Revista Digital del Departamento de Ingeniería e Investigaciones Tecnológicas de la Universidad Nacional de La Matanza

ISSN: 2525-1333
Vol.:4-Nro.1 (Julio-2019)

comunicación en congreso

Transferencia de Estilo entre Audios Mediante Redes Neuronales

Style Transfer between Audios Using Neural Networks

Hernán Ordiales[(1)], *Gabriel Martín Barrera*[(2)]

[(1)] Facultad de Ingeniería, Universidad de Palermo
hordia@palermo.edu

[(2)] Facultad de Ingeniería, Universidad de Palermo
gbarre4@palermo.edu

Revista Digital del Departamento de
Ingeniería e Investigaciones
Tecnológicas de la Universidad
Nacional de La Matanza

ISSN: 2525-1333
Vol.:4-Nro.1 (Julio-2019)

Resumen:

Este trabajo tiene como objetivo aplicar en archivos de audio las técnicas de procesamiento con redes neuronales desarrolladas para la transferencia de estilo en imágenes. En particular, aquellas que son de reciente publicación y dentro de su arquitectura utilizan una o más capas de redes neuronales convolucionales. Para ello, se construyen representaciones de la señal audible en matrices de estructura similar a las que normalmente se utilizan para procesar imagenes. Se evalúan diferentes aproximaciones al problema utilizando técnicas de análisis/síntesis como la transformada de tiempo corto de Fourier (STFT) y la descomposición de la señal de entrada en sinusoides y residuo, derivada del Spectral Modelling Synthesis, históricamente utilizado en señales de voz. Aunque la definición de estilo puede ser subjetiva, se ensayan algunas aproximaciones en su definición y reconocimiento. Para esto, se desarrollan e implementan diferentes programas en Python utilizando el framework TensorFlow, pensado para construir y entrenar redes neuronales. El resultado es un enfoque diferente para la aplicación de efectos digitales en señales de audio.

Abstract:

This paper aims to apply to audio files some neural networks processing techniques originally developed to do style transfer in the image domain. In particular, recently published work using one ore more layers of convolutional neural networks in its architecture. For this purpose, representations of the audible signal are constructed in matrices very similar like those used to process images. Different approaches to the problem are evaluated using analysis/synthesis techniques such as the Short Time Fourier Transform (STFT) and the decomposition of the input signal into sinusoids and noise or residual, a transformation derived from the Spectral Modeling Synthesis (SMS) historically usted with voice signals. Although the definition of style can be subjective, some approximations were made in its definition and recognition. For this, different programs were developed in Python using the TensorFlow framework, designed to build and train neural networks. The result is a different approach to the application of digital effects into audio signals.

Palabras Clave: *Transferencia de estilo, aprendizaje profundo, recuperación de información musical, procesamiento digital de señales, síntesis de modelado espectral*

Key Words: *style transfer, deep learning, music information retrieval, digital signal processing, spectral modelling synthesis*

Revista Digital del Departamento de Ingeniería e Investigaciones Tecnológicas de la Universidad Nacional de La Matanza

ISSN: 2525-1333

Vol.:4-Nro.1 (Julio-2019)

Revista Digital del Departamento de Ingeniería e Investigaciones Tecnológicas de la Universidad Nacional de La Matanza

ISSN: 2525-1333
Vol.:4-Nro.1 (Julio-2019)

I. CONTEXTO

En la última década y media, la publicación de trabajos que utilizan Deep Learning para procesar imágenes tuvieron un crecimiento acelerado, en parte, gracias a las posibilidades de procesamiento disponibles, mejoras en los procesadores y proliferación de GPUs (Graphic Processor Units). Por otra parte, la posibilidad de representar el audio matricialmente, permite en forma casi directa aplicar y explorar los mencionados algoritmos en este otro dominio. Este trabajo se desarrolló en el contexto del Trabajo Final de Grado de la carrera de Ingeniería en Informática de la Universidad de Palermo.

II. INTRODUCCIÓN

La técnica de transferencia de estilo con redes neuronales, conocida en el ámbito académico y técnico por sus palabras en inglés style transfer, es relativamente reciente y data de solo hace un par de años. Su aplicación inicial, propuesta por Gatys et al. en 2015 [1] utiliza imágenes bidimensionales de tres canales (Rojo, Verde y Azul o RGB). Basada en la minimización de energía entre sus representaciones, se demuestra o ejemplifica extrayendo estilos de obras de pintores famosos y aplicándolos a fotografias de paisajes reales.

El objetivo de este trabajo es trasladar estas técnicas a la transferencia de estilo en archivos de audio previamente grabados y almacenados en formato digital. Los cuales no necesariamente tienen que ser canciones, sino que puede tratarse de cualquier tipo de sonido con características identificables. Por ejemplo, ruido de tránsito, sonido de la lluvia, el cantar de unos pájaros, voz hablada (speech), estilos o formas de ejecutar un instrumento dentro de un mismo género musical.

Para ello, se construyen representaciones matriciales que reemplazan las habituales dimensiones utilizadas en imagenes (altura, ancho y cantidad de canales) por otras que representen la señal audible. Es decir, excediendo su representación bidimensional típica que consiste en ubicar la dimensión tiempo y su magnitud instantánea.Definiendo un proceso que comienza con el análisis de la señal de audio, es decir su descomposición en otros dominios, su procesamiento y re-síntesis o vuelta al dominio temporal bidimensional.

El primer enfoque consiste en entrenar la red neuronal convolucional con la información proveniente del espectograma [2], el cuál solo representa magnitudes espectrales a lo largo del tiempo, y por lo tanto se debe reconstruir su fase luego del procesamiento. Otro camino, explora las posibilidades de descomponer la señal original en sus componentes armónicas sinusoidales que evolucionan a lo largo del tiempo y su residuo o ruido de fase. Permitiendo el procesamiento (o no) de cada parte y reconstrucción independiente.

En la comparación, que por tratarse de estilos incluye varios factores subjetivos, surgen ventajas y desventajas de cada técnica, se observan pequeñas diferencias manteniendo en general una calidad aceptable, que no alcanza para requerimientos de uso profesional, pero se considera una buena primer aproximación en el uso de esta técnica.

III. MÉTODOS

Revista Digital del Departamento de Ingeniería e Investigaciones Tecnológicas de la Universidad Nacional de La Matanza

ISSN: 2525-1333
Vol.:4-Nro.1 (Julio-2019)

En cuanto a los que aplican este tipo de técnicas de style transfer, se encuentran las de Gatys et al. [1][3] que como se mencionó, fue el primero en utilizar redes convolucionales (Convolutional Neural Networks o CNN) para sintetizar texturas o estilos. Explora las posibilidades de este enfoque y proporciona algunos ejemplos útiles. Utiliza una arquitectura de redes convolucionales VGG-19 , compuesta por capas de filtros convolucionales y de Max Pooling, cuyas características se explicaran más adelante. La publicación se jacta de poder manipular por separado las representaciones de estilo y contenido para producir nuevas imágenes. A grandes rasgos, sostiene que las activaciones de capas superiores (o profundas) son las que contienen una representación del estilo.

Entonces, ¿Por qué utilizar redes neuronales? Los seres humanos distinguimos casi naturalmente, con poco o nulo entrenamiento, entre estilos musicales. Por ejemplo entre una canción de rock y una de música clásica, un rap o un reggae. Solo por citar algunos ejemplos. Lo mismo ocurre entre compositores dentro de un mismo género. Sabemos que es así, lo podemos identificar y entender, pero no necesariamente comprendemos el por que lo sabemos . Por lo tanto, si sabemos que el cerebro puede aprender esas cosas ¿Por qué no usar técnicas que se basan en su funcionamiento como las
redes neuronales, para resolver el problema?

A. Redes neuronales convolucionales

El interés en utilizar una técnica de aprendizaje profundo denominado Convolutional Neural Networks (CNN), reside en aprovechar sus características para modelizar una experiencia. Habilidad, como se mencionó, compartida con el resto de las redes neuronales y el Machine Learning en general, pero en este caso, en lugar de utilizarla para predecir o reconocer, o además de ello, se aprovechará la posibilidad de manipularlas para construir algo nuevo con características parecidas a los elementos que se utilizaron durante el entrenamiento. La transferencia de estilo, a priori, tiene mucho que ver con eso.

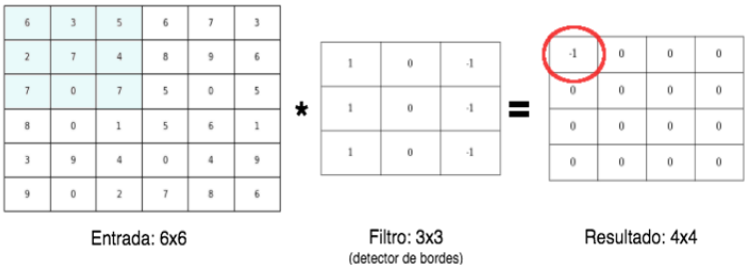

Fig.1. Ejemplo de convolución matricial.

Generalmente forman parte de una arquitectura en las que se combinan como unidades de procesamiento con capas Fully Connected (FC) todas las neuronas interconectadas y Max o Average Pooling como en la Figura 2 (valor máximo o promedio tomando zonas o ventanas). Como función de activación se suele utlizar ReLU (unidad lineal rectificadora). Hay diversas convenciones, pero la convolución se puede entender como una operación matricial como la de Figura 1. Donde la entrada, una matriz que puede representar uno de los canales de una imagen bidimensional, se convoluciona con un filtro detector de bordes verticales.En este caso, a modo de ejemplo se muestra como se calcula la primer posición del resultado:

Revista Digital del Departamento de Ingeniería e Investigaciones Tecnológicas de la Universidad Nacional de La Matanza

ISSN: 2525-1333
Vol.:4-Nro.1 (Julio-2019)

$$6x1 + 2x1 + 7x1 + 3x0 + 7x0 + 0x0 + 5x(-1) + 4x(-1) + 7x(-1) = -1$$

El resto de los valores se calcula de igual forma, desplazando la ventana de en horizontal y vertical de a una unidad. Salvo en los casos que se utiliza stride o salto, que por ejemplo con un valor igual a dos, genera reducción de la dimensionalidad [4].

La fórmula para calcular la dimensión del resultado es:

$$n_{salida} = \frac{(n_i + 2p - f)}{s} + 1 \qquad (1)$$

Con dimensión de la entrada, el padding (si se agregran ceros), el tamaño del filtro, s el stride mencionado y 1 que corresponde al bias. Para el caso de la Figura $1 n_{salida} = (6 + 2x0 - 3)/1 + 1 = 4$. Que por ser la entrada una imagen cuadrada de 6x6, la salida es una matriz de 4x4.

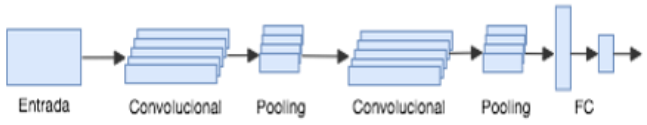

Fig.2. Arquitectura típica de una RN convolucional.

Una ReLU computa una función lineal de las entradas, y tiene como salida el mismo resultado si es positiva, o de lo contrario es igual a 0 [4]. ReLU viene a reemplazar lo que se usaba históricamente para manejar alinealidad con la función sigmoidea y tanh (arcotangente) como función de activación. Por ejemplo, esta puede tener la siguiente forma: $h_{w,b}(X) = max(Xw + b, 0)$, con X matriz, w peso y b bias. La capa de Max Pooling, equivale a realizar un downsampling. Se toma un cuadrante de tamaño predefinido de la matriz original y se toma el mayor valor en el caso de Max Pooling y el promedio en Average Pooling. Se puede pensar como que generaliza la información de acuerdo a su ubicación.

La función softmax, es opcional y se usa para reducir dimensionalidad, como capa final de un clasificador.

Durante el entrenamiento, la CNN encuentra los filtros o kernels más útiles para la tarea, y aprende como combinarlos en patrones más complejos [4] .

Las mencionadas aplicaciones de OCR y reconocimiento de dígitos, dieron un salto cualitativo con el uso de una arquitectura de este tipo en 1998, la denominada LeNet-5 [5] que demostró ser muy eficiente para reconocer dígitos manuscritos. Cuenta con 5 capas, 3 convolucionales y 3 de Average Pooling (downsampling promedio) intercaladas. Son convenientes para procesar imágenes, ya que su dimensionalidad matricial es muy alta, por ejemplo una imagen muy pequeña de 64x64, con canales RGB, tiene 64x64x3 = 12288 valores. Mientras que para una imagen de alta resolución, de 1000x1000x3 , se está en el orden de los 3 millones.

Esto significa que con las técnicas habituales, a la entrada a la red neuronal se tiene una entrada de 3M, y si en la primer capa oculta tenemos 1000 neuronas, nos queda una matriz de dimensión [1000,3M] para procesar, es decir 3000 millones de parámetros, que además de ser costoso de manejar, trae problemas con el overfitting o modelo sobremuestreado. En cambio con el enfoque que utiliza capas convolucionales, con un esquema similar al de la Figura 2, en una imagen de 32x32x3 = 3072, con filtros de 5x5 = 25, más 1 del bias, da 26 features por cada filtro.

Revista Digital del Departamento de Ingeniería e Investigaciones Tecnológicas de la Universidad Nacional de La Matanza

ISSN: 2525-1333
Vol.:4-Nro.1 (Julio-2019)

Siendo 6 filtros, quedan parámetros para entrenar. Un número mucho más chico que el calculado anteriormente.

La misma situación, con capas Fully Connected, con 6 filtros de 5x5, queda en 28x28x6 = 4704. Interconectar 3072 por 4704 da aproximadamente 14 millones de parámetros. Considerando que la imagen es muy pequeña, son un montón de parámetros para entrenar, versus los 156 de las CNN. Estas son alguna de las razones porque las CNN funcionan tan bien en visión por computadora. A partir de la práctica, se empezaron a reconocer patrones sobre que arquitecturas son más efectivas. Lo que muchos suelen hacer es tomar la arquitectura que algún otro publico en algún paper de investigación y aplicarla . Este mismo camino es el que toma el presente trabajo, y se espera poder aplicar estas bondades buscando diferentes objetivos en las señales de audio.

B. Arquitecturas de CNN

La arquitectura es la estructura principal del algoritmo de procesamiento. Se diseña utilizando bloques básicos de construcción, las mencionadas capas convolucionales de Max Pooling, ReLU, Fully Connected, etc. Para ganar intuición en la materia, se analizan otros casos de arquitecturas efectivas, así como para aprender a programar y escribir código fuente en un lenguaje en particular se observa código escrito por otras personas para otras aplicaciones. Ejemplos de otras arquitecturas convolucionales efectivas son: LenNet-5, AlexNet, GoogLeNet, ResNet y VGG-19 [4] . Por ejemplo, VGG es lo que usan en los papers de Gatys [1][3] .

Uno de las tendencias actuales es utilizar cada vez menos features o descriptores de los datos como entrada, una suerte de preproceso de los mismos, y alimentar las redes neuronales directamente con raw data (datos en crudo). La idea detrás de esto, es no condicionar el aprendizaje. En el caso de una imagen, por ejemplo, se ingresa directamente con los valores RGB de la misma, en lugar de valores calculados a partir de la imagen original o features. Y por lo tanto, se deja de lado lo que se conoce como Ingeniería de Features.

En este caso, una arquitectura típica para procesamiento de audio es la que se propone en la Figura 3.

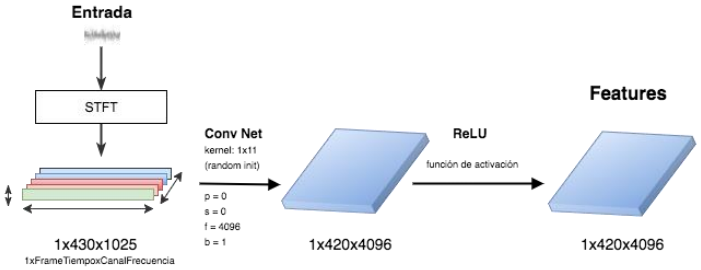

Fig.3. Arquitectura CNN para audio

La entrada es una señal de audio que contiene el estilo que se quiere aplicar a otro audio, poseedor del contenido. Luego de transformar la señal en una matriz por medio de la STFT. Una capa convolucional de 2 dimensiones, a la que se aplica un kernel o filtro, que a diferencia de lo que se ejemplifico y se utiliza en imágenes, no tiene dimensión de 3x3, sino de 1x11, ya que la dimensión correspondiente a la altura permanece en 1. El tamaño del filtro es un

Revista Digital del Departamento de Ingeniería e Investigaciones Tecnológicas de la Universidad Nacional de La Matanza

ISSN: 2525-1333
Vol.:4-Nro.1 (Julio-2019)

parámetro de la red y puede ser otro. En este caso se utiliza 11. Luego se aplican 4096 de estos filtros o kernels, todos inicializados con valores aleatorios, algo también inspirado en el paper de Gatys y la literatura en general sobre visión por computadoras, donde este tipo de técnica funciona bien como extractor de features o características. Sin padding (p=0) y con stride igual a 1. La es del bias. Utilizar relleno (padding) implicaría mantener la dimensión de la matriz, que puede ser útil por cuestiones de borde, pero en este caso a priori no es necesario. La dimensionalidad se calcula de igual forma a la explicada anteriormente, la matriz inicial de [1x430x1025] corresponde a 430 muestras temporales y 1025 canales de frecuencia generados por el cálculo del espectrograma. Que luego de la capa convolucional queda en tensor de [1x420x4092] según la ecuación 1. Luego se aplica la ReLU como función de activación max(features, 0) y que determina la alinealidad, para terminar de generar las features del estilo. El mismo proceso se aplica entrando con la señal de audio del contenido para obtener sus features de la misma manera.

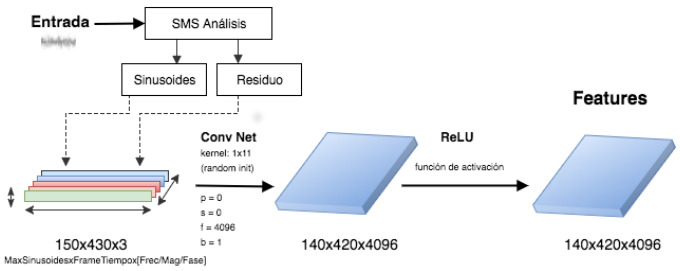

Fig.4. Arquitectura CNN con Análisis Espectral

La variación propuesta en este trabajo, se puede ver en la Figura 4 y consiste en generar la matriz de audio reemplazando el espectrograma que se genera con la STFT por la transformación SMS [6].
Quedando dimensionalmente de la forma [MaxSinusoides, FramesTiempo, 3], donde MaxSinusoides es un parámetro de la transformación y determina cuál es el máximo de sinusoides detectables. FramesTiempo es similar al caso anterior con STFT, indica la ubicación temporal. Mientras que la dimensión 3 correspondiente a los canales almacena los valores de Frecuencia, Magnitud y Fase de cada sinusoide determinada por la primer coordenada.

C. Implementación

El algoritmo esta basado en la minimización de energía. Una vez obtenidos los features del contenido y el estilo, al igual que en otros algoritmos de Deep Learning, se procede a definir una función que represente el objetivo que se busca y minimizarla. Esto se denomina loss, perdida o costo. En este caso es una manera de representar matemáticamente la transferencia de estilo, y consiste en minimizar la distancia matemática entre el estilo del sonido aplicado y el estilo del resultado generado (costo_estilo). Y por otro lado la distancia entre el contenido del sonido original y del generado (costo_contenido). Por lo tanto, se procede a minimizar una función del tipo:

costo = costo_contenido + costo_estilo (2)

Revista Digital del Departamento de Ingeniería e Investigaciones Tecnológicas de la Universidad Nacional de La Matanza

ISSN: 2525-1333
Vol.:4-Nro.1 (Julio-2019)

Luego se construye el grafo de la red neuronal, se genera un x de entrada con valores aleatorios, y se optimiza el mencionado costo o función de loss (perdida).

Se setea una cantidad de iteraciones, por ejemplo 300, para optimizar este valor. Puede ser el algoritmo del gradiente descendiente Adam, o cualquier función para optimizar todos los parámetros de la red neuronal y minimizar la función de costo [3]. El armado del grafo de procesamiento de la red neuronal se implementó en TensorFlow (TF), con cuatro o dos capas, según la convención utilizada. Se repite un bloque de una unidad convolucional seguida por una de Max Pooling y la ReLU al final, como se puede ver en el código fuente [1]. Dependiendo del autor, MaxPooling puede no considerarse una capa, ya que no tiene parámetros para entrenar, aunque si los posee para su configuración.

Parámetros de configuración de la red son FILTER_DIM (dimensión o cantidad de valores de los filtros o kernel), N_MUESTRAS (cantidad de muestras de audio a procesar), N_CANALES (determina la dimensión de la matriz de entrada), N_FILTROS (la cantidad de filtros que se utilizan), PADDING (si se rellena con ceroso o no, en el caso ejemplificado no se usa) y STRIDE (el salto, configurado en 1). La variable PROC_DEVICE es para elegir si el procesamiento se hace con la CPU o GPU (en caso de estar disponible). En el último caso, hay que tener consideraciones sobre la RAM como se detallará en las conclusiones. La función tf.nn.conv2d incluida en el paquete TensorFlow realiza una convolución en dos dimensiones. Cuya entrada x queda definida en cuatro dimensiones [batch, ALTO, ANCHO, N_CANALES] y un grupo de filtros W1. Esta función convoluciona los filtros W1 en X. El tercer argumento representa los strides para cada dimensión de la entrada, que para el caso del audio se representa por: posición en el batch o lote, 1 de altura, posición temporal o ancho y frecuencia o canal.

Dado que el batch es 1 y para el canal se mantiene en 1, se define por [1,s,s,1], y si se usa stride s=1 queda seteado como [1,1,1,1]. También se configura como VALID el padding, que como se dijo, implica que no hay rellenado con ceros o padding y que puede cambiar el tamaño de la salida (output).

IV. RESULTADOS Y OBJETIVOS

Al observar el resultado como imágenes (ver Figura 5) se nota como las características de contenido y estilo se ven reflejadas en el espectrograma final. El audio se encuentra online[2] para poder ser escuchado y comparar resultados. El mismo corresponde al ejemplo mencionado en la Figura 6.

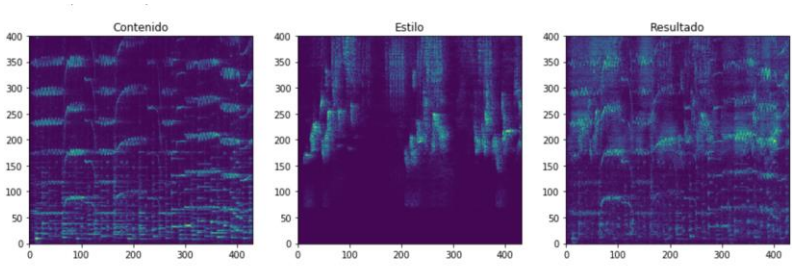

Fig.5. Contenido, estilo aplicado y resultado

[1] Código fuente: https://github.com/hordiales/transferencia-estilo-sms/blob/master/Demo/Demo.ipynb

[2] Disponible para escuchar en https://soundcloud.com/hordia/choclo-estilo-transferido-piazzolla/s-hYFVB

Revista Digital del Departamento de Ingeniería e Investigaciones Tecnológicas de la Universidad Nacional de La Matanza

ISSN: 2525-1333
Vol.:4-Nro.1 (Julio-2019)

Al despreciar la fase original se pierden las características finas a nivel temporal contenidas en ella, lo que se traduce en una perdida de calidad final. En este trabajo se procesan archivo de un solo canal (mono), pero esto puede complicar la espacialización en archivos estéreo o multicanal. Así como afectar el vibrato y efectos modelables como LFO (Low Frequency Oscillation).

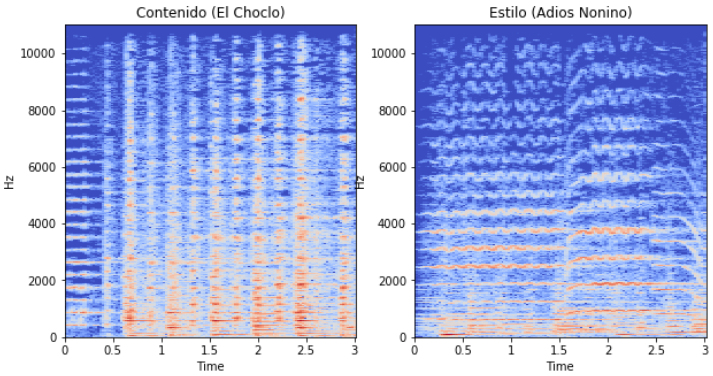

Fig.6. Comparación de 2 espectrogramas con estilos diferentes (3 segundos)

En las pruebas auditivas la transferencia de estilo se evidencia con una calidad aceptable en el modelo original a partir del espectrograma. En la implementación con SMS, el resultado fue dispar. Con la representación propuesta se obtuvo resultados cercanos al ruido blanco, a pesar de que se intentó con diferentes parámetros y configuraciones de red. Se probó de reducir la cantidad de filtros y su tamaño, por ejemplo inicializados en aleatorio de 3x3 y de 11x11. Se intentó agregando una capa de MaxPooling (downsampling), armando la estructura típica de capa convolucional seguida por MaxPooling que suele estar presente en las arquitecturas más utilizadas para imágenes, pero esto solo se tradujo una en una perdida de definición en el resultado final. Por lo tanto, se terminó descartando y se volvió a la arquitectura que solo procesa con la capa convolucional. La única que aprende parámetros.

MaxPooling o AvgPooling, no aprenden durante el entrenamiento, solo se configuran el tamaño y el stride.

Inicializando los filtros con valores aleatorios, escalados por un desvío estándar, se obtuvieron mejores resultados que definiendo filtros detectores de manera fija. Otras variantes fueron diferentes valores para la constante que determina la función loss que se optimiza. En este caso, la mayoría de las opciones arrojaron resultados similares, siendo la diferencia en el resultado solo una cuestión estética. Con un esquema como el de la Figura 3, pero alimentándolo solo con la parte sinusoidal de la señal obtenida con el procesamiento SMS, se obtuvieron mejores resultados. En particular cuando se utilizó una baja cantidad de sinusoides en la configuración del modelo, tomando entre 16 y 32 posibles, en lugar de las 150 iniciales. Con esto se logró concentrar la transferencia de estilo en las componentes sinusoidales y se descarto el ruido, que solo agregaba defectos. Mediante este camino se obtuvieron resultados más definidos. Incluso se probó procesar cada canal con diferente kernel. Aplicando valores aleatorios, como en el trabajo original, en algunos y en otros filtros detectores, pero no se logró identificar completamente su impacto. El valor de final_loss obtenido sirvió de referencia para el resultado, esperando que si se había logrado analíticamente una mejor minimización, es decir más cercana a cero, se esperaría un mejor resultado final. Aunque como se dijo, en este aspecto también intervienen cuestiones subjetivas.

Revista Digital del Departamento de Ingeniería e Investigaciones Tecnológicas de la Universidad Nacional de La Matanza

ISSN: 2525-1333
Vol.:4-Nro.1 (Julio-2019)

V. DISCUSIÓN

También se probó con la transformación de Mel-Spectrum y un algoritmo de reconstrucción, el cual funcionó, pero la calidad final fue muy baja. Es decir, el audio queda reconocible, pero muy deteriorado. Las pruebas en general se dificultaron debido a que el procesamiento se extendía por un largo periodo, consumiendo todos los recursos de procesamiento y memoria disponibles en la computadora de trabajo. Se modificó la cantidad de iteraciones del algoritmo, y se realizaron diferentes configuraciones de memoria virtual en los discos de estado sólido, de mayor velocidad de lectura/escritura. La aplicación de estilo en los casos de prueba se vio mejorada en su velocidad de procesamiento notablemente cuando se actualizó la versión de TensorFlow de la 1.4 a la 1.9 actualmente en desarrollo y con releases nocturnas, es decir la última versión compilada por el proyecto en el momento de la redacción de este trabajo. Otras opciones que se investigaron fueron las siguientes:

- Compilar TensorFlow con soporte para los flags soportados en los CPU disponibles (AVX2, FMA, SSE, etc)
- Investigar la posibilidad e incorporar una GPU externa o utilizar un dispositivo dedicado (embebido)
- Trabajar con muestras de audio aún más cortas, en las pruebas para reducir las matrices y el costo computacional. Lo mismo para el kernel de los filtros.
- Modificación del algoritmo, utilizando placeholders TF.
- Configurar restricciones de memoria y cpu con el ulimit para no bloquear la máquina.

En general, tomando filtros kernel de más tamaño, estos impactaron más directamente en el resultado, aplicando más estilo. Aunque por otra parte también se notó que pueden generar artifacts, o un efecto similar a cuando se comprime audio con bajo bitrate, que en la jerga se conoce como warbling. Algo que se percibe como una distorsión modulada y por lo tanto como baja calidad final. Los ejemplos de este trabajo se corrieron en un Intel i7-7700 CPU 3.60GHz, 16 GB de RAM y 8 núcleos, pero sin GPU. Lo cual arroja para la versión 1.4 de TF que para procesar 3 segundos, se demora aproximadamente 45 minutos. Las pruebas con el kit NVIDIA Jetson TX1 embebido, que cuenta con un procesar ARM y una GPU de 256 CUDA cores tardan un tiempo similar. Aquí el problema resultó los escasos 4GB de RAM necesarios para el entrenamiento, por lo cual se debió recurrir a memoria virtual en un disco de estado sólido externo.

VI. CONCLUSIONES

Debido al costo computacional, y la necesidad de ajustar parámetros de la red para cada ejemplo, se trata de un método que esta lejos de ser un método automático y que pueda correr en tiempo real. Para poner en contexto y tener un punto de comparación, el reciente trabajo de Facebook [7]. Reporta que utilizó durante 6 días 100 GPU's Tesla V100 (5120 CUDAS cores) y cada una de ellas promete el funcionamiento equivalente a 100

ISSN: 2525-1333
Vol.:4-Nro.1 (Julio-2019)

CPU's[3]. La necesidad de RAM disponible para la etapa de entrenamiento de la red neuronal es una de las características del uso de arquitecturas CNN[4]. Y la falta de la misma, puede limitar la extensión de los audios sobre los que se aplican estas técnicas. También es interesante indagar en el aprendizaje artificial generado, para comprender como los seres humanos generamos este tipo de conocimiento. Lo cual puede tener aplicaciones en otras áreas. La principal contra radica en que cualquier experimento o uso del Deep Learning, requiere de mucha infraestructura disponible, recursos de hardware o tiempo disponible para esperar a que concluyan los procesos, tanto de aprendizaje como de ejecución final. Al entrenar con más de un archivo de audio por estilo. El tiempo de procesamiento se incrementa exponencialmente, pero la diferencia de resultado es mucho más pequeña. O sea, para lograr una pequeña mejora, hay que hacer un gran esfuerzo.

En cuanto a hardware, es conveniente contar con placas de video GPU CUDA compatible, para acelerar el proceso. Incluso armar un rack con varias de ellas. Al tratarse de aprendizaje automático y contar con muchas capas intermedias, es muy difícil conocer cómo está realmente aprendiendo el sistema, y es muy probable que se termine utilizando como una caja negra, de la que no se puede conocer del todo su comportamiento, es decir, como va a reaccionar, qué salida va a producir ante determinada entrada. El estado por el momento es muy experimental y de nicho, pero abre la puerta a futuras investigaciones basadas en esta experiencia.

Queda pendiente utilizar varios archivos de audio para modelar el estilo, como se hace con el style transfer de imágenes. En el presente trabajo, aunque se hicieron algunas pruebas en otro sentido, en general solo se toma una muestra de audio para extraer features del estilo, pero se podrían tomar varios extractos de diferentes canciones de un mismo autor y evaluar resultados. Probar variaciones en el tempo o BPM del estilo, evaluando como afecta transformar primero esta propiedad en el mismo, ajustándola al contenido. En cuanto a arquitecturas de Deep Learning, quedan por explorar las posibilidades de utilizar GAN (Generative Adversarial Networks) donde se utilizan dos redes, una que trata de discriminar evaluando si lo que genera la otra tratando de engañar al modelo es original y en base a eso a podría aprender sobre su estilo tal como lo hace un crítico de arte [8][9]. Y por otro lado, resta evaluar las técnicas neuronales de autoencoders[10], de los citados trabajos del neural synth de Google Magenta y el más reciente de Facebook [7]. Proponer métricas para poder caracterizar el estilo con valores numéricos y así poder evaluar los resultados con cierta objetividad. Una posibilidad es utilizar descriptores o features provenientes del Music Information Retrieval (MIR) como centroide espectral (directamente asociado al timbre), el pitch salience (asociado a sonidos percusivos), evaluar el contenido de alta o baja frecuencia o la complejidad armónica entre otros posibles valores calculables [2]. También queda pendiente una mejor validación cruzada entre varias implementaciones para encontrar los mejores parámetros para la red neuronal.

[3] https://research.fb.com/facebook-researchers-use-ai-to-turn-whistles-into-orchestral-music-and-power-other-musical-translations/

ISSN: 2525-1333
Vol.:4-Nro.1 (Julio-2019)

Entre las aplicaciones posibles, una interesante, podría ser la de una aplicación móvil de entrenimiento, similar a la comentada de los filtros de Prisma, pero aplicando transferencias de estilo divertidas al audio. Cabe destacar, que la próxima generación de celulares contará con chips dedicados para Machine Learning y redes neuronales, denominados TPU (Tensor Process Unit), similares a las unidades de proceso gráficas como GPU, también disponibles en Cloud [4]. Otras aplicaciones incluyen utilizarlo como efectos para instrumentos, por ejemplo una guitarra, reemplazando otras técnicas para modelar amplificadores o procesamientos específicos. Incluso, a futuro, de lograr una implementación de tiempo real, se podría emular el sonido de tal o cual guitarrista reconocido. De esto se desprende la necesidad de optimizar el algoritmo para reducir su tiempo de procesamiento. Otro objetivo podría ser evaluar como se comporta el style transfer con voces, investigar como atacar los formantes, si procesarlos por separado con SMS o no. Este es un tema que puede traer aparejado cuestiones éticas a considerar. Porque de lograrlo con alta calidad, fácilmente se podrían generar audios falsos de personas influyentes, que podría causar confusiones en la población, o más grave aún, problemas políticos.

VII. REFERENCIAS Y BIBLIOGRAFÍA

[1] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge, “A Neural Algorithm of Artistic Style,” p. 16, aug 2015.

[2] Alexander Lerch, An introduction to audio content analysis: Applications in signal processing and music informatics, IEEE PRESS, 2012.

[3] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge, “Texture Synthesis Using Convolutional Neural Networks,” may 2015.

[4] Aurélien Géron, “Hands-on machine learning with scikit-learn and tensorflow: concepts, tools, and techniques to build intelligent systems,” 2017.

[5] YannLeCun,LeónBottou,YoshuaBengio,and Patrick Haffner, “Gradient-based learning applied to document recognition” Proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2324, 1998.

[6] Xavier Serra, Xavier Serra, and Xavier Serra Ph. D, “A System for Sound Analysis/Transformation/Synthesis Based on a Deterministic Plus Stochastic Decomposition,” 1989.

[7] Noam Mor, Lior Wolf, Adam Polyak, and Yaniv Taig- man, “A universal music translation network,” arXiv preprint arXiv:1805.07848, 2018.

[8] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio, “Generative Adversarial Nets,” 2014.

[9] Chris Donahue, Julian Mcauley, and Miller Puckette, “Synthesizing Audio with Generative Adversarial Networks,” 2018.

[10] Joseph Colonel, Christopher Curro, and Sam Keene, “Improving Neural Net Autoencoders for Music Synthesis,” 2017.

[4] Cloud TPU: https://cloud.google.com/tpu/

Revista Digital del Departamento de Ingeniería e Investigaciones Tecnológicas de la Universidad Nacional de La Matanza

ISSN: 2525-1333
Vol.:4-Nro.1 (Julio-2019)

Recibido: 2019-04-30
Aprobado: 2019-06-20
Datos de edición: Vol. 4 - Nro. 1 - Art. 6
Fecha de edición: 2019-06-28
URL: http://reddi.unlam.edu.ar