para inferencia.

Para el codificador posterior se utilizan los bloques residuales no causales de WaveNet. Un bloque residual de WaveNet consta de capas convolucionales dilatadas con una puerta unidad de activación y salto de conexión. La proyección lineal capa por encima de los bloques produce la media y la varianza de la distribución posterior normal.

El codificador anterior consiste en un codificador de texto que procesa los fonemas de entrada, y un flujo de normalización que mejora la flexibilidad de la distribución anterior. El codificador de texto es un codificador transformador que utiliza representación posicional relativa en lugar de la codificación posicional absoluta. Mientras que el flujo de normalización es una pila de capas de acoplamiento afines [21] conformada por una pila de bloques residuales de WaveNet.

El decodificador es esencialmente el generador HiFi-GAN V1[17]. Se compone de

una pila de convoluciones transpuestas, cada una de las cuales va seguida de un módulo de fusión de campo multirreceptivo (MRF). El modelo continua la arquitectura del discriminante multiperíodo discriminador propuesto en HiFi-GAN. El discriminante multiperíodo es una mezcla de subdiscriminadores Markovianos basados en ventanas, cada uno de los cuales opera en diferentes patrones periódicos de formas de onda de entrada.

El predictor de duración estocástica estima la distribución de la duración del fonema a partir de una entrada condicional. Para la parametrización eficiente del predictor de duración estocástica, son apilados bloques residuales con bloques dilatados y capas convolucionales separables en profundidad. También se aplican flujos de ranunas neuronales[22] , que toman la forma de transformaciones no lineales invertibles mediante el uso de ranuras monótonas racionales-cuadráticas, aplicadas a capas de acoplamiento. flujos de ranunas neuronales mejoran la expresividad de la transformación con un número de parámetros en comparación con los de uso común en capas de acoplamiento.

Una vez concluido el proceso de entrenamiento, y según la literatura al compararse este modelo de extremo a extremo con sistemas de dos etapas, a través de los modelos preentrenados Tacotron2 y Glow-TTS como modelos de primer escenario e HiFi-GAN como modelo de segundo escenario, se comprueba que VITS obtiene un habla que suena más natural y cercana a la realidad. Una evaluación humana subjetiva (puntuación de opinión media, o MOS) en LJ Speech[23], un conjunto de datos de un solo hablante, muestra que el modelo supera a los mejores sistemas TTS disponibles públicamente y logra un MOS comparable a la realidad del terreno.

Ha mostrado la capacidad de ampliarse a la síntesis de voz de múltiples hablantes, generando un discurso con diversos tonos y ritmos de acuerdo a diferentes identidades de hablantes. Esto demuestra que aprende y expresa varias características del habla en un contexto de extremo a extremo.

1.3. Coqui-TTS

Existen conjuntos de sistemas TTS que abarcan la gran mayoría de los modelos explicados anteriormente, entre las más populares se encuentran **Mozilla-TTS**[24] y **Coqui-TTS**[25]. Ambas se comportan de forma similar, se instalan con el mismo comando `pip install TTS`, y el comando para ejecutarse tiene la misma sintaxis. Coqui-TTS fue fundado por miembros del equipo de Mozilla-TTS, y es su sucesor, pues Mozilla dejó de actualizar su proyecto STT y TTS.

Coqui TTS es una biblioteca para la generación avanzada de texto a voz. Se basa en las últimas investigaciones y se diseñó para lograr el mejor equilibrio entre la facilidad de entrenamiento, la velocidad y la calidad. Coqui viene con modelos preentrenados,

herramientas para medir la calidad del conjunto de datos y ya se utiliza en más de 20 idiomas para productos y proyectos de investigación.

Capítulo 2

Propuesta

Con el objetivo de lograr una síntesis de voz satisfactoria y después de realizar un estudio de las tecnologías que se utilizan para este fin, se elige Coqui TTS como herramienta base. Coqui cuenta con una gran variedad de modelos preentrenados en más de 20 idiomas. El primer paso en el desarrollo del sintetizador fue instalar la biblioteca Coqui TTS, de acuerdo a las indicaciones del repositorio oficial[25]. Se realizó un estudio del comportamiento de combinaciones de parejas modelo TTS y VoCoder, para evaluar cuáles ofrecían mejores resultados:

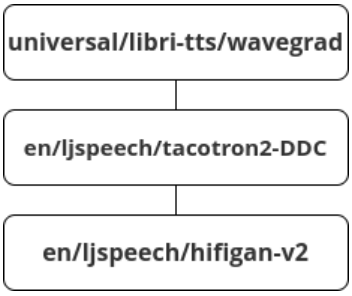

Figura 2.1: Modelo TTS Tacotron2-DDC entrenado en Inglés

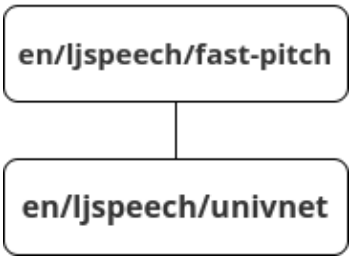

Figura 2.2: Modelo TTS Fast-Pitch entrenado en Inglés

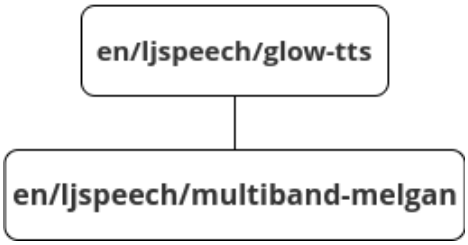

Figura 2.3: Modelo TTS Glow-TTS entrenado en Inglés

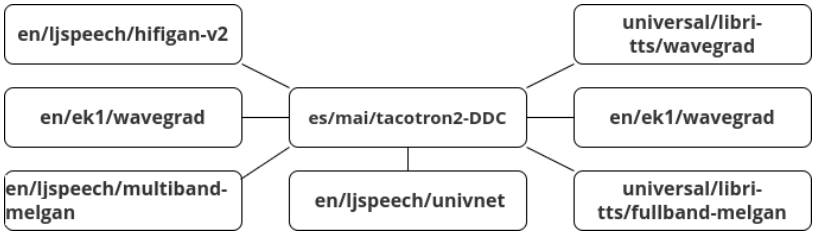

Figura 2.4: Modelo TTS Tacotron2-DDC entrenado en Español

Los modelos **Tacotron2-DDC, Fast-pitch y Glow-tts** preentrenados en idioma inglés al combinarse con VoCoders, como se observa en las figuras 2.1, 2.2, 2.3 para un texto en español arroja distintos resultados, en algunos casos solo producen ruido mientras que los más satisfactorios, producen un discurso con una pronunciación propia de una persona de habla inglesa hablando español.

Por otro lado Coqui solo cuenta con un modelo preentrenado en español, el **Tacontron-DDC**, que está entrenado sobre la base de datos de M-AILABS[26]; este modelo fue probado junto a los VoCoders preentrenados como se muestra en la figura 2.4. Los mejores resultados fueron arrojados por las combinaciones de **Tacontron-DDC** con los modelos: Univnet entrenado sobre la base de datos Ljspeech en inglés, y Wavegrad entrenado sobre el conjunto de datos LibriTTS, aunque ambos presentan problemáticas como la mala pronunciación de la letra ñ.

De acuerdo con la documentación de Coqui, y con el objetivo de la investigación que consiste en desarrollar un sintetizador de voz en español con voz cubana, se proponen tres enfoques:

1. Realizar un *fine-tuning* o reentrenamiento, utilizando una base de datos con voces cubanas, al modelo preentrenado de Coqui[25], **Tacontron-DDC**, que es el único disponible en español.

2. Realizar el entrenamiendo desde cero de un modelo, en este caso se podría

entrenar cualquier modelo disponible. **VITS** es un buen candidato pues en general produce resultados bastante satisfactorios.

3. Realizar un *fine-tuning* a modelos VITS preentrenados en idiomas distintos al español.

Y a partir de los anteriores evaluar y comparar resultados.

Para el desarrollo de cualquiera de los anteriores es imprescindible la construcción de un conjunto de datos que se adapten al modelo y a las necesidades de la investigación.

2.1. Creación de la base de datos

Para entrenar un modelo TTS, se necesita un conjunto de datos con grabaciones de voz y transcripciones, en este caso fue una base de datos en español con voces cubanas. El discurso debe dividirse en clips de audio y cada clip necesita su transcripción correspondiente. La base de datos debe poseer una organización específica, que se eligirá luego, de forma tal que el cargador de datos de Coqui TTS sea capaz de cargarlos para utilizar en el entrenamiento[27].

Formato wav

(***)

Hablar del idioman español

(***)

La base de datos conformanda debe tener una buena cobertura del idioma en el que se desea entrenar el modelo. Debe cubrir la variedad fonémica, los sonidos excepcionales y las sílabas.

Fine-Tuning

Entrenar correctamente un modelo de aprendizaje profundo requiere generalmente de una gran base de datos y de un extenso entrenamiento.

Si se dispone del material necesario y del tiempo para entrenar el algoritmo, estos requisitos no suponen ningún problema, pero, si la base de datos es pequeña o el modelo no se entrena lo suficiente, el aprendizaje podría no ser completo.

El *fine-tuning* es una forma de transferencia de aprendizaje, consiste en aprovechar la información que se ha aprendido de la resolución de un problema y utilizarla sobre

otro distinto, pero que comparte ciertas características. Se usan los conocimientos adquiridos por una red convolucional para transmitirlos a una nueva red convolucional. Esta nueva red convolucional que se crea no tiene por qué modificar la red original y puede simplemente aprender de ella, sin embargo también es válido el caso donde no solo se modifica la red original, sino que se vuelve a entrenar para aprender más conceptos.

En la presente investigación, se utiliza el reentrenamiento, para a partir modelo previamente entrenado y realizar un nuevo entrenamientor para mejorar su rendimiento en un conjunto de datos diferente.

2.2. Fine-Tuning de Tacontron-DDC

Se selecciona como primera variante el reentrenamiento(*fine-tuning* en inglés) del modelo **Tacontron-DDC** en español, pues plantea ventajas tales como un aprendizaje más rápido, ya que un modelo preentrenado ya tiene aprendidas funcionalidades que son relevantes para la tarea de producir un discurso. Además convergerá más rápido en el nuevo *dataset*, lo que reducirá el costo de entrenamiento y permitirá el proceso de experimentación más rápidamente. Y de acuerdo a la teoría se pueden obtener buenos resultados con un conjunto de datos más pequeño.

Luego de tener entrenado el modelo acústico(TTS) con una base de datos construida con voces cubanas, es posible que alguno de los VoCoders preentrenados disponibles produzca una salida con las características deseadas, en caso contrario se debería entrenar el VoCoder con los datos del *dataset* construido.

El proceso consiste en modificar la configuración original del modelo preentrenado seleccionado, es decir, especificar la base de datos a utilizar en el reentrenamiento, los detalles acústicos que reflejen las características del nuevo conjunto de datos, el nombre del nuevo modelo ajustado, la dirección donde se guardará el modelo reentrenado, entre otras cuestiones. Para la mayoría de los parámetros se tomaron las características del modelo original **Tacontron-DDC** en español.

2.3. Entrenamiento de modelo VITS desde cero

Existen varios modelos de texto a voz de extremo a extremo que permiten el entrenamiento en una sola etapa y el muestreo en paralelo, sin embargo, generalmente la calidad de la muestra no coincide con la de los sistemas TTS de dos etapas.

Se selecciona VITS porque es un método TTS paralelo de extremo a extremo que genera un sonido de audio más natural que los modelos actuales de dos etapas. Y de

acuerdo a una evaluación humana subjetiva (puntuación de opinión media, o MOS), muestra que el modelo supera a los mejores sistemas TTS disponibles públicamente y logra un MOS comparable a la realidad del terreno.

Con este enfoque se debe entrenar la red neuronal de VITS partiendo de cero. Una desventaja es que cae una gran responsabilidad sobre el conjunto de datos de entrenamiento, pues debe tener una gran riqueza y muchos clips de audio.

2.4. Fine-tuning de modelos VITS preentrenados en otros idiomas

2.4.1. Italiano

Las culturas hispánicas tienen mucho en común con la cultura italiana, sobre todo en lo que concierne al lenguaje y la comunicación. Tanto el español como el italiano son lenguas romances, derivadas del latín, siendo justamente de las más similares, incluso más que el francés o el rumano.

Es por este hecho que el italiano y el español comparten palabras muy parecidas y siguen la misma estructura gramatical. Entre ellos existe un grado de similitud léxica del 82 %, lo que además indica que son idiomas fáciles de aprender para sus respectivos hablantes. Sin embargo, las características que tienen en común van más allá de su origen y de la gramática.

Una similitud entre ambas lenguas es la cantidad de vocales en el alfabeto, aunque en italiano las vocales tienden a matizarse con sonidos más abiertos y con un acento grave. Es muy común que el parecido existente entre las palabras en italiano y español se vea afectada sólo por una o más vocales, como por ejemplo: vecino y vicino, cámara y camera, igual y uguale. Por supuesto existen un gran número de diferencias, aunque son más notables en vocabulario que en lo que sería la pronunciación.

Debido a todo esto un modelo entrenado en italiano sería el candidato ideal para realizar un ajuste sobre una base de datos en español.

2.4.2. Inglés

A pesar de las diferencias evidentes, en realidad el español y el inglés se parecen más de lo se cree. La primera de las semejanzas entre el inglés y el español es que usan el alfabeto romano, lo cual provoca que los sonidos de ambos idiomas sean similares, aunque no plenamente iguales. El mejor ejemplo lo constituyen las vocales: mientras en el español sólo se reconocen 5 sonidos, uno para cada vocal, en el inglés encontramos más de 14 sonidos, pues hay vocales cortas y largas, las cuales se catalogan así por la duración de su pronunciación. Estos dos idiomas son alfabéticos, de modo que a

través de letras se pueden representar sus sonidos; el español comparte 2/3 de sus fonemas con el inglés.

La estructura gramatical del inglés es similar a la del español en más de un 90 %, pero la del inglés es muchísimo más simple. Además, el vocabulario se forma con prefijos y sufijos tanto en inglés como en español (que son en su mayoría de raíz latina en ambos idiomas).

Las consonantesv, ll, h, j, r, rr, z, y la x son pronunciadas muy diferentes en español y en inglés. La consonante ñ no existe en inglés; el sonido que se conoce de esta consonante en español se escribe en ingles ny. Las combinaciones de algunas consonantes con vocales cambia totalmente el sentido de la pronunciación. Por ejemplo: la combinación de la "q" y la "u" en las palabras como queen, quiet o quick la pronunciación de la "u" en inglés se percibe.

Teniendo en cuenta la similitud, y al mismo tiempo las marcadas diferencias entre ambos idiomas, se escoge un modelo VITS preentrenado en inglés para realizar un *fine-tuning* con la base de datos cubana.

Capítulo 3

Detalles de Implementación y Experimentos

3.1. Instalación de la biblioteca Coqui

Coqui [25] es un repositorio de código abierto que implementa las últimas investigaciones en materia de síntesis de voz, como Tacotron 2 y VITS que son los modelos base utilizados en el presente proyecto. Este repositorio ha sido usado para generar modelos en más de 20 idiomas y cuenta además con múltiples "recetas" para el entrenamiento de modelos.

La biblioteca se instala de acuerdo a las instrucciones orientadas a desarrolladores en [25]. Con esto ya es suficiente para probar los modelos preentrenados disponibles de Coqui.

3.2. Creación de base de datos con voces cubanas.

¿Qué hace a un buen Dataset?

- Debe cubrir una cantidad considerable de clips cortos y largos.
- Libre de errores. Se debe eliminar cualquier archivo incorrecto o roto.
- Para escuchar una voz con la mayor naturalidad posible con todas las diferencias de frecuencia y tono, por ejemplo, usando diferentes signos de puntuación.
- Es necesario que el *dataset* cubra una buena parte de fonemas, difonemas y, en algunos idiomas, trifonemas. Si la cobertura de fonemas es baja, el modelo puede tener dificultades para pronunciar nuevas palabras difíciles.

- Las muestras de la base de datos deben estar lo más limpio posible, es decir, se debe limpiar de ruido y cortar los espacios de tiempo entre expresiones,donde no se hable.

Cuban Voice Dataset

La base de datos está conformada por 160 clips de audio con sus respectivas transcripciones recogidas en el archivo metadata.csv. Cada clip tiene una duración de 2 a 15 segundos, no más, para evitar sobrecargar los métodos que tienen que ver con la alineación.

Los clips de audio poseen formato .wav y se organizan dentro de una carpeta de nombre *wavs* de la siguiente forma:

/wavs
| - audio1.wav
| - audio1.wav
| - audio2.wav
| - audio3.wav
...

Las transcripciones se recogen dentro del archivo metadata.csv. Donde audio1, audio2, etc se refieren a los archivos audio1.wav, audio2.wav etc.

audio1|Esta es mi transcripción 1.
audio2|Esta es mi transcripción 2.
audio3|Esta es mi transcripción 3.
audio4|Esta es mi transcripción 4.

Se adoptará en la conformación de la base de datos con voces cubanas, la misma estructura de la base de datos en Español de *The M-AiLabs Speech Dataset*, pues algunos de los modelos que se utilizaron fueron preentrenados sobre estos conjuntos de datos. Finalmente queda la siguiente organización:

MyDataset/by_book/female/[creador del dataset]/[nombre del hablante]
|/wavs
|metadata.csv

3.2.1. Procesamiento de audio

RNNoise es una biblioteca basada en una red neuronal para la eliminación de ruido en grabaciones, se utliza en este proyecto para obtener clips de audio libres de ruidos y con la frecuencia de muestreo deseada.

Se realizó un procesamiento para la eliminación de ruido en el audio del *dataset* original, y se estableció una frecuencia de muestreo de acuerdo a las necesidades de cada experimento.

3.3. Herramientas

3.3.1. Colab

3.3.2. Google Drive

3.3.3. Phonemizer espeak

3.4. Modificación en el código fuente de Coqui TTS

El código fuente del repositorio ocasionaba problemas al conformar ruta del archivo metadata.csv, por lo que se realizó un pequeño cambio en el método `mailabs` del archivo `formatter.py`

```
def mailabs(root_path, meta_files=None, ignored_speakers=None):
"""Normalizes M-AI-Labs meta data files to TTS format

Args:
root_path (str): root folder of the MAILAB language folder.
meta_files (str):  list of meta files to be used in the training. If
    None, finds all the csv files
recursively. Defaults to None
"""
speaker_regex = re.compile("by_book/(male|female)/(?P<speaker_name
    >[^/]+)/")
if not meta_files:
csv_files = glob(root_path + "/**/metadata.csv", recursive=True)
else:
csv_files = meta_files

items = []
print(f"{csv_files}")
for csv_file in [csv_files]:
if os.path.isfile(csv_file):
txt_file = csv_file
else:
```

```
txt_file = os.path.join(root_path, csv_file)


folder = os.path.dirname(txt_file)
# determine speaker based on folder structure...
speaker_name_match = speaker_regex.search(txt_file)
if speaker_name_match is None:
continue
speaker_name = speaker_name_match.group("speaker_name")
# ignore speakers
if isinstance(ignored_speakers, list):
if speaker_name in ignored_speakers:
continue
print(" | > {}".format(csv_file))
with open(txt_file, "r", encoding="utf-8") as ttf:
for line in ttf:
cols = line.split("|")
if not meta_files:
wav_file = os.path.join(folder, "wavs", cols[0] + ".wav")
else:
wav_file = os.path.join(root_path, folder.replace("metadata.csv", ""), "
    wavs", cols[0] + ".wav")

if os.path.isfile(wav_file):
text = cols[1].strip()
items.append(
{"text": text, "audio_file": wav_file, "speaker_name": speaker_name, "
    root_path": root_path}
)
else:
# M-AI-Labs have some missing samples, so just print the warning
print("> File %s does not exist!" % (wav_file))
return items
```

Ejemplo de código 3.1: Códigos QR

3.5. Fine-Tuning de Tacontron-DDC

Como ya se expuso en otros capítulos, el *fine-tuning* resulta una idea prometedora, pues en teoría salva tiempo y recursos.

Para el proceso de ajuste de Tacotron2 a la base de datos personalizada, se utilizó la configuración del modelo preentrenado en español sobre el *dataset* de M-AILABS.

La frecuencia de muestreo(*sample rate*) que se establece en la configuración es 16000Hz, pues el conjunto de datos de M-AILABS sobre el que se preentrenó el modelo seleccionado, se encuentra en esta misma frecuencia. Finalmente se debe utilizar un

cargador de datos(*formatter*) compatible con la base de datos usada, en este caso se selecciona la variante `mailabs`, que se puede apreciar en la sección anterior.

El próximo paso es desacargar el modelo Tacotron2, para luego comenzar el reentrenamiento.

```
tts - -model_name tts_models/es/mai/tacotron2-DDC - -text "Hola."
>Downloading model to /home/ubuntu/.local/share/tts/tts_models-en-ljspeech
-glow-tts
```

El reentrenamiento se llevó a cabo utilizando el GPU Premium de Google Colab, y CUDA. Requirió una gran cantidad de memoria RAM, siendo 25GB una cantidad insuficiente, se comprobó que con un procesador de 83GB de RAM, sí podía realizarse. El reentrenamiento fue extremandamente costoso, en tiempo y en *computer units*[1], consumiendo más de 1000CU

(Resultados)

-Con 160 epochs se escucha una grabación con mucho ruido, sin embargo en algún momento se puede distinguir alguna sílaba.

-Con 210 epochs algo pasa que las salidad para una oración son audios de 2 minutos donde solo se distingue ruido, y a partir de ahí para 451 y 580 epochs, este comportamiento se mantienen invariante. Se tenía previsto alcanzar las 2000 epochs, pero como el modelo consumía muchos recursos y los resultados no eran los deseados, no se continuó.

-Los resultados no podían estar más lejos de lo esperado, primero se pensaba que iba a ser un reentrenamiento veloz e iba a converger rápidamente en el nuevo dataset

3.6. Entrenamiento de modelo VITS desde cero

Como los resultados de Tacotron2 no fueron los mejores, se optó por realizar experimentos con otros modelos, se eligió VITS por ser de los que mejores resultados arroja por encima de Tacotron2 y Glow-TTS[20].

Esta vez se entrena el modelo desde cero utilizando el conjunto de datos con voces cubanas, y la receta[28] que provee Coqui[25] para entrenar VITS. Para este entrenamiento, siguiendo ejemplos de entrenamientos anteriores, se cambia la frecuencia de muestreo del *dataset* a 22050. Por otro lado el *formatter* utilizado es la variante `mailabs`, que se encuentra en `formatters.py`

El entrenamiento se produjo utilizando el GPU Premium de Google Colab, y CUDA. Requirió una gran cantidad de memoria RAM, siendo 25GB una cantidad insuficiente, se comprobó que con un procesador de 83GB de RAM, sí podía reali-

[1]Una unidad de cómputo (CU) es la unidad de medida de los recursos consumidos por ejecuciones y compilaciones de actores.

zarse. Se produjo completa e ininterrumpidamente por 2000 *epochs*, resultando en que el último mejor modelo se genera en la *epoch* número 967. El entrenamiento no representó un gran costo, en tiempo y en *computer units*, demorando alrededor de 7 horas y consumiendo alrededor de 200CU.

Finalmente se obtiene un modelo que permite la emición de sonidos comprensibles, aunque no completamente inteligibles, pues produce un discurso robótico y tiene dificultad en la combinación de difonos, por lo que hay frases y palabras indescifrables para el oyente.

3.7. Fine-tuning de modelos VITS preentrenados

3.7.1. Modelo preentrenado en italiano

El modelo disponible en Coqui en idioma italiano fue preentrenado sobre la base de datos en italiano de M-AILABS, cuyas grabaciones poseen una frecuencia de muestreo de 16000Hz, por tanto el *dataset* de voces cubanas que se utiliza para el *fine-tuning* fue llevado a la misma frecuencia. El *formatter* utilizado es igualmente la variante `mailabs`.

El proceso de *fine-tuning* se llevó a cabo utilizando el GPU Premium de Google Colab, y CUDA. Requirió una gran cantidad de memoria RAM, siendo 25GB una cantidad insuficiente, no se precisa exactamente la cantidad de RAM necesaria, sin embargo, se comprobó que con un procesador de 83GB de RAM, sí podía realizarse sin problemas. Además de esto, no representó un gran costo, en tiempo y en *computer units*, demorando alrededor de 3 horas y consumiendo alrededor de CU.

El modelo que se genera luego de un reentrenamiento inninterrumpido durante 1000 *epochs*, arroja como resultado que, para una frase escrita dada, produce un discurso bastante comprensible, aunque un poco robótico, y entrecortado en alguna partes. Además con palabras que el oyente no puede descifrar. Es importante destacar que el modelo original de Coqui en italiano produce también una voz ruidosa, así que la cuestión del ruido es probable que venga desde el modelo inicial, agravada con el *fine-tuning* a partir del *Cuban Voice Dataset*.

3.7.2. Modelo preentrenado en Inglés

La variante del modelo VITS entrenada sobre el conjunto LJ-Speech Dataset en inglés, se seleccionó por ser el inglés un idioma, más distante del español que el italiano. Se lleva a cabo el mismo proceso que en los caso anterior, con la diferencia de que la base de datos *Cuban Voice Dataset* cambia su frecuencia de muestreo a 22050Hz. El reentrenamiento se realizó siguiendo las mismas características que en el modelo en italiano, y consumió alrededor del mismo tiempo y recursos.

Para terminar el modelo del inglés ajustado a la base de datos cubana arroja diferentes resultados que el modelo que se obtiene a partir del modelo italiano. Una ventaja es que el ruido, y la pronunciación robótica no están presentes en el nuevo discurso, y la voz sintética suena bastante parecida a la del hablante del conjunto de datos, está es un objetivo que hasta este punto no había sido alcanzado. Sin embargo, y como era de esperar gramaticalmente y fonéticamente presenta más problemas, entre ellos la pronunciación de la ñ y las r unidas a vocales, entre otros bastante evidentes al escuchar la salida.

3.8. Entrenamiento con M-AILABS DATASET

Conclusiones

Conclusiones

Recomendaciones

Recomendaciones

Bibliografía

[1] J. C. Tordera Yllescas, *Lingüıéstica computacional. Tecnologıéas del habla*, 2011 (vid. pág. 2).

[2] E. Rodrıéguez León, «Sintetizador de vocales sostenidas,» Tesis doct., Universidad Central"Marta Abreu"de Las Villas, 2013 (vid. pág. 2).

[3] A. Hernández Araujo y A. H. Araujo, «Siéntesis de voz esofágica,» 2018 (vid. pág. 2).

[4] Y. Wang y col., «Tacotron: Towards end-to-end speech synthesis,» *arXiv preprint arXiv:1703.10135*, 2017 (vid. pág. 7).

[5] J. Shen y col., «Natural tts synthesis by conditioning wavenet on mel spectrogram predictions,» en *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, IEEE, 2018, págs. 4779-4783 (vid. págs. 8, 9).

[6] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho e Y. Bengio, «Attention-based models for speech recognition,» *Advances in neural information processing systems*, vol. 28, 2015 (vid. pág. 8).

[7] D. Bahdanau, K. Cho e Y. Bengio, «Neural machine translation by jointly learning to align and translate,» *arXiv preprint arXiv:1409.0473*, 2014 (vid. pág. 8).

[8] Baidu. «Neural Voice Cloning with a Few Samples.» (), dirección: `http://research.baidu.com/Blog/index-view?id=81` (vid. pág. 9).

[9] S. Arik, J. Chen, K. Peng, W. Ping e Y. Zhou, «Neural voice cloning with a few samples,» *Advances in neural information processing systems*, vol. 31, 2018 (vid. págs. 9, 10).

[10] Baidu. «Neural Voice Cloning with a Few Samples.» (), dirección: `http://research.baidu.com/Blog/index-view?id=81` (vid. pág. 10).

[11] Y. Ren y col., «Fastspeech: Fast, robust and controllable text to speech,» *Advances in Neural Information Processing Systems*, vol. 32, 2019 (vid. pág. 11).

[12] A. Łańcucki, «Fastpitch: Parallel text-to-speech with pitch prediction,» en *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2021, págs. 6588-6592 (vid. pág. 12).

[13] J. Kim, S. Kim, J. Kong y S. Yoon, «Glow-tts: A generative flow for text-to-speech via monotonic alignment search,» *Advances in Neural Information Processing Systems*, vol. 33, págs. 8067-8077, 2020 (vid. pág. 12).

[14] D. Griffin y J. Lim, «Signal estimation from modified short-time Fourier transform,» *IEEE Transactions on acoustics, speech, and signal processing*, vol. 32, n.º 2, págs. 236-243, 1984 (vid. pág. 12).

[15] H. Kawahara, I. Masuda-Katsuse y A. De Cheveigne, «Restructuring speech representations using a pitch-adaptive time–frequency smoothing and an instantaneous-frequency-based F0 extraction: Possible role of a repetitive structure in sounds,» *Speech communication*, vol. 27, n.º 3-4, págs. 187-207, 1999 (vid. pág. 12).

[16] M. Morise, F. Yokomori y K. Ozawa, «WORLD: a vocoder-based high-quality speech synthesis system for real-time applications,» *IEICE TRANSACTIONS on Information and Systems*, vol. 99, n.º 7, págs. 1877-1884, 2016 (vid. pág. 12).

[17] J. Kong, J. Kim y J. Bae, «Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis,» *Advances in Neural Information Processing Systems*, vol. 33, págs. 17 022-17 033, 2020 (vid. págs. 13, 14).

[18] W. Jang, D. Lim, J. Yoon, B. Kim y J. Kim, «UnivNet: A neural vocoder with multi-resolution spectrogram discriminators for high-fidelity waveform generation,» *arXiv preprint arXiv:2106.07889*, 2021 (vid. pág. 13).

[19] N. Chen, Y. Zhang, H. Zen, R. J. Weiss, M. Norouzi y W. Chan, «WaveGrad: Estimating gradients for waveform generation,» *arXiv preprint arXiv:2009.00713*, 2020 (vid. pág. 13).

[20] J. Kim, J. Kong y J. Son, «Conditional variational autoencoder with adversarial learning for end-to-end text-to-speech,» en *International Conference on Machine Learning*, PMLR, 2021, págs. 5530-5540 (vid. págs. 14, 27).

[21] L. Dinh, J. Sohl-Dickstein y S. Bengio, «Density estimation using real nvp,» *arXiv preprint arXiv:1605.08803*, 2016 (vid. pág. 14).

[22] C. Durkan, A. Bekasov, I. Murray y G. Papamakarios, «Neural spline flows,» *Advances in neural information processing systems*, vol. 32, 2019 (vid. pág. 15).

[23] LJ-Speech. «LJ-Speech Dataset.» (), dirección: `https://keithito.com/LJ-Speech-Dataset/` (vid. pág. 15).

[24] Mozilla. «Mozilla TTS.» (), dirección: `https://github.com/mozilla/TTS` (vid. pág. 15).

[25] Coqui. «Coqui TTS.» (), dirección: https://github.com/coqui-ai/TTS (vid. págs. 15, 17, 18, 23, 27).

[26] I. Solak. «The M-AiLabs Speech Dataset.» (), dirección: https://www.caito.de/2019/01/03/the-m-ailabs-speech-dataset/ (vid. pág. 18).

[27] C. TTS. «Formatting Your Dataset.» (), dirección: https://tts.readthedocs.io/en/latest/formatting_your_dataset.html#formatting-your-dataset (vid. pág. 19).

[28] C. TTS. «Coqui Recipe to Train VITS.» (), dirección: https://github.com/coqui-ai/TTS/blob/dev/recipes/ljspeech/vits_tts/train_vits.py (vid. pág. 27).