Clasificación de especies de aves a partir de entrenamiento de redes neuronales con cantos sintéticos

Marcos Wappner

Tesis de licenciatura Ciencias Físicas

Departamento de Física

Univarisdad de Buenos Aires

Argentina • Marzo 2020

TEMA: Física de sistemas biológicos

ALUMNO: Marcos Wappner

LU N° : 400/14

LUGAR DE TRABAJO: Laboratorio de Sistemas Dinámicos

DIRECTOR DEL TRABAJO: Gabriel B. Mindlin

FECHA DE INICIACION:

FECHA DE FINALIZACION:

FECHA DE EXAMEN:

INFORME FINAL APROBADO POR:

Autor	Jurado
Director	Jurado
Profesor de Tesis de Licenciatura	Jurado

Agradecimientos

A mis viejos, a Dani y a Meli, agradezco un montón. También a Gabo que me bancó incansablemente, aún cuando yo andaba en otra. A la gente de la facu, sin quienes el trabajo se habría hecho durísimo, y a la gente de fuera de la facu, sin quienes muchas cosas habrían sido también durísimas. Agradezco también al laboratorio y la institución por brindarme esta educación de excelencia.

Resumen

En este trabajo construyo redes neuronales profundas (DNNs) para la clasificación automatizada de especies de aves a parir de su canto. Implemento una técnica nueva de aumentado de datos para el entrenamiento que consiste en utilizar cantos sintéticos. El canto se sintetiza utilizando un modelo dinámico del aparto fonador del pájaro y alimentándolo con los parámetros adecuados. Los datos de entrenamiento para cada especie se derivan de variaciones aleatorias en los parámetros de los cantos sintéticos, inspirándose cada uno en un único canto de un individuo de cada especie. Utilizando chingolos (*Zonotrichia capensis*) y benteveos (*Pitangus sulphuratus*), obtuve un clasificador basado en redes convolucionales con eficiencia del 97 %. Esto representa una prueba de concepto satsfactoria para la idea de entrenar redes a partir de cantos sintetizados.

Índice

Capítulo 1

Introducción

Clasificar especies animales es importante en distintas áreas de la ecología. Lo es por un lado para fines científicos o de conservación, cuando se realizan relevamientos o seguimientos de especies, o cuando se estudian la biodiversidad o hábitos reproductivos o alimentarios, entre otros ejemplos. Por otro lado es importante para fines recreativos: comunidades de personas que pasan su tiempo libre en la naturaleza catalogando sus observaciones.

Esta actividad no es sólo relevante en sí misma, sino que también aporta a la concientización ambiental[1]; un punto clave en esta época de calentamiento global y declive ecológico generalizado[2]. En particular, la comunidad de observadores amateurs de aves (coloquialmente llamados *birders* en inglés) es muy numerosa y se agrupa en asociaciones en todas partes del mundo. En la Argentina, la organización *Aves Argentina* crea los Clubes de Observadores de Aves (COA), que cuentan con varios miles de miembros activos[1].

Tanto para la actividad recreativa como para la científica resulta relevante la identificación y clasificación de especies a partir de su canto. En general este proceso es laborioso y lento y requiere pericia y experiencia en el área. Ante el constante incremento de volúmenes de datos a analizar, el factor limitante en estos estudios puede fácilmente resultar el trabajo de clasificación en sí.

Es aquí donde herramientas automatizadas resultan de gran utilidad. Hace ya algunos años se realizan concursos de clasificación de grabaciones con tareas de variada dificultad[3-7]. Estas son organizadas por entidades que se dedican a la conservación o al desarrollo de herramientas automatizadas de clasificación, y en algunos casos, especializadas en cantos de aves.

Las tareas más simples consisten en detectar la mera presencia o ausencia de canto de aves en la grabación. Aumentando en complejidad sigue identificar la única especie presente en

una grabación, o detectar la especie más prominente, teniendo ruido de otras aves cantando de fondo. En las tareas más complejas se pide identificar todas las especies presentes y ubicarlas temporalmente en una grabación larga. En el caso de la competencia *BirdCLEF* 2019 las grabaciones en muchos casos duraban más de una hora y pedía clasificar un total de 1500 especies en 350 horas totales de grabación.

La herramienta de aprendizaje automático o *machine learning* (ver sección 2) resulta la más adecuada para resolver este tipo de tareas[8-10]. Todas las entradas ganadoras que presentaron sistemas totalmente automatizados lo hicieron con distintas técnicas de aprendizaje automático. En particular, los ganadores de los últimos años utilizaron distintas técnicas de aprendizaje profundo (*deep learning*) y redes neuronales. Una característica común a todas estas técnicas es que es necesaria una gran cantidad de datos iniciales con los cuales entrenar el programa para que aprenda a clasificar por sí solo. En los casos en los que la cantidad o calidad de la información no es suficiente o suficientemente diversa como para obtener resultados satisfactorios, se recurre a aumentar artificialmente el volumen datos, técnica que se denomina precisamente aumentado de datos.

1.1. El problema

En esta tesis utilizo un tipo de red de aprendizaje profundo relativamente sencilla para clasificar cantos de dos especies autóctonas de la provincia y ciudad de Buenos Aires: el benteveo (*Pitangus sulphuratus*) y el chingolo (*Zonotrichia capensis*). Elegí estas dos especies de aves porque ambas tienen un canto estereotipado con poca variabilidad interespecimen. Lo que esto quiere decir es que entre un individuo y otro de la misma especie el canto tiene las mismas características generales. Un ejemplo claro se puede ver en la figura 1.1, donde aparecen los sonogramas de tres individuos de cada especie. Un sonograma es una represencatión gráfica del canto donde el eje vertical representa frecuencia, el horizontal tiempo y la intensidad de cada píxel, la potencia de esa frecuencia en ese instante (ver sección 4). Vemos que el canto del benteveo se compone de tres sílabas o fragmentos de canto cualitativamente similares de un individuo al otro. El chingolo, en cambio presenta una estructura general consistente, pero varía en el detalle. Todos los cantos se componen de un tema, que representa el principio del canto, y un trino, compuesto por una única sílaba corta repetida velozmente. La variabilidad entre individuos estará dada en la cantidad de repeticiones en el trino y la composición del tema.

Es interesante señalar que el benteveo pertenece a un conjunto de aves que poseen su canto

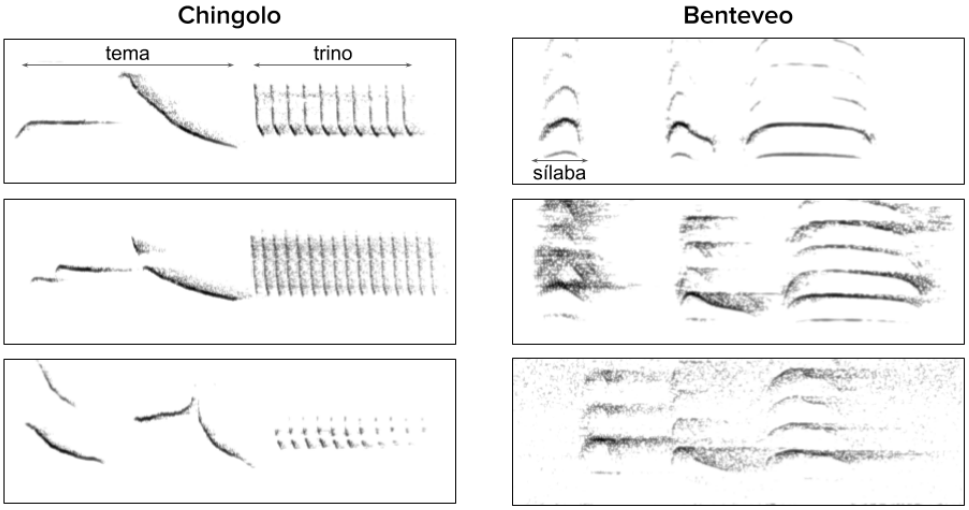

Figura 1.1: *Tres cantos de chingolos y tres de benteveo, cada uno de un individuo distinto. Se ve por un lado lo estereotipado de los cantos, y por otro las diferencias en el tema en distintos chingolos.*

de forma innata, en contraste con las aves que necesitan ser expuestas a un tutor para aprender su canto. El chingolo es una de estas y en particular, una vez que un individuo aprendió su canto, realizará sólo variaciones menores entre distintas emisiones, con lo que en principio es posible distinguir un chingolo de otro a partir de su canto únicamente.

La clasificación de estas dos especies representa un escenario perfecto para probar una nueva técnica de aumentado de datos: utilizaré un modelo dinámico del aparato fonador de un pájaro para generar cantos sintéticos, inspirados en cantos reales. El aumentado de datos se genera a partir de variaciones en los parámetros de la síntesis para generar cantos cualitativamente distintos. El entrenamiento se realizará exclusivamente con cantos sintetizados, y las pruebas de eficacia de clasificación, sobre cantos reales. En el contexto de los tipos de tareas a realizar descritas arriba, esta es de las más simples: voy a clasificar grabaciones donde en cada muestra hay un único canto de un único ave, sin otras aves de fondo, y la extensión de la grabación es tal que dura sólo lo que dura el canto.

Para disponer de un volumen de de datos suficientemente grande como para poder realizar las pruebas luego del entrenamiento, recurro a la base de datos digital llamada Xeno-canto[11]. Es un proyecto abierto a la comunidad donde usuarios de todas partes del mundo suben y clasifican grabaciones de cantos de aves. Al día de la fecha cuenta con más de 510.000 cantos de 10.000 especies. En particular, la base cuenta 650 grabaciones de chingolo y 450 de benteveo, de las cuales utilizo sólo una fracción.

Este es un problema que en principio es realtivamente simple. Los dos grupos de datos a

clasificar tienen grandes diferencias entre sí, pero son relativamente similares dentro de cada grupo. Elegí justamente una tarea simple para realizar una prueba de concepto respecto de la técnica de aumentado. De funcionar bien, esta misma se puede aplicar a todas las tareas descritas arriba.

Lo que sigue de esta tesis está estructurada de la siguiente forma:

- En la sección 2 presento la idea de aprendizaje automático y en particular de redes profundas, describo los bloques con los que se construye una de estas redes y cómo aprenden a resolver la tarea para la que se las crea.
- En la sección 3 hablo sobre el sintetizador de canto realista que utiliza un modelo dinámico del aparato fonador del pájaro para generar los cantos con los que entrenar al clasificador. Detallo el proceso utilizado para sintetizar un canto y para obtener la variabilidad necesaria para entrenar una red neuronal.
- En la sección 4 describo las redes creadas, los resultados de sus entrenamientos y detallo las métricas utilizadas para evaluarlas.
- En la sección 5 elaboro la discusión final del trabajo, evaluando la viabilidad de esta técnica de aumentado de datos y considerando posibles aplicaciones.

Todo el código generado durante este trabajo está disponible en `https://github.com/mwappner/TesisLic`.

Capítulo 2

Redes neuronales y aprendizaje profundo

En este capítulo introduzco el concepto de aprendizaje automatizado y redes neuronales, y delineo los elementos básicos utilizados para construirlas.

2.1. Aprendizaje automático

El aprendizaje automático (*machine learning*) es una rama de la inteligencia artificial. Engloba distintos algoritmos que son capaces de resolver una tarea para la que no fueron explícitamente programados. Para esto debe pasar por una etapa de "entrenamiento" en la que se presentan al sistema datos a partir de los cuales este crea un modelo matemático capaz de hacer predicciones o tomar decisiones respecto a información a la que no fue expuesto durante el entrenamiento. Es precisamente a este comportamiento al que llamamos aprendizaje: cuando el programa se desempeña mejor en una tarea a medida que gana experiencia en ella. El aprendizaje será exitoso no sólo cuando el programa pueda reproducir la tarea inicial (es decir, aprender la información con la que se lo entrenó), sino cuando pueda además obtener resultados satisfactorios con información que nunca le había sido presentada. Cuando logra esto decimos que el programa *generaliza* bien. Si no lo logra, en cambio, decimos que *sobreajusta*.

El aprendizaje automático es aplicado en áreas tan diversas como la medicina, agricultura, finanzas, lingüística, marketing personalizado y motores de búsqueda, entre muchos otros. En la última década, sistemas de aprendizaje automático vencieron a maestros del go[12] y de videojuegos[13, 14], realizaron diagnósticos automatizados[15] y reconocimiento automático de habla en asistentes virtuales[16], crearon videos falsificados extremadamente creíbles[17], y son el alma detrás de los sistemas de navegación automática de los autos sin conductor[18] y del

sistema de recomendaciones de servicios de entretenimiento como Netflix[19] y YouTube[20]. En función de qué tarea se desee resolver, se eligirá un algoritmo distinto. La lista de estos algoritmos es extensa, por lo que sólo voy a hablar del algoritmo llamado aprendizaje supervisado y, para contrastar, del no supervisado.

En el primer caso, la información de entrenamiento que se entrega al sistema viene apareada con el resultado que se desea obtener. Tomando por ejemplo el caso de un clasificador automático de imágenes de animales, cada imagen viene apareada con el nombre de la clase a la que se la quiere asociar (una imagen de elefante con la etiqueta "elefante"). El entrenamiento consiste entonces en lograr que el programa aprenda a asociar la imagen con la clase deseada. Siguiendo este ejemplo, los algoritmos de aprendizaje no supervisado toman las imágenes, pero no las clases, de modo que deben poder extraer las características de las imágenes por sí solos y así agruparlas en clases sin etiquetas. La información devuelta por este tipo de sistemas deberá ser posteriormente interpretada por el usuario, a diferencia de la devuelta por los sistemas supervisados, que es procesada por humanos previo a entregársela al sistema.

Además de distintos algoritmos de aprendizaje automático, existen distintos modelos. Un modelo en este contexto significa de qué forma se relacionará y procesará la información de entrada para obtener las predicciones buscadas. En este trabajo utilizaremos el modelo llamado redes neuronales, sobre las que ahondo en la sección 2.2.

Debemos notar que las predicciones o decisiones que toma el sistema son sólo tan buenas como la información con la que fue entrenado. Cualquier sesgo presente en el conjunto de datos inicial será reproducido por el programa una vez entrenado. Esto es algo que, en lo posible, debemos tener en cuenta al construir o elegir nuestros datos de entrenamiento.

2.2. Redes neuronales

El aprendizaje profundo engloba un conjunto de algoritmos de aprendizaje automático que utilizan redes neuronales de distintos tipos. Las redes neuronales son modelos matemáticos computacionales vagamente inspirados en la estructura del cerebro animal[21]. Una red se compone de neuronas, cada una de las cuales almacena una cantidad pequeña de información en forma de un único número real y puede comunicarse con otras neuronas a través de conexiones (figura 2.1), análogas a las sinapsis. Las redes suelen organizarse en capas, y se denomina redes neuronales profundas (**DNN**, *deep neural networks*) a los sistemas con varias capas. En el caso de las redes prealimentadas (en inglés llamadas *feedforward networks*), como lo son las que utilizo

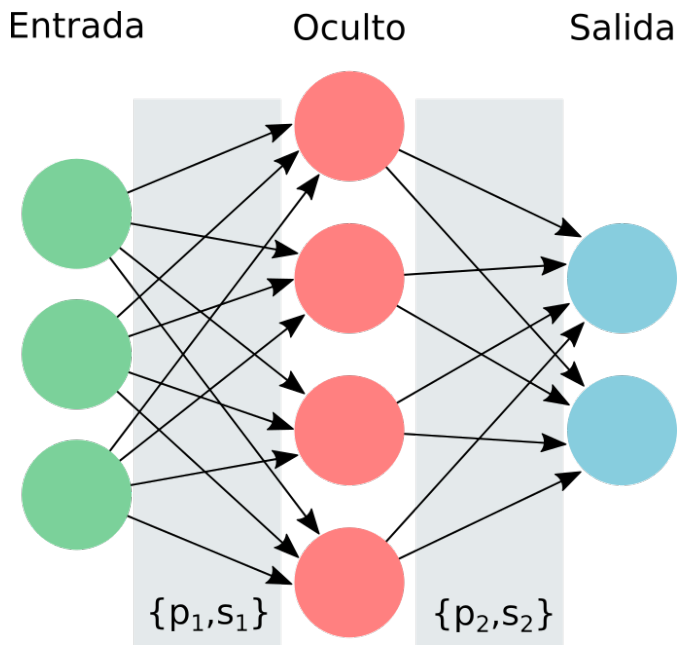

Figura 2.1: *Esquema mínimo de una red neuronal con una capa de entrada con tres neuronas, una capa oculta con cuatro neuronas y una capa de salida con dos neuronas. Los valores de cada neurona de una capa dependen de los de las neuronas en la capa anterior a través de pesos y sesgos (p_1, s_1 para la capa oculta y p_2, s_2 para la capa de salida).*

en este trabajo, cada capa recibe información sólo de las anteriores y se la transmite sólo a las siguientes. Nuevamente, la idea de apilar capas se inspira en la estructura de cómo (se cree que) los cerebros procesan la información: las primeras capas de neuronas son sensibles a patrones simples, las siguientes se activan según patrones en las anteriores, logrando en las últimas capas un gran nivel de abstracción[22]. Como veremos más adelante, un gran atractivo de las redes prealimentadas es que hacen posible el uso de la retropropagación para el entrenamiento, lo cual no es posible con redes conectadas globalmente.

Se suele llamar capa de entrada a la primera, que recibe los datos, por ejemplo la imagen a clasificar, y capa de salida a la última, que devuelve el valor que se desea predecir, por ejemplo la etiqueta de la imagen. Las capas intermedias, las que realizan la abstracción que representa ir de la imagen a la etiqueta, se llaman capas ocultas.

El valor de cada neurona de las capas ocultas y de la de salida es alguna función no lineal de la suma de las señales de la capa anterior, modificadas por algún peso y un sesgo. El proceso de aprendizaje por el cual este tipo de redes aprende representaciones se lleva a cabo ajustando estos pesos y sesgos (llamados parámetros entrenables) en un proceso llamado retropropagación (ver sección 2.2.6). Esto implica que una vez definida la cantidad de neuronas por capa y la forma en que se comunican entre sí, lo que se llama la *arquitectura* de la red, toda la capacidad de aprendizaje depende exclusivamente del ajuste de los pesos y sesgos. Estos parámetros no

entrenables suelen llamarse hiperparámetros de la red.

Como las DNN son sistemas de aprendizaje supervisado, el entrenamiento se realiza alimentando al sistema con pares de datos y resultados esperados (o más bien, los resultados "correctos"). El entrenamiento se lleva a cabo con el objetivo explícito de que las predicciones que realiza la red se correspondan con los resultados, pero sobre todo con la esperanza de que los patrones que aprenda generalicen bien, es decir sean válidos para datos a los que nunca fue expuesta.

Para lograr lo primero, se expone a la red a grandes cantidades de datos y sus resultados asociados. Se introducen los datos y se obtiene un resultado que, con toda probabilidad, estará lejos del resultado correcto. Para evaluar esta cercanía, se utiliza una función de pérdida diseñada de forma que su valor sea menor cuanto más se parezca el resultado predicho al resultado correcto. Luego, utilizando la retropropagación, se modifican los parámetros entrenables de la red en una cantidad que se espera disminuya el valor de la función de pérdida. Se continúa iterando este proceso durante una cantidad fija de iteraciones o hasta obtener resultados por debajo de un umbral dado.

La mayor dificultad yace en asegurarse de que el modelo generalice bien. Para corroborar que el modelo no sobreajusta, es usual separar el total de los datos de entrenamiento de los que se dispone en tres grupos:

- sobre el **conjunto de entrenamiento** se realizan las iteraciones en las que se ajustan los parámetros entrenables de la red;
- después de un ciclo de la iteración, se utiliza le **conjunto de validación** para evaluar la función de pérdida con datos a los que la red no fue expuesta hasta el momento, con la idea de que una red que generaliza bien disminuirá el valor de la pérdida tanto en los datos de entrenamiento como en los de validación;
- si bien los datos de validación no se utilizan explícitamente para ajustar los parámetros, es posible que alguna información de ellos se filtre a la red. Es por esto que el tercer grupo que se separa es el de **datos de prueba**, con los que se evalúa al final del entrenamiento si el modelo efectivamente logró generalizar. Para problemas de clasificación, la métrica con la que se avalúa las redes es la exactitud del sistema que está dada por qué porcentaje de los datos de prueba clasifica satisfactoriamente.

Para lograr una buena generalización es necesaria una gran cantidad de datos, lo cual tiene sentido: cuando se enseña a una persona a hablar, se la debe exponer repetidas veces a una gran variedad de sonidos y relacionarlos con los conceptos asociados, antes de que ella misma

pueda empezar a utilizarlos y aprender palabras sin que le sean explicadas. De hecho, si bien la teoría detrás del aprendizaje profundo existe desde entre mitades y fines del siglo pasado, sólo en la última década la disponibilidad de datos a gran escala y el abaratamiento de los recursos computacionales necesarios para procesarlos hizo viable la implementación de DNNs[22].

Para minimizar problemas de sobreajuste existen distintas técnicas que se llaman de regularización. Estas consisten en los casos más simples en realizar las iteraciones de entrenamiento en el momento en que el valor de la función de pérdida del conjunto de validación comienza a aumentar, e incluye técnicas como penalizar a la red por utilizar valores grandes en los pesos (regularización de pesos) y tipos de capas especiales (ver 2.2.5).

Alternativamente, si la falta de generalización se debe a falta de datos de entrenamiento, no existen técnicas de regularización que eviten el sobreajuste. Por esta razón se recurre al proceso llamado aumentado de datos. Este consiste en tomar el conjunto de datos de entrenamiento y modificarlos de forma de crear nuevos datos realistas con los que entrenar la red. Dependiendo de las técnicas de aumentado utilizadas, puede llegar a agrandarse los conjuntos de entrenamiento en más de un orden de magnitud.

Esta es justamente la inspiración de este trabajo: por razones que se verán más adelante (ver sección 4), los clasificadores de especies de pájaros a partir de su canto utilizan representaciones en imágenes de los sonidos llamados sonogramas. Las técnicas de aumentado de datos que se suele usar en problemas de clasificación de imágenes están diseñadas para trabajar con fotografías o videos, y no necesariamente son óptimas para tratar con sonogramas. Estas consisten en transformaciones que simulan distintas perspectivas de una misma imagen: rotaciones, estiramientos y deformaciones leves, tomar la imagen especular, desplazamientos verticales u horizontales, hacer un zoom a la imagen y alterar le contraste, brillo y saturación. De todas estas, la única que representa una transformación realista para sonogramas es el desplazamiento horizontal.

Existen técnicas de aumentado explícitamente creadas para cantos de aves[23], pero en este trabajo presento una técnica novedosa basada en sintetizar cantos a partir de un modelo del aparato fonador del pájaro que los genera.

En lo que sigue describo cómo se representan los datos en una DNN, algunos tipos de capas y ahondo en el concepto de retropropagación.

2.2.1. Una nota sobre representación

Como veremos, el álgebra lineal es una parte esencial del lenguaje de las DNN. Es por esto que para tratar los datos, debe acondicionárselos de forma adecuada. Usualmente se utilizan tensores, que en este contexto son una generalización de matrices y vectores: un tensor de una dimensión es un vector, de dos dimensiones una matriz y de dimensiones mayores, simplemente tensores. Un tensor de tres dimensiones puede ser imaginado como matrices apiladas en capas sucesivas.

Es fácil imaginarse cómo datos secuenciales, como una serie temporal, puede representarse como un vector. Por otro lado, una imagen en escala de grises deberá ser escrita como una matriz donde cada elemento representa el valor de un píxel. Una imagen a color, en cambio, suele representarse por tres imágenes escaladas dentro de nu rango prescripto, donde cada una de ellas corresponde a los canales de colores rojo, verde y azul. Esta es entonces un tensor 3D con tamaño $ancho \times alto \times 3$. De la misma forma, un video se puede representar como un tensor 4D de $ancho \times alto \times 3 \times cuadros$.

Una DNN tomará uno o más de estos tensores como dato de entrada y construirá a partir de él tensores de otros tamaños y otra cantidad de dimensiones que, idealmente, almacenan representaciones cada vez más abstractas de la información original.

2.2.2. Capas densas

El tipo de capa más paradigmático de las redes neuronales profundas se denomina capa densa o densamente conectada. En ellas el valor de cada neurona está dado por una función no lineal f aplicada sobre la combinación lineal con pesos w de todas las neuronas de la capa anterior, modificada por un sesgo b. Esto es, si la capa n tiene neuronas con valores $x_i^{(n)}$, la capa $n+1$ tendrá valores

$$x_j^{(n+1)} = f\left(\sum_i w_{ij} x_i^{(n)} + b_j\right) \tag{2.1}$$

o equivalentemente, en versión matricial

$$X^{(n+1)} = f(\mathbb{W} X^{(n)} + B). \tag{2.2}$$

Aquí $X^{(n)}$ es un vector de dimensión N igual a la cantidad de neuronas en la capa n, $X^{(n+1)}$ un vector de dimensión M, la cantidad de neuronas en esa capa, $\mathbb{W}$ es la matriz de pesos de

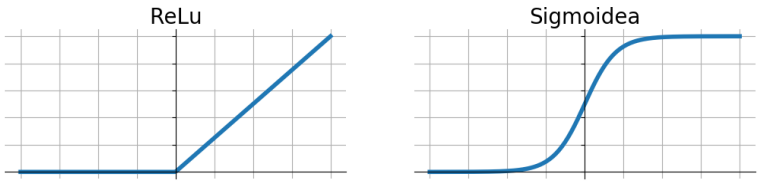

Figura 2.2: *Funciones de activación ReLU y sigmoidea, dos de las más comunmente utilizadas.*

tamaño $M \times N$, y B es el vector de sesgos de tamaño M. f es la misma función no lineal que en el caso anterior donde, con un abuso de notación, aplicarla sobre matrices significa aplicarla elemento a elemento. Se ve entonces que le nombre de este tipo de capas proviene del hecho de que cada neurona está conectada con todas las neuronas de la capa anterior y su estado depende del de todas ellas. El aprendizaje en una capa densa consiste por supuesto en ajustar los pesos y los sesgos y los hiperparámetros son la cantidad de neuronas por capa.

La función f se llama función de activación y la razón por la que insisto con el hecho de que es no lineal es porque la capacidad de una red neuronal de aprender representaciones complejas yace en la función de activación. Si esta fuera lineal, todo lo que podría aprender serían mapas lineales de la información de entrada a las predicciones. Con el agregado de la no linealidad sin embargo, las redes neuronales se transforman en aproximadores universales[24]. Esto eso, una red con al menos una capa oculta de tamaño suficiente y cualquier función de activación no lineal puede aprender cualquier mapeo con precisión arbitraria.

Existen distintos tipos de funciones de activación, pero las más utilizadas[25] son ReLU, que es implemente $ReLU(x) = max(0, x)$, la sigmoidea o logística, que es una función derivable que toma valores entre 1 y 0, o su generalización a varias dimensiones, *Softmax* (figura 2.2).

2.2.3. Capas convolucionales

El uso de capas convolucionales revolucionó el área de visión artificial cuyo objetivo es obtener representaciones abstractas de videos o imágenes, para realizar tareas específicas. Como su nombre sugiere, el valor de cada neurona de una capa convolucional será la convolución de la matriz de pesos con las neuronas de la capa anterior, a la que se le suma un sesgo y aplica una función de activación.

La operación de convolución (discreta y en 2D para este ejemplo) toma como argumentos una imagen y un tensor usualmente más pequeño que esta llamado *kernel* o en algunos contextos, filtro, y devuelve una nueva imagen que es la convolución. Para realizar la operación, se calcula el promedio del producto elemento a elemento entre el kernel y un subconjunto de píxeles de

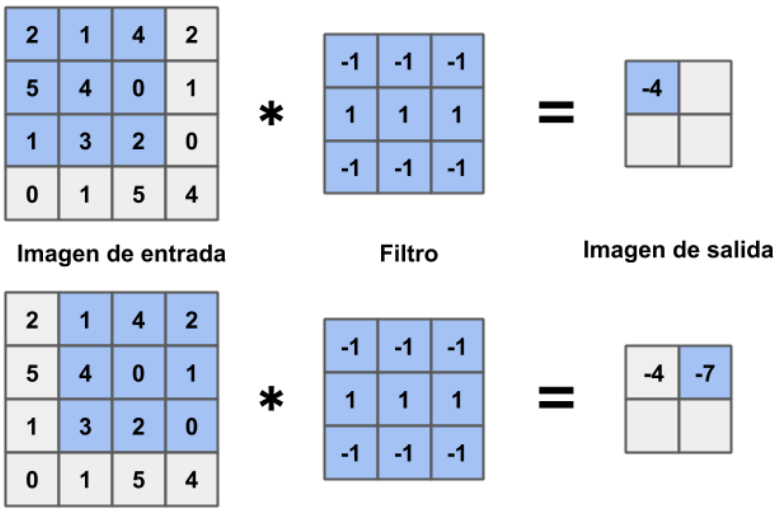

Figura 2.3: *Representación gráfica de la operación de convolución. En este caso se ilustra el cálculo realizado para obtener los valores de los primeros dos pixeles de una imagen resultante a partir de una imagen de salida de* 4×4 *y un filtro de* 3×3*. Se ve que la imagen resultante es más pequeña que la imagen de entrada, porque no es posible aplicar el filtro en elementos cerca del borde. Esto puede mitigarse de distitnas formas, si fuera necesario.*

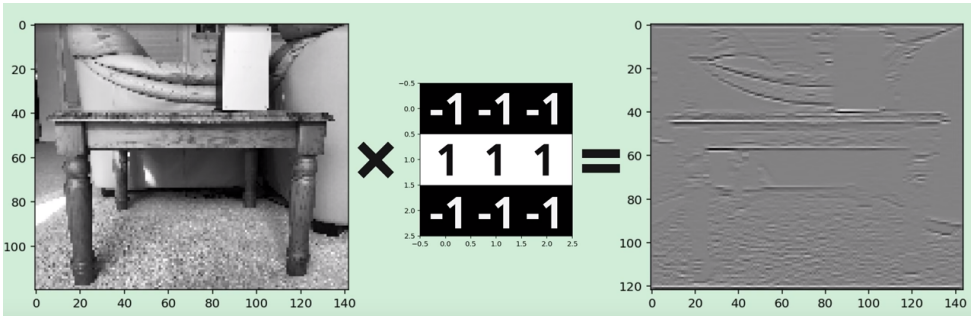

Figura 2.4: *Efecto de la convolución de una imagen con un kernel utilizado para filtrar bordes horizontales. Adaptado de [26].*

la imagen del mismo tamaño que el kernel en cuestión. Se repite esta operación para todos los subconjuntos de la imagen original centrados en todos los píxeles y se asigna el valor de cada una de ellas a un elemento del tensor resultante (figura 2.3). Debido a efectos de borde y dependiendo de cómo se los trate, el tensor resultante tendrá un tamaño igual a la imagen original o será más chico en una cantidad de píxeles semejante al tamaño del kernel.

Se llama filtro al tensor más pequeño, porque el efecto de esta operación es buscar en la imagen grande patrones similares al del kernel. Por ejemplo, un filtro que busca rayas horizontales resaltará en la convolución resultante todas las rayas horizontales presentes (figura 2.4).

Se llama 2D a este tipo de operación porque el barrido se realiza en dos dimensiones, se recorren todos los pixeles de la imagen. Esto no impide sin embargo que tanto el filtro como el tensor imagen tengan más de dos dimensiones. Por ejemplo, se puede realizar una convolución 2D entre un tensor de $128 \times 128 \times 3$ (una imagen a color) y un filtro de $5 \times 5 \times 3$ barriendo

sobre las primeras dos dimensiones.

Una capa convolucional en una red neuronal suele tener varios filtros del mismo tamaño. Sobre cada uno se realiza la convolución con el tensor de entrada, se suma un sesgo y aplica una función de activación. Cada uno de los tensores resultantes se guarda en un tensor de dimensión mayor, que representa los valores de las neuronas de la capa. Así, si se ingresa una imagen de $128 \times 128 \times 3$ y se utilizan 16 filtros de $5 \times 5 \times 3$, el tensor resultante tendrá un tamaño de $124 \times 124 \times 16$ (recordando que la imagen resultante puede resultar más pequeña por efectos de borde).

Durante el aprendizaje, el sistema ajusta los valores de los filtros para obtener características de interés y los valores de los sesgos para asignar a cada filtro una relevancia mayor o menor relativa al resto. Los hiperparámetros son entonces la cantidad y el tamaño de los kernels utilizados.

La gran ventaja de este tipo de capas por sobre las capas densas, y la razón por la que resultan tan importantes en el campo de visión artificial, es que las características que el sistema aprende a reconocer son traslacionalmente invariantes: si el sistema aprende a reconocer un ojo humano, podrá hacerlo en cualquier parte de la imagen, de la misma forma que puede reconocer rayas horizontales en cualquier parte de la imagen. Esta ventaja es sin embargo también un problema, porque carece de capacidad para (o es muy ineficiente a la hora de) obtener una idea global de la imagen. Es por esto que un clasificador usual combina primero capas convolucionales, que realizan la extracción de características, y luego capas densas, que realizan la interpretación posterior de las características.

2.2.4. Pooling

Las capas de *pooling* se utilizan usualmente en conjunto con capas convolucionales para reducir la dimensión de la información de entrada. La más común es la *max pooling*, que toma subsecciones que no se solapan entre sí del tensor de entrada y se queda sólo con el máximo valor en cada una de ellas. Así, si las subsecciones son por ejemplo de 2×2 elementos, el tensor de salida tendrá un cuarto de la cantidad total de elementos. Esto permite a las capas convolucionales obtener rápidamente (es decir, en pocas capas) características a mayor escala en la imagen original, sin recurrir al filtros demasiado grandes, lo cual es computacoinalmente costoso[27].

Existen otros tipos de operaciones en las cuales en vez de tomar el máximo se toma por

ejemplo el promedio, u otras cantidades útiles, aunque lo más usual es utilizar *max pooling*. Es interesante notar que este tipo de capas no tiene parámetros libres que el modelo ajuste durante la etapa de aprendizaje.

2.2.5. Dropout

Las capas *dropout* son un tipo de regularización que se agrega a redes neuronales que tienen problemas de sobreajustado. Se utilizan usualmente en conjunto con capas densas y su efecto es simplemente desactivar algunas neuronas de la capa previa. Esto se lleva a cabo de forma aleatoria: cada neurona tiene una probabilidad p de ser desactivada, y se seleccionan las neuronas a desactivar en cada iteración del aprendizaje. La desactivación se realiza únicamente durante el entrenamiento; cuando se pide a la red que prediga resultados, todas las neuronas están activas y la capa *dropout* no realiza ninguna operación.

Desactivar algunas neuronas al azar permite que el sistema no aprenda la codependencia de algunas características, haciendo más eficiente el uso de sus recursos (los parámetros disponibles). Esto disminuye el sobreajustado considerablemente, pero aumenta la cantidad de iteraciones necesarias para converger[28].

Como el único parámetro es la probabilidad global p de desactivar una neurona en particular, y este no se ajusta durante el aprendizaje, este tipo de capas no tiene parámetros entrenables.

Si bien existen otros tipos de capas de DNNs prealimentadas, con los elementos descritos hasta aquí es posible construir una red de clasificación automática de imágenes del tipo que hoy en día se utiliza para resolver un sinnúmero de problemas. Nos falta sólo saber cómo entrenar la DNN que construyamos.

2.2.6. Retropropagación

La propagación hacia atrás o retropropagación fue propuesta en los orígenes del uso de redes neuronales[29] y es el corazón detrás del entrenamiento eficiente de redes neuronales. Como vimos, una red neuronal profunda es un gran conjunto de operaciones simples que llevan de la información de entrada a una predicción. El objetivo del entrenamiento es que este camino lleve a predicciones que se asemejen lo más posible a los resultados conocidos. Para evaluar esta "semejanza" utilizamos la función de pérdida que se elegirá en función del problema que se desee resolver, con lo que el entrenamiento se traduce en minimizar la función de pérdida. La función de pérdida se elige dependiendo del problema al que se enfrente.

El algoritmo de retropropagación calcula el gradiente de la función de pérdida para un dato de entrada respecto de todos los parámetros entrenables de la red utilizando la regla de la cadena. De esta forma, inocentemente podemos decir que lo que debemos hacer para mejorar el rendimiento de la red es movernos en dirección opuesta al gradiente dentro del espacio de parámetros, con la esperanza de acercarnos al mínimo de pérdida. Estrictamente, existen distintos algoritmos llamados optimizadores que difieren en cómo deciden dar un paso una vez calculado un gradiente. Estos van desde los más básicos, como SGD (descenso estocástico por gradiente), hasta algoritmos que tienen en cuenta la historia y ajustan las correcciones sobre la marcha, según sea conveniente, como los algoritmos con momento, RMSprop y Adam; los últimos dos siendo los más utilziados hoy en día.

Estrictamente, es usual calcular las correcciones a los parámetros sobre un subconjunto de los datos, promediando las correcciones de todos ellos; a esto se lo llama dar un paso. Se repiten tantos pasos como sea necesario para utilizar todos los datos de entrenamiento. Una vez hecho esto se dice que se concluyó una época. Usualmente se entrena una red durante varias épocas, realizando predicciones sobre el conjunto de validación al final de cada una.

Cabe destacar que la estrategia utilizada en esta tesis es la de clasificar cantos como sonogramas utilizando técnicas de visión artificial. Existe también otra familia de redes llamada recurrente (en contraste con las redes neuronales prealimentadas). Este tipo de redes suele utilizarse para aprendizaje profundo de series de datos secuenciales[25], el más común de los cuales es datos que dependen del tiempo: fluctuaciones en la bolsa, valores de temperatura y presión atmosférica, caudal de acuíferos de una región. El problema propuesto en este trabajo, el de clasificar especies de aves a partir de su canto, puede pensarse como un problema de datos secuenciales: las fluctuaciones en la presión del aire a lo largo del tiempo. Sin embargo, resultados previos en el área muestran que es mucho más eficaz utilizar algoritmos de redes neuronales profundas, en particular redes convolucionales[8]. Las razones detrás de esto no serán discutidas en este trabajo.

Capítulo 3

Síntesis de cantos

En este capítulo explico el modelo utilizado para sintetizar cantos de aves, basado en un modelado dinámico del aparato fonador.

3.1. El modelo

Un gran porcentaje de las especies de aves requieren de cierto grado de aprendizaje por imitación para generar sus cantos, y las aves cantoras u oscinas son la mayor parte de ellas [30]. Esto, junto con el hecho de que su aparato fonador presenta similitudes con el del humano, constituye algunas de las razones por las que son un modelo de estudio interesante y muy utilizado en áreas de neurociencias y comportamiento.

Para emitir cualquier sonido "voceado", el humano aumenta la presión de aire en los pulmones y lo expulsa, haciendo vibrar las cuerdas vocales en la laringe. Este sonido es luego filtrado y modulado por el resto del tracto vocal. Análogamente, las aves generan sonido haciendo vibrar los labios (Lm, Ll) o membranas de la siringe (S), su órgano fonador, al aumentar la presión y liberar aire de los sacos aéreos[31]. Cuando esto sucede, decimos que el ave está *fonando*, al igual que los humanos. La onda de presión generada por la vibración de los labios se propaga por la tráquea (T) hasta la cavidad orofaríngea (OEC) y sale del ave por el pico (P) (figura 3.1).

La siringe se encuentra en la base de la tráquea, donde comienzan los bronquios, por lo que la mayor parte de las especies de aves cantoras poseen dos pares de labios que pueden controlar independientemente y generar dos sonidos con fuentes independientes de forma simultánea[32]. A su vez, cada par de labios se compone de un labio lateral y un labio medial que no son simétricos.

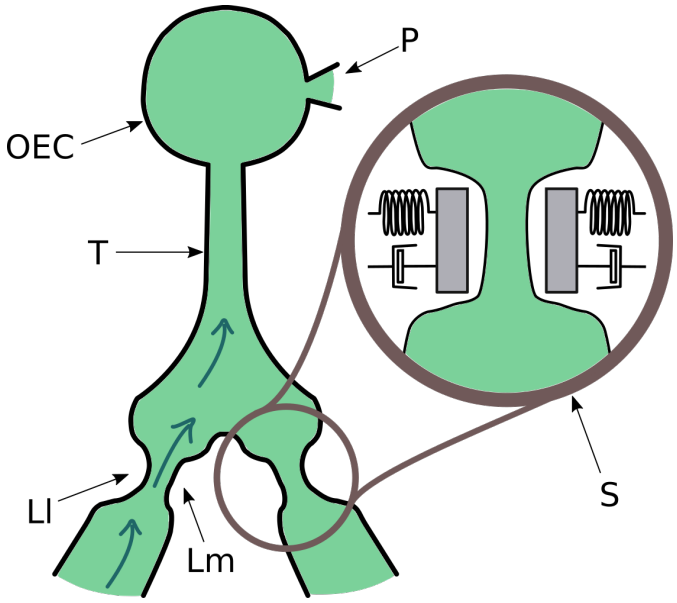

Figura 3.1: *Esquema del aparato fonador del ave y modelo de la dinámica de la siringe. S: Siringe, Lm: labio medial, Ll: labio lateral, T: tráquea, OEC: cavidad oroesofaríngea, P: pico.*

El ave es capaz de modular su canto a través de la presión de los sacos aéreos y de un conjunto de músculos unidos a los bronquios. Estos son capaces de tensar o relajar los labios, cambiando así la frecuencia a la que vibran con el paso del aire, o acercarlos entre sí, para frenar completamente la fonación.

Para estudiar características del canto y, sobre todo, los mecanismos fisiológicos y neuronales que lo gobiernan, el grupo desarrolló una serie de modelos dinámicos del aparato fonador (ver referencia [30] para un resumen exhaustivo). Un modelo inicial[33] asume que ambos labios lateral y medial son iguales y se comportan simétricamente, y se preocupa de describir sólo uno de los pares de labios. Se modela el sistema utilizando las ecuaciones de Newton para una variable x que corresponde al desplazamiento de los labios respecto de la posición de equilibrio. Estos están sometidos a una fuerza restitutiva con una componente no lineal, disipación lineal proporcional a la velocidad del movimiento de los labios y dependiente de la presión del aire en el conducto, una disipación no lineal, que acota el rango de movimiento posible y unas fuerzas responsables de acercar y alejar los labios, que no depende de las variables de movimiento:

$$\begin{aligned} \frac{dx}{dt} &= y \\ \frac{dy}{dt} &= -k(t)\gamma^2(x+\epsilon_1 x^3) - \beta(t)\gamma y - \gamma(x^2+\epsilon_2 x^3)y + f(t) \end{aligned} \tag{3.1}$$

Aquí $k(t)$ es la constante de restitución, $\beta(t)$ la de disipación, $f(t)$ las fuerzas externas y γ la escala temporal del problema. ϵ_1 y ϵ_2 son dos parámetros adimensionales que definen la

intensidad de la no linealidad del modelo y sus valores están disponibles más adelante en la tabla 3.1. Todas las dependencias temporales en este sistema de ecuaciones corresponden a variables que el pájaro puede modificar a voluntad para modular las características sonoras de su canto.

Está bien demostrado[34] que controlando la fuerza externa, que representa la capacidad del pájaro de alejar y acercar los labios siríngeos entre sí, la tensión y la presión, el modelo es capaz de reproducir los gestos sonoros y la riqueza espectral de los cantos verdaderos. Sin embargo, para este trabajo nos interesa un modelo más simple en el que utilicemos tan pocas variables como sean necesarias. Trabajos posteriores mostraron que es suficiente trazar trayectorias en el espacio de parámetros (k, β) para sintetizar sonidos realistas[30].

Los tiempos característicos de variación de estos parámetros (que son controlados activamente) son de entre decenas y centenas de milisegundos. Por otro lado, los tiempos característicos de los comportamientos oscilatorios del sistema (o equivalentemente, las frecuencias presentes en el canto) son del orden de los kHz, por lo que es razonable analizar la dinámica de estos sistemas para valores estacionarios de los parámetros y estudiar luego cómo varía el espacio de fases en función de estos.

Lo primero que es interesante observar es que la presencia de las no linealidades en el sistema de ecuaciones implica que las soluciones periódicas tendrán frecuencias que dependen de $k(t)$ y de $\beta(t)$. A su vez, existe un valor mínimo de β debajo del cual no se generarán oscilaciones autosostenidas.

Integrando numéricamente el sistema y barriendo en estos dos parámetros se obtienen las curvas de isofrecuencia que, como era esperado, comienzan partir de un valor umbral de la presión de los sacos aéreos (parámetro relacionado con β) (figura 3.2). Las curvas de isofrecuencia son trayectorias en el espacio de parámetros a lo largo de las cuales el comportamiento oscilatorio que presenta el sistema tendrá siempre la misma frecuencia. A partir de esto, se puede sintetizar una sílaba (una unidad del canto vagamente definida como un sonido con silencio antes y después, figura 1.1) realizando una trayectoria en este espacio de parámetros e integrando numéricamente el sistema para cada valor de $(k(t), \beta(t))$. El sonido generado se lo puede pensar como ondas de presión a las frecuencias a las que oscila el sistema.

Se grabó durante el canto de un ave la señal eléctrica transmitida al músculo responsable de controlar la tensión de los labios, y la presión en la entrada de los bronquios. Se utilizó estos valores para alimentar los parámetros activamente controlados del sistema y se obtuvo cantos realistas (figura 3.3).

Si bien el contorno de las frecuencias fundamentales es consistente con el del canto real, el

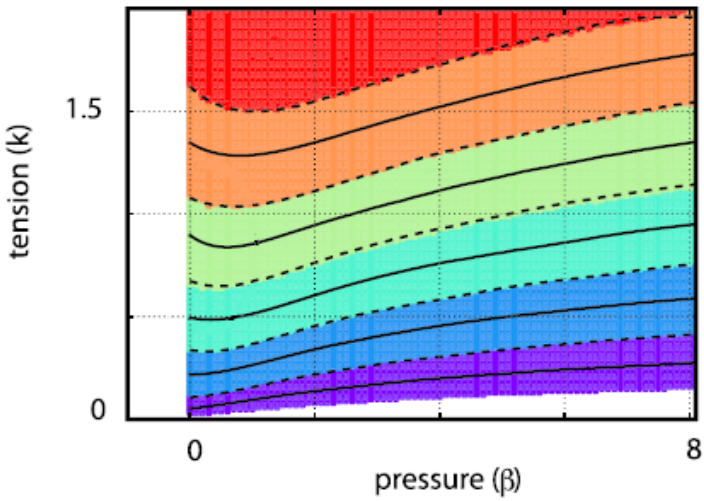

Figura 3.2: *Curvas de isofrecuencia en el espacio de fases de (k, β). Cada curva delimitada corresponde a trayectorias en el espacio que generan comportamientos oscilatorios con la misma frecuencia. Vemos que por debajo de cierto valor de beta, el sistema no entra en régimen de oscilaciones autosostenidas, y que en general a mayores valores de k, la frecuencia generada es mayor. Adaptado de [30].*

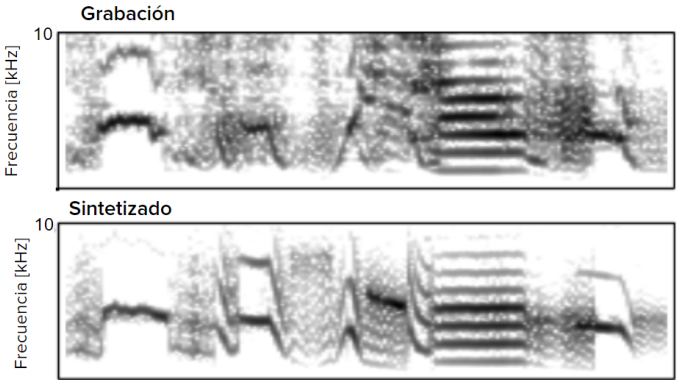

Figura 3.3: *Grabación del canto de un canario (superior) y su síntesis correspondiente (inferior), adaptado de [30].*

contenido espectral no coincide. El contenido espectral de un sonido mide cuánta energía está presente en la frecuencia fundamental y cuánta se distribuye en otros modos. Esta resultó ser una característica importante del canto, porque se observó que las aves reaccionaban distinto la reproducción de su propio canto y a la del sonido sintetizado. Esta observación se realizó midiendo la actividad de un núcleo de neuronas que son altamente selectivas respecto al canto propio: si se expone al ave dormida a grabaciones de su canto, se observa un pico de actividad, mientras que no presenta actividad alguna frente a dos controles dados por el canto de un conespecífico y su propio canto reproducido hacia atrás (este último siendo interesante porque contiene las mismas frecuencias, pero con con una estructura temporal distinta). Corrigiendo, sin embargo el contenido espectral, se observa un pico de actividad, lo que sugiere que, en cierto grado al menos, el pájaro reconoce el canto sintetizado como el suyo propio[35].

Para que el contenido espectral del sonido sintetizado coincida con el real, es necesario agregar al modelo el filtrado dado por la propagación del sonido en la tráquea y la cavidad orofaríngea[36]. La traquea se modela como un tubo recto por el cual el sonido se propaga hasta llegar la final, donde parcialmente se refleja e interfiere con el sonido en la fuente, y parcialmente se transmite:

$$\begin{aligned} p_i(t) &= Ax(t) + p_{back}\left(t - \frac{L}{c}\right) \\ p_{back}(t) &= -rp_i\left(t - \frac{L}{c}\right) \\ p_{out}(t) &= (1-r)p_i\left(t - \frac{L}{c}\right) \end{aligned} \tag{3.2}$$

donde p_i es la presión en el comienzo de la tráquea, p_{back} la reflejada y p_{out} la transmitida. Los parámetros r para el coeficiente de reflexión, L para la longitud de la tráquea, c para la velocidad del sonido y A para la intensidad del sonido o flujo de aire estarán dados por valores fisiológicos (tabla 3.1). p_{out} es ahora la presión de aire que excita un resonador de Helmholz, que modela la cavidad orofaríngea. Todo este sistema puede expresarse como un conjunto de ecuaciones dinámicas que se acoplarán a las ecuaciones (3.1) y cuya deducción no detallo en este trabajo:

$$\begin{aligned} \frac{di_1}{dt} &= i_2 \\ \frac{di_2}{dt} &= -\frac{i_1}{L_g C} - \left(\frac{r_d}{L_b} + \frac{r_d}{L_g}\right) i_2 + \left(\frac{1}{L_g C} - \frac{r_d r_b}{L_g L_b}\right) i_3 + \frac{dp_{out}}{dt}\frac{1}{L_g} + \frac{r_d p_{out}}{L_g L_b} \\ \frac{di_3}{dt} &= -\frac{L_g}{L_b} i_2 - \frac{r_b}{L_b} i_3 + \frac{p_{out}}{L_b} \end{aligned} \tag{3.3}$$

Parámetro	Valor benteveo[1]	Valor chingolo
ϵ_1	0.3	0.3
ϵ_2	0.1	0.1
γ	24000	24000
r	0.01	0.65
L	0.045m	0.025m
c	$350\frac{m}{s}$	$350\frac{m}{s}$
C	$1{,}25 \times 10^{10}$	7×10^9
L_b	0.1	10^{-4}
L_g	1/35	1/20
r_b	10^7	5×10^6
r_d	10^4	$2{,}4 \times 10^5$

Tabla 3.1: *Tabla de valores de los parámetros necesarios para la síntesis de los cantos de cada especie. Algunos valores son globales al modelo, como los valores de γ, ϵ_1 y ϵ_2 y otros dependen de las dimensiones del ave, como los parámetros del filtro, con lo que es relativamente fácil reescalarlos para distintas especies.*

donde los nombres de las variables (i_n, r_x, L_x y C_x) se deben a que es usual en el área de acústica describir los filtros (en particular, el resonador de Helmholtz) como el circuito eléctrico equivalente. Todos los valores corresponden a variables fisiológicas y fueron obtenidos a partir de simulaciones en la referencia [34]. Para las especies que se modelan en este trabajo (benteveo y chingolo), los parámetros necesarios se encuentran en la tabla 3.1. En este sistema de ecuaciones, el sonido generado será proporcional a i_3.

El modelo presentado hasta aquí resulta ser suficientemente bueno como para generar cantos realistas. Se pueden hacer mejoras a partir de correcciones a segundo orden de las ecuaciones de propagación, en las que se tiene en cuenta términos de vorticidad[35], o a partir de introducir la asimetría entre los labios lateral y medial[30] que antes descartamos. Sin embargo, con la idea de mantener un sintetizador simple y sobre todo, con pocos parámetros de control (es decir baja dimensionalidad), dejaremos estas correcciones de lado.

De hecho, para seguir en la misma ruta de simplificar la descripción, descartaremos un control fino sobre la variable de presión de los sacos aéreos β. En vez de utilizar trayectorias en el espacio de parámetros definidas *ad hoc* para cada sílaba, tomaremos un valor fijo para β que llamaremos β_0, cuyo valor numérico es irrelevante, dado que estamos trabajando en un sistema adimensional. De esta forma, cuando queremos que el modelo fone, tomamos $\beta = \beta_0$ y cuando no, utilizamos $\beta = 0$. Mover luego el parámetro k será equivalente a recorrer dos líneas verticales en el diagrama de fases en la figura 3.2, una en la región coloreada (fonación) y una

en la región blanca (silencio). Esto va en detrimento de la fidelidad del contenido espectral del sonido generado por el modelo, pero veremos más adelante que el clasificador será entrenado en sonograma filtrados de forma que sólo la frecuencia más prominente aparezca, por lo que podemos prescindir de dicha fidelidad.

Finalmente, para facilitar la síntesis, vamos a modificar la dependencia del modelo para pasar de k, la intensidad de la fuerza restitutiva, a ω, la frecuencia de respuesta del sistema para dicho valor de k dado $\beta = \beta_0$. La relación necesaria es

$$k = 6{,}56 \times 10^{-8}\omega^2 + 4{,}23 \times 10^{-5}\omega + 2{,}67 \times 10^{-2} \tag{3.4}$$

que se obtiene a partir de una cantidad de simulaciones sobre el modelo[37].

De esta forma, para sintetizar un canto alcanza con diagramar la frecuencia fundamental de las sílabas a sintetizar y cómo son moduladas en el tiempo. Utilizando (3.4) se obtiene una serie de valores para $k(t)$ y se establece $\beta = \beta_0$ cuando $k(t) \neq 0$ (es decir, cuando se desea que el modelo fone). Con estos dos parámetros se integra numéricamente un paso temporal las ecuaciones (3.1) y se alimenta a través (3.2) el filtro dado en (3.3). Finalmente, el sonido generado con la riqueza espectral adecuada y el énfasis en las frecuencias correctas está dado por la variable i_3. Es importante notar que, todo este proceso es algorítmico, salvo por el primer paso, por lo que el único trabajo manual que implica la síntesis de un canto es diagramar la modulación de la frecuencia fundamental del sonido deseado (figura 3.4).

3.2. Síntesis

Recordemos que el objetivo de este sintetizador es generar cantos realistas para poder entrenar un clasificador. Por esta razón, será necesario agregar algún tipo de variabilidad a los cantos sintetizados. Como describí en la sección 3.1, es suficiente describir la dependencia temporal de la frecuencia fundamental del canto a sintetizar. Una opción es definir punto a punto la modulación (definir manualmente la frecuencia del sonido en cada instante). El problema es que de este forma, toda la variabilidad que podemos agregar consiste en agregar un ruido aleatorio, lo que generaría una serie de cantos que se parecen entre sí, pero no reflejan variabilidad realista.

Sería más interesante en cambio que la variabilidad consistiera en que una sílaba fuera más aguda, o durara más, o que la pendiente de una bajada (como la sílaba repetida del trino del chingolo en la figura 1.1) fuera más pronunciada. Esto requeriría describir las modulaciones

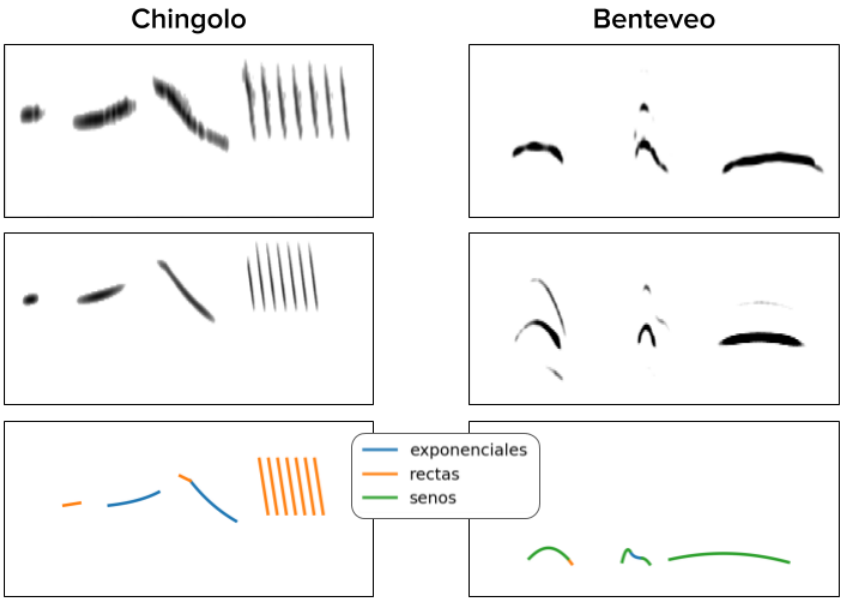

Figura 3.4: *Cantos de chingolo y benteveo utilizados para el entrenamiento de las redes con duracoines de 1.66 y 0.95 segundos, respectivamente. Los sonogramas abarcan un rango de frecuencais de 0kHz a 8kHz. El canto real (superior) fue inspiración del canto sintético (medio). En modulación de la frecuencia fundamental (inferior) aparecen diferenciados en colores los tres tipos de gestos utilizados.*

Modulación	Parametrización	Parámetros
lineal	$\omega(t) = \omega_i + (\omega_f - \omega_i)\tilde{t}$	$t_i, t_f, \omega_i, \omega_f$
exponencial	$\omega(t) = \omega_f + (\omega_i - \omega_f)e^{-\tau\tilde{t}}$	$t_i, t_f, \omega_i, \omega_f, \tau$
sinusoidal	$\omega(t) = m + Asen(\alpha_i + (\alpha_f - \alpha_i)\tilde{t})$	$t_i, t_f, \alpha_i, \alpha_f, m, A$

Tabla 3.2: *Tabla de parametrizaciones y parámetros requeridos para cada una. En todos los casos,* $\tilde{t} = \frac{t-t_i}{t_f-t_i}$. *La parametrización utilizando el seno es la que más trabajo requiere, porque es menos directo partir partir de la frecuencia para obtener los* α *requeridos.*

de forma paramétrica. De hecho, la modulación de la frecuencia fundamental en los cantos reales se puede describir como instrucciones fisiológicas simples que llamamos “gestos”. Para benteveos y chingolos (las especies tratadas en este trabajo), se puede conseguir una descripción relativamente completa de los cantos utilizando modulaciones dadas por senos, funciones lineales y exponenciales. En la tabla 3.2 muestro los parámetros de los que depende cada una de estas modulaciones. Si bien se mostró[38] que en principio estos gestos son suficientes para describir cualquier sílaba del canto de estas especies, para tener un poco más de flexibilidad, decidí tener la libertad de combinar varios gestos en una sílaba. En la figura 3.4 se puede ver la modulación utilizada para una síntesis con los distintos gestos claramente diferenciados.

El último punto que nos queda aclarar antes de sintetizar los datos de entrenamiento es qué canto vamos a sintetizar. Como comenté en la sección 1, el canto del benteveo es extremadamente

estereotipado, y el del chingolo, sólo un poco menos. Por esta razón, espero poder entrenar completamente el clasificador a partir de síntesis inspiradas en un único canto de cada especie. Es interesante pensar también en especies con cantos menos estereotipados, como el zorzal. En este caso, sería necesario estudiar la estadística de los gestos utilizados por el ave en su canto, lo cual queda para trabajo futuro.

Como comenté en la sección 2.2, voy a utilizar representaciones gráficas de los cantos para alimentar el clasificador. Estas representaciones se llaman sonogramas y se construyen calculando la transformada de fourier (la FFT, en particular) del sonido en pequeñas ventanas de tiempo, para obtener el espectro del sonido en ese intervalo. Se desplaza la ventana para barrer todo el canto y se arregla cada espectro en una imagen donde el eje vertical indica frecuencia, el horizontal, tiempo, y la intensidad de cada pixel representa la energía presente en esa frecuencia en un dado instante, usualmente en escala logarítmica. En este trabajo construí los sonogramas con ventanas gaussianas de 0.012 segundos una desviación estándar (σ) de 0.003 segundos y un solapamiento de 0.01 segundos, y los recorté para que no excedan el rango de 8 kHz. Luego aplico a los sonogramas un postprocesado para limpiar el sonograma de ruido de la siguiente forma

- aplico un filtro gaussiano pequeño con $\sigma = 1$ píxel
- descarto los píxeles con valores muy pequeños, para minimizar el ruido y agrandar el rango dinámico al doble
- utilizo el método de Otsu[39] para calcular un umbral que separe entre ruido y señal y utilizo este umbral para descartar nuevamente el ruido
- aumento el contraste levemente

Este conjunto de filtros y reducciones de ruido fueron diseñados para poder utilizar cantos reales que de otra forma habrían resultado demasiado ruidosos y deberían haber sido descartado. Se puede ver en la figura 3.5 el sonograma de un canto (real) crudo, con el ruido ambiente que implica su grabación en campo, y el mismo canto filtrado. Se ve que además de eliminarse el ruido, el sonograma ahora muestra únicamente la frecuencia más prominente en el canto original. Al hacer esto debemos tener en cuenta que el clasificador que generemos estará poniendo a prueba la hipótesis de que este puede clasificar basándose únicamente en la modulación general del canto, y no necesita el contenido espectral para hacerlo.

Para sintetizar los datos de entrenamiento tomo uno de los cantos de la base de datos de Xenocanto para cada especie, genero su sonograma y diagramo manualmente los gestos que pueden costituir cada sílaba. Luego utilizo la plataforma online desmos (una calculadora gráfica

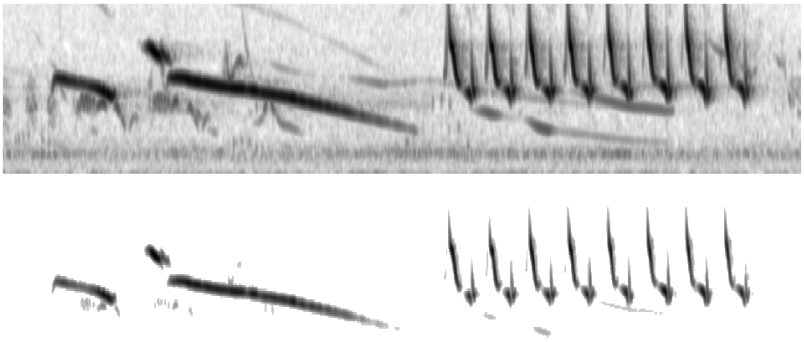

Figura 3.5: *Canto original de un chingolo del conjunto de prueba (superior) y canto filtrado para eliminar ruido utilizando los pasos descritos (inferior).*

con capacidades interactivas muy poderosa) para estimar los parámetros que gobiernan los gestos. En particular, los cantos utilizados, junto con su modulación estimada y síntesis aparecen en la figura 3.4. Para generar la variabilidad entre las síntesis altero la duración y separación entre sílabas de forma aleatoria en un rango de 20 % del valor original modifico los parámetros que gobiernan cada gesto en un 10 % del original y agrego un ruido de 10 % a la modulación en sí. Finalmente, integro el modelo con la modulación generada siguiendo los pasos diagramados en 3.1 2500 veces para cada especie, separo 2000 para el entrenamiento, 500 para la validación y 100 para prueba. Recordemos sin embargo que el testeo sobre este último grupo no es más que una formalidad para corroborar el correcto funcionaiento y entrenamiento de la red. La verdadera prueba será clasificar cantos reales, aún sin haber sido expuesto nunca a ninguno.

Capítulo 4

Diseño de CNNs y entrenamiento

Es este capítulo hablo sobre los clasificadores utilizados, los resultados del entrenamiento y su capacidad para clasificar cantos reales.

4.1. Clasificadores

Todas las redes profundas creadas fueron implementadas utilizando la librería Keras para *python3*. Keras expone al usuario una interfaz de alto nivel que funciona independientemente del servicio de fondo utilizado para realizar efectivamente las cuentas necesarias para aprendizaje profundo. En particular, en este trabajo utilizo Tensorflow. Con la interfaz de Keras, el usuario sólo debe preocuparse por definir la arquitectura y el resto de los hiperparámetros de la red. Tanto Keras como Tensorflow son proyectos de código abierto de Google.

Para diseñar el clasificador convolucional a utilizar hay una cantidad de variables que se pueden modificar. La primera es poco ortodoxa, por decirlo de alguna manera. El kernel de las capas convolucionales (2D) suele ser cuadrado, dado que de esta forma no se establece una dirección privilegiada en las características que la capa es capaz de extraer. Los sonogramas, sin embargo poseen intrínsecamente una distinción entre la dirección horizontal y la vertical en la imagen: una representa frecuencais y la otra tiempo. Por esta razón, podría en principio resultar beneficioso utilizar kernels rectangulares, asimétricos.

El resto de las variables serán elegidas con la idea de generar la red más pequeña o simple posible que clasifique bien los cantos. La red será alimentada con versiones de 300×200 píxeles en escala de grises de los sonogramas. Comenzaré utilizando kernels pequeños, de entre tres y siete píxeles. Eventualmente, si es necesario, utilizaré redes con kernels más grandes. De la

misma forma, las redes utilizadas serán relativamente poco profundas: con entre tres y cautro capas convolucionales y una o dos capas densas ocultas, además de la capa de salida.

Con todo esto en mente, entrené diez redes distintas desde cero utilizando grupos de 20 sonogramas para realizar 100 pasos por época durante un total de ocho épocas. El paso de aprendizaje se fijó en 0.001, el valor por defecto para RMSprop, el optimizador utilizado. En el apéndice describo la arquitectura de cinco de ellas, que llamaré *asimétrica*, *pequeña*, *pequeña densa*, *grande* y *grande profunda*, cada una distinta en algún aspecto. De las redes faltantes, dos tienen bloques convolucionales con más capas, y el resto se asemeja a redes que sí fueron incluidas en la lista; ninguna de ellas agrega información valiosa.

Existe una técnica que hasta el momento no comenté que es la de utilizar redes neuronales preentrenadas. Estas son clasificadores entrenados sobre enormes conjuntos de datos en sistemas de alta capacidad de procesado y muchas están disponibles al público con todos sus parámetros ya entrenados. Si recordamos lo comentado en la sección 2.2, las capas convolucionales de un clasificador realizan lo que se llama la extracción de características, y las capas densas constituyen el clasificador en sí: relacionan las característias con las clases.

Una posibilidad, entonces, es tomar de una red preentrenada sólo el extractor de características y entrenar sobre este un clasificador propio, que aprenda a conectar las características con las salidas pertinentes al problema en particular. De ser necesario, también pueden ajustarse los pesos de las últimas capas convolucionales. En este trabajo usé la red VGG16[40]. Es un clasificador profundo entrenado sobre catorce millones de imágenes pertenecientes a mil categorías. Sobre el extractor de características de esta red entrené dos clasificadores densos distintos, uno de ellos en el apéndice.

4.2. Resultados

Para presentar los resultados del entrenamiento en sí, es tradicional utilizar un gráfico de exactitud y pérdida en los grupos de entrenamiento y de validación a lo largo de las épocas. En estos se observa cómo la pérdida disminuye y la exactitud aumenta, y es relativamente fácil identificar a partir de qué época el sistema comienza a sorbreajustar, porque se observa que la precisión de validación comienza a disminuir.

En el entrenamiento realizado en estas redes, se observó que ya en la segunda época la exactitud de entrenamiento y validación converge a 100 % y la pérdida mejora marginalmente en las épocas subsiguientes (esta es parte de la razón por la que sólo entrené por ocho epócas).

Sobre los datos de prueba la exactitud también fue perfecta. Esto lo que sugiere es que los cantos sintetizados presentan poca variabilidad intraespecífica y/o mucha variabilidad interespecífica. Es importante también comentar que todas las redes clasificaron correctamente los dos cantos que fueron inspiración de las síntesis, aún no habiendo sido expuestas a ningún canto real.

En principio, esto puede sugerir que las síntesis realizadas no son suficientemente buenas. La prueba de fuego, sin embargo, consiste en pedir al sistema que clasifique cantos reales a los que nunca fue expuesto y cuyos equivalentes sintetizados tampoco conoció. Tomé para ello 50 cantos de benteveo y 60 cantos de chingolo, los procesé con los mismos filtros que a los cantos utilizados para la síntesis y los pasé por las redes.

Aquí es donde los resultados entre distintas redes difieren, y para evaluarlos voy a introducir algunas métricas más además de meramente la exactitud. Dadas dos categorías, se puede construir lo que se llama matriz de confusión:

	A_{verd}	$\bar{A}_{verd}$
A_{pred}	VP	FP
$\bar{A}_{pred}$	FN	VN

donde VP significa verdaderos positivos y está dado por la cantidad de aciertos en la categoría A, FP son falsos positivos que corresponden a predecir erróneamente que un dato de la categoría $\bar{A}$ correspondía en realidad a A, y VN y FN son verdaderos y falsos negativos, dados por predecir correcta o incorrectamente que un dato pertenece o no a la categoría $\bar{A}$, respectivamente.

Para aclarar esto, tomemos como ejemplo los resultados de la red más pequeña utilizada. Esta acertó correctamente la clasificación para 49 cantos de benteveo y 58 cantos de chingolo, y clasificó erróneamente un benteveo y dos chingolos. La matriz de confusión resultante es entonces

	B_{verd}	C_{verd}
B_{pred}	49	1
C_{pred}	2	58

donde B corresponde a benteveos y C a chingolos. Este tipo de matrices puede generalizarse a más categorías, por supuesto. Lo interesante de esto es que a partir de la matriz puede calcularse una cantidad de métricas. Yo voy a utilizar cuatro en total: exactitud, error, precisión y exhaustividad. La primera ya la conocemos, es simplemente la cantidad total de aciertos por sobre el total de datos, el error es uno menos la exactitud. La precisión es una métrica que se calcula por categoría y representa cuántas de las veces que predijo que un dato corresponde a una dada

categoría, la predicción es acertada, penaliza por falsos positivos. Finalmente, la exhaustividad indica cuántos de los datos de una dada categoría fueron identificados correctamente, penaliza por falsos negativos. Numéricamente, se calculan de la siguiente forma:

$$\begin{aligned} exac &= \frac{VP + VN}{total} \\ error &= 1 - exac \\ prec_A &= \frac{VP_A}{VP_A + FP_A} \\ exha_A &= \frac{VP_A}{total_A} \end{aligned} \tag{4.1}$$

donde A es alguna clase arbitraria. Siguiendo el ejemplo de antes de la red más pequeña utilizada, tenemos exactitud 0.97, error 0.03, precisión de benteveo y chingolo 0.96 y 0.98, respecivamente, y exhaustividad para cada clase 0.98 y 0.97. En la figura 4.1 ilustro los resultados para todas las redes utilizadas. En esta también muestro la cantidad de parámetros que tiene cada red, que es una forma muy básica e inocente de cuantificar el tamaño de una red.

Lo primero que se observa es que la red más pequeña es la que mejor se desempeña en todas las métricas, en algunos casos igualada por otras arquitecturas. Es interesante notar que la red asimétrica es la que peor exactitud tiene, dado evidentemente por una gran cantidad de chingolos identificados erróneamente como benteveos. De hecho, la tendencia general de los errores aparenta ser esa: chigolos identificados como benteveos. Esto en principio parece ser el efecto de sobreajuste: ante una mayor libertad de parámetros estos se invierten en memorizar conexiones entre los datos y las etiquetas de entrenamiento, perdiendo así capacidad de generalización. Sin embargo, sobreajuste y generalización no son términos adecuados en este contexto, dado que estrictamente, estos aplican sobre los datos de validación y prueba, que en este caso son cantos sintetizados. Aún así, la explicación más plausible es que la red haya aprendido asociaciones específicas a las síntesis que no generalizan bien a los cantos reales.

Se ve también que la red pequeña se desempeña mejor con un clasificador denso pequeño, mientras que la red más grande necesita un clasificador con más capas y parámetros libres. Esto es razonable, considerando que un extractor de características más grande generará más datos que requerirán más neuronas en las capas densas para asociar con las clases. Sería esperable también que un clasificador denso aún más grande eventualmente se desempeñe peor, como sucede con la red pequeña. Finalmente, el clasificador entrenado sobre el bloque convolucional

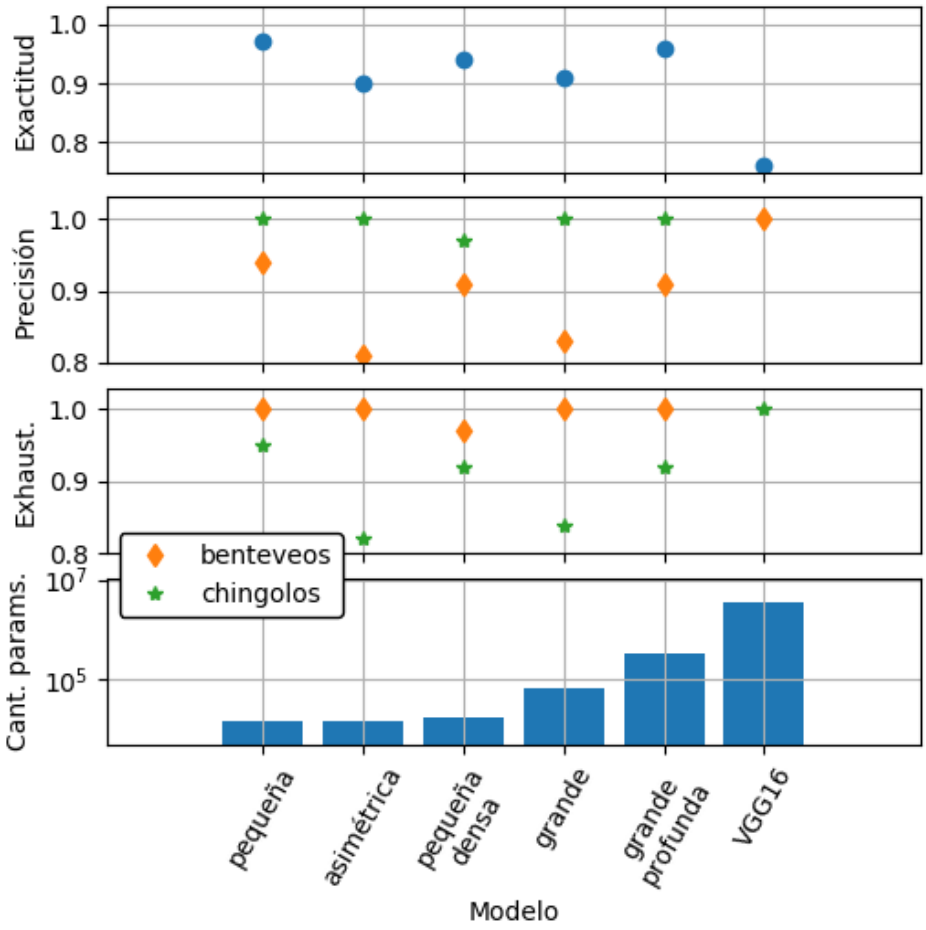

Figura 4.1: *Valores de las métricas obtenidas para algunas de las redes utilizadas (tres paneles superiores) y una estimación del tamaño de cada red dada por la cantidad de parámetros (panel inferior). Nótese que los ejes no comienzan en 0. Los valores de VGG16 para precisión de chingolos y exhaustividad de benteveos se van de escala y son 0.7 y 0.48, respectivamente. La cantidad de parámetros indicada para la misma red corresponde sólo al clasificador entrenado por sobre el extractor de características utilizado, que tiene una cantidad de parámetros un orden mayor que el clasificador que se muestra en la figura.*

de VGG16 es la red que peor se desempeña en todas las métricas, en este caso invirtiendo la tendencia e identificando erróneamente benteveos como chingolos, en su mayoría. Esto puede deberse a lo mismo que los casos anteriores: las redes demasiado grandes aprenden relaciones demasiado específicas a los cantos sintetizados y no generalizan bien a cantos reales. En otras palabras, notan diferencias entre los cantos reales y los sintetizados.

Esta es una hipótesis que es relativamente fácil de poner a prueba. Tomo todos los cantos reales y una cantidad similar de cantos sintéticos y los separo en categorías reales y sintéticos, mezclando cantos de benteveo y chingolo en cada una. Separando en grupo de entrenamiento, prueba y validación y entrenando las mismas redes que antes, se obtiene que la red pequeña alcanza una exactitud de 0.97, mientras que las llamadas pequeña densa y grande profunda alcanzan 99 % de exactitud. A su vez, la red grande con un clasificador denso relativamente pequeño, se desempeña similar a la red pequeña. Todo esto respalda la hipótesis de que redes más grandes tiene la capacidad de diferenciar entre cantos sintéticos y cantos reales mejor que redes más pequeñas, aunque sólo marginalmente.

Capítulo 5

Discusión

En esta tesis implementé una nueva técnica de aumentado de datos para entrenar clasificadores de especies de aves según su canto. Tradicionalmente, para hacer esto es necesaria una cantidad relativamente grande de ejemplos de cada categoría manualmente etiquetados, lo cual implica trabajo de campo extensivo y tiempo de postprocesado. Los métodos de aumentado de datos tradicionalmente utilizados en visión artificial no son de gran utilidad en este problema, por lo que existen complejas técnicas de aumentado que implican recortar y recombinar las muestras existentes.

La técnica presentada utiliza un modelo dinámico del aparato fonador de las aves para generar cantos sintéticos, inspirándose en un canto del ave real. Entrenando redes con distintas arquitecturas con estos cantos sintéticos y evaluándolas por su capacidad de clasificar cantos reales, observo que una red muy pequeña para los estándares de visión artificial obtiene exactitud casi perfecta. Redes más grandes también se desempeñan excelentemente, pero lo que es más interesante es el hecho de que se puede solucionar el problema de clasificación con una red muy simple. También se observa que redes más grandes tiene la capacidad de diferenciar entre cantos reales y sintéticos, lo cual puede ser el origen del decrecimiento marginal en exactitud que presentan. Es posible realizar una búsqueda exhaustiva de hiperparámetros de la red, pero la red en sí no es objeto central de este trabajo, sino la técnica de aumentado. Por esto resulta interesante que la técnica haya sido relativamente exitosa con todas las redes utilizadas.

Esta técnica de aumentado es interesante para todos los tipos de tareas descritas en la sección 1.1. Para el problema más simple de identificar presencia o ausencia de aves es trivial sintetizar muestras. El siguiente problema, el de clasificar cantos aislados, fue sobre el que trabajé en esta tesis. Queda por ver en cómo de desempeña este mismo sistema con cantos menos

esteretipados. Dos candidatos ideales son el zorzal colorado (*Turdus rufiventris*) y el hornero común (*Furnarius rufus*), que comparten hábitat con el chingolo y el benteveo. El primero tiene un canto de duración variable compuesto por sílabas tomadas de un repertorio, generando cantos más variados que los utilziados en este trabajo. Para sintetizar este tipo de cantos sería necesario realizar una estadística sobre las sílabas utilizadas y su distribución, y luego generar los perfiles de modulación de frecuencia fundamental utilizando cadenas de Markov u otro proceso estocástico.

Para generar muestras para una tarea donde se debe identificar la especie más prominente en una grabación con otras aves de fondo, la síntesis comienza igual que en esta tesis: se toman cantos individuales de todas las especies a utilizar y se generan las síntesis correspondientes. Luego sigue un trabajo de seleccionar aleatoriamente la especie prominente para cada muestra y rellenar con síntesis de las otras aves, ya sea de la misma o de otra especie. De las misma forma se pueden sintetizar grabaciones de larga duración con muchas especies cantando en simultáneo o en secuencia.

Esta técnica de aumentado también abre las puertas a un tipo de tarea que hasta el momento resultaba altamente poco práctica: la identificación de individuos. Para muchos trabajos de ecología es interesante poder hacer un seguimiento de individuos en particular dentro de una región, por ejemplo para estudiar hábitos. Para entrenar un clasificador automático de forma tradicional es necesario contar con decenas o cientos de cantos del individuo en particular, dependiendo de qué tan estereotipado sea el canto. Utilizando esta técnica de aumentado, en cambio, se puede partir de un único canto y realizar variaciones sobre los parámetros más pequeñas que las que usé en este trabajo. A partir de esto se puede entrenar un clasificador para diferenciar individuos de una misma especie.

Lo interesante de la aplicación de síntesis a todos estos problemas es que los datos de entrenamiento no deben ser clasificados previamente de forma manual, sino que la síntesis provee también las etiquetas para la clasificación. De esta forma, el trabajo manual previo requerido no escala linealmente con la cantidad de datos que se desea tener para el entrenamiento. Antes de involucrarse con este tipo de problemas, sin embargo, el primer paso a seguir es, como dije, ampliar este clasificador simple de cantos aislados con más especies.

Apéndice A

Arquitectura de las redes utilizadas

Pequeña

Tipo de capa	Características	Dimensión de salida
Convolución 2D	4 kernels de 5×5	$148 \times 98 \times 4$
Max Pooling	kernel de 4×4	$37 \times 24 \times 4$
Convolución 2D	4 kernels de 3×3	$35 \times 22 \times 4$
Max Pooling	kernel de 2×2	$17 \times 11 \times 4$
Convolución 2D	8 kernels de 3×3	$15 \times 9 \times 8$
Max Pooling	kernel de 2×2	$7 \times 4 \times 8$
Aplanar		244
Densa	64 neuronas, activación ReLU	64
Densa	Capa de salida	2

Asimetrica

Tipo de capa	Características	Dimensión de salida
Convolución 2D	4 kernels de 5×3	$148 \times 99 \times 4$
Max Pooling	kernel de 4×4	$37 \times 24 \times 4$
Convolución 2D	4 kernels de 5×3	$33 \times 22 \times 4$
Max Pooling	kernel de 2×2	$16 \times 11 \times 4$
Convolución 2D	8 kernels de 3×3	$14 \times 9 \times 8$
Max Pooling	kernel de 2×2	$7 \times 4 \times 8$
Aplanar		244
Densa	64 neuronas, activación ReLU	64
Densa	Capa de salida	2

Pequeña densa

Tipo de capa	Características	Dimensión de salida
Convolución 2D	4 kernels de 5×5	$148 \times 98 \times 4$
Max Pooling	kernel de 4×4	$37 \times 24 \times 4$
Convolución 2D	4 kernels de 3×3	$35 \times 22 \times 4$
Max Pooling	kernel de 2×2	$17 \times 11 \times 4$
Convolución 2D	8 kernels de 3×3	$15 \times 9 \times 8$
Max Pooling	kernel de 2×2	$7 \times 4 \times 8$
Aplanar		244
Densa	64 neuronas, activación ReLU	64
Densa	32 neuronas, activación ReLU	32
Densa	Capa de salida	2

Grande

Tipo de capa	Características	Dimensión de salida
Convolución 2D	8 kernels de 7×7	$294 \times 194 \times 8$
Max Pooling	kernel de 4×4	$73 \times 48 \times 8$
Convolución 2D	16 kernels de 5×5	$69 \times 44 \times 16$
Max Pooling	kernel de 4×4	$17 \times 11 \times 16$
Convolución 2D	32 kernels de 3×3	$15 \times 9 \times 32$
Max Pooling	kernel de 2×2	$7 \times 4 \times 32$
Aplanar		896
Dropout	factor = 0.5	896
Densa	64 neuronas, activación ReLU	64
Densa	Capa de salida	2

Grande Profunda

Tipo de capa	Características	Dimensión de salida
Convolución 2D	4 kernels de 5×5	$294 \times 194 \times 4$
Max Pooling	kernel de 2×2	$148 \times 98 \times 4$
Convolución 2D	4 kernels de 5×5	$144 \times 94 \times 4$
Max Pooling	kernel de 2×2	$72 \times 47 \times 4$
Convolución 2D	8 kernels de 3×3	$70 \times 45 \times 8$
Max Pooling	kernel de 2×2	$35 \times 22 \times 8$
Convolución 2D	16 kernels de 3×3	$33 \times 20 \times 16$
Max Pooling	kernel de 2×2	$16 \times 10 \times 16$
Aplanar		2560
Densa	128 neuronas, activación ReLU	128
Densa	Capa de salida	2

VGG16 (clasificador denso)

Tipo de capa	Características	Dimensión de salida
Densa	64 neuronas, activación ReLU	64
Dropout	factor = 0.5	64
Densa	64 neuronas, activación ReLU	64
Dropout	factor = 0.5	64
Densa	Capa de salida	2

Bibliografía

[1] *Organización Aves Argentinas, y COA*. URL: https://www.avesargentinas.org.ar/quienes-somos-0 (visitado 05-01-2020).

[2] Alison Johnston y col. "Observed and predicted effects of climate change on species abundance in protected areas". En: *Nature Climate Change* 3.12 (nov. de 2013), págs. 1055-1061. DOI: 10.1038/nclimate2035.

[3] *Competencias Bird Audio Detection*. 2020-01-12. URL: http://dcase.community.

[4] *Competencia anual de clasificación de Kaggle*. URL: https://www.kaggle.com/c/mlsp-2013-birds (visitado 12-01-2020).

[5] *Competencia anual de Neural Information Processing Scaled for Bioacoustics: NIPS4B*. URL: https://figshare.com/articles/Transcriptions_of_NIPS4B_2013_Bird_Challenge_Training_Dataset/6798548 (visitado 12-01-2020).

[6] *Competencia anual BirdCLEF*. URL: https://www.imageclef.org/BirdCLEF2019 (visitado 12-01-2020).

[7] *Versión extendida de la competencia Bird Audio Detection*. URL: http://machine-listening.eecs.qmul.ac.uk/bird-audio-detection-challenge/ (visitado 12-01-2020).

[8] Dan Stowell y col. "Automatic acoustic detection of birds through deep learning: The first Bird Audio Detection challenge". En: *Methods in Ecology and Evolution* 10.3 (nov. de 2018). Ed. por David Orme, págs. 368-380. DOI: 10.1111/2041-210x.13103.

[9] Louis Ranjard y Howard Ross. "Unsupervised bird song syllable classification using evolving neural networks". En: *The Journal of the Acoustical Society of America* 123 (jul. de 2008), págs. 4358-68. DOI: 10.1121/1.2903861.

[10] Botond Fazeka y col. “A Multi-modal Deep Neural Network approach to Bird-song identification”. En: (2018). arXiv: 1811.04448 [cs.SD].

[11] *Xenocanto*. URL: https://www.xeno-canto.org (visitado 11-03-2020).

[12] David Silver y col. “Mastering the game of Go without human knowledge”. En: *Nature* 550.7676 (oct. de 2017), págs. 354-359. DOI: 10.1038/nature24270.

[13] Peng Peng y col. “Multiagent Bidirectionally-Coordinated Nets: Emergence of Human-level Coordination in Learning to Play StarCraft Combat Games”. En: (2017). arXiv: 1703.10069 [cs.AI].

[14] Julian Schrittwieser y col. “Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model”. En: (nov. de 2019).

[15] Oleksii Kharkovyna. “Artificial Intelligence & Deep Learning for Medical Diagnosis”. En: *Towards Data Science* (13 de nov. de 2019).

[16] Matthew B. Hoy. “Alexa, Siri, Cortana, and More: An Introduction to Voice Assistants”. En: *Medical Reference Services Quarterly* 37.1 (ene. de 2018), págs. 81-88. DOI: 10.1080/02763869.2018.1404391.

[17] Oscar Schwartz. “You thought fake news was bad? Deep fakes are where truth goes to die”. En: *The Guardian* (12 de nov. de 2018).

[18] Nicolas Gallardo y col. “Autonomous decision making for a driver-less car”. En: *2017 12th System of Systems Engineering Conference (SoSE)*. IEEE, jun. de 2017. DOI: 10.1109/sysose.2017.7994953.

[19] Robert M. Bell y Yehuda Koren. “Lessons from the Netflix prize challenge”. En: *ACM SIGKDD Explorations Newsletter* 9.2 (dic. de 2007), pág. 75. DOI: 10.1145/1345448.1345465.

[20] Paul Covington, Jay Adams y Emre Sargin. “Deep Neural Networks for YouTube Recommendations”. En: *Proceedings of the 10th ACM Conference on Recommender Systems - RecSys ’16*. ACM Press, 2016. DOI: 10.1145/2959100.2959190.

[21] Marcel van Gerven y Sander Bohte. “Editorial: Artificial Neural Networks as Models of Neural Information Processing”. En: *Frontiers in Computational Neuroscience* 11 (dic. de 2017). DOI: 10.3389/fncom.2017.00114.

[22] Yann LeCun, Yoshua Bengio y Geoffrey Hinton. "Deep learning". En: *Nature* 521.7553 (mayo de 2015), págs. 436-444. DOI: 10.1038/nature14539.

[23] John Martinsson. "Bird Species Identification using Convolutional Neural Networks". Tesis de mtría. Department of Computer Science y Engineering, UNIVERSITY OF GOTHENBURG, 2017.

[24] Maxwell Stinchcombe y Halbert White. "Universal approximation using feedforward networks with non-sigmoid hidden layer activation functions". En: *IJCNN International Joint Conference on Neural Networks*. 1989, págs. 613-617.

[25] Francois Chollet. *Deep Learning with Python*. 1st. USA: Manning Publications Co., 2017. ISBN: 1617294438.

[26] Hello World HD. *Understanding Convolutional Neural Networks: Making a Handwritten Digit Calculator | Keras #5*. 2019. URL: https://youtu.be/eyKwPyOqMg4?t=302 (visitado 05-02-2020).

[27] Dominik Scherer, Andreas Müller y Sven Behnke. "Evaluation of pooling operations in convolutional architectures for object recognition". En: ene. de 2010, págs. 92-101. DOI: 10.1007/978-3-642-15825-4_10.

[28] Geoffrey E. Hinton y col. *Improving neural networks by preventing co-adaptation of feature detectors*. 2012. arXiv: 1207.0580 [cs.NE].

[29] David E. Rumelhart, Geoffrey E. Hinton y Ronald J. Williams. "Learning representations by back-propagating errors". En: *Nature* 323.6088 (oct. de 1986), págs. 533-536. DOI: 10.1038/323533a0.

[30] Gabriel B. Mindlin. "Nonlinear dynamics in the study of birdsong". En: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 27.9 (sep. de 2017), pág. 092101. DOI: 10.1063/1.4986932.

[31] Clive Catchpole y 1942- Slater P. J. B. (Peter James Bramwell). *Bird song : biological themes and variations*. English. Includes bibliographical references (p. [221]-240) and index. Cambridge [England] : New York, NY, USA : Cambridge University Press, 1995. ISBN: 0521417996 (hardback).

[32] Roderick A. Suthers. "Contributions to birdsong from the left and right sides of the intact syrinx". En: *Nature* 347.6292 (oct. de 1990), págs. 473-477. DOI: 10.1038/347473a0.

[33] Rodrigo Alonso, Franz Goller y Gabriel B. Mindlin. “Motor control of sound frequency in birdsong involves the interaction between air sac pressure and labial tension”. En: *Physical Review E* 89.3 (mar. de 2014). DOI: 10.1103/physreve.89.032706.

[34] Rodrigo Laje, Timothy J. Gardner y Gabriel B. Mindlin. “Neuromuscular control of vocalizations in birdsong: A model”. En: *Physical Review E* 65.5 (mayo de 2002). DOI: 10.1103/physreve.65.051921.

[35] Ana Amador y col. “Elemental gesture dynamics are encoded by song premotor cortical neurons”. En: *Nature* 495.7439 (feb. de 2013), págs. 59-64. DOI: 10.1038/nature11967.

[36] Yonatan Sanz Perl y col. “Reconstruction of physiological instructions from Zebra finch song”. En: *Physical Review E* 84.5 (nov. de 2011). DOI: 10.1103/physreve.84.051909.

[37] P.L. Tubaro y G.B. Mindlin. “A dynamical system as the source of augmentation in a deep learning problem”. En: *Chaos, Solitons & Fractals: X* 2 (jun. de 2019), pág. 100012. DOI: 10.1016/j.csfx.2019.100012.

[38] Tim Gardner y col. “Simple Motor Gestures for Birdsongs”. En: *Physical Review Letters* 87.20 (oct. de 2001). DOI: 10.1103/physrevlett.87.208101.

[39] Nobuyuki Otsu. “A Threshold Selection Method from Gray-Level Histograms”. En: *IEEE Transactions on Systems, Man, and Cybernetics* 9.1 (ene. de 1979), págs. 62-66. DOI: 10.1109/tsmc.1979.4310076.

[40] Karen Simonyan y Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2014. eprint: arXiv:1409.1556.