Análisis de sonido

Sebastian Pineda, Daniel Duque

***Abstract*— The researchers attend to the problem of noise in sound recordings. A self encoding from the spectrogram using different convolutional, transformers and multilayer neural networks is the approach to reduce the noise from the recordings. In order to simulate a noise environment an addition to the wave files of other sound inputs was given. Our results show a reduction in the loss functions, but also suggest that bigger architecture and stronger hardware might be needed to fulfill a correct cleaning of the sound.**

I. Introducción

El sonido es esencial para muchos procesos dentro de la humanidad. Nos permite comunicarnos a través del lenguaje, expresarnos a través de la música y entender de alguna manera nuestro entorno. En este documento, estudiaremos cómo desarrollar algunas herramientas para detectar diferentes fuentes del sonido, y reducir el ruido que se presenta en ellos.

Para esto utilizaremos diferentes metodologías para analizar series de sonido, por un lado, utilizaremos redes convolucionales sobre la representación del espectrograma del sonido, por otro lado, utilizaremos redes recurrentes sobre la "serie de tiempo" de una grabación, al igual que una estructura de atención inspirada en transformers.

En todos estos casos utilizaremos diferentes aproximaciones para eliminar el sonido de las diferentes grabaciones.

II. Justificación

El problema a analizar busca atender varios problemas, para esto esperamos poder reducir el ruido ambiental de distintas grabaciones, que puede ser útil por varias razones.

En caso de grabaciones musicales hay muchos artistas que no tienen los medios para aislar sus lugares de grabación, por lo que esta herramienta podría ser útil para ellos al no depender del silencio exterior. Dependiendo lo preciso del modelo podría ser incluso capaz de atender eventos en vivo donde el ruido es general.

La eliminación de ruido diferente a voces por ejemplo, podría ser útil para eliminar ruido de llamadas o mensajes de voz que pueden interrumpir y perjudicar la transmisión de mensajes y la comunicación entre personas.

Otra posible utilidad es en dispositivos diseñados para aumentar el sonido en personas con discapacidad auditiva, donde podrían mejorar la calidad de la recepción al eliminar ruido blanco.

Por último esta herramienta, de ser fructífera, podría servir de pre procesamiento para otros modelos, tanto de clasificación de sonido como de modelos generativos inicializados en grabaciones ruidosas.

III. Estado del arte

El análisis de sonido no es nuevo. En [1] los autores clasifican el ambiente sonoro con diferentes etiquetas, como lo son un lugar con tráfico o una cafetería. Los modelos que utilizan son gaussian mixture models (GMM) Deep multilayer perceptron, redes recurrentes y convolucionales. Lo que nos da una buena idea de algunos modelos utilizados en este periodo.,

Además, en [2] encontramos que el análisis sonoro a través de autoencoders y transformaciones de fourier en el sonido de balineras y rodamientos pueden encontrar fallas dentro de procesos industriales. Estas estrategías nos pueden ayudar en el pre procesamiento y modelado del problema.

Otras técnicas descritas en [3] muestran estrategias basadas en el análisis del espectrograma, para ver los sentimientos de una persona que habla en una grabación usando redes recurrentes y convolucionales.

Si nos centramos en los estudios acerca de música específicamente nos encontramos con [4] que utilizan redes recurrentes con boltzmann machines para generación de música.

Por [5] vemos que hay varios modelos para analizar los aspectos relacionados con el sonido, incluyendo los que vimos anteriormente, pero además viendo (GANS) modelos de atención en redes recurrentes, entre otras.

IV. Análisis de las bases de datos

Para este ejercicio analizaremos datos provenientes de MusicNet de kaggle que contiene un conjunto de canciones clásicas en formato wav junto con sus etiquetas. El formato wav es básicamente una lista de la frecuencia que debería tener un altavoz en cada instante, por lo que básicamente es una serie de tiempo. La base cuenta con 320 canciones en un conjunto subdividido de train y 10 de test.

El manejo de estos datos se puede trasladar directamente a numpy que genera genera una cinta manipulable de la información que utilizaremos directamente.

Esta información también se puede manejar a manera de espectrograma, que pasa los datos a una estructura de dos dimensiones, y que facilita el desarrollo con metodologías tradicionales de análisis de imagen.

V. Estrategias

Vamos a utilizar varias estrategias para aproximarnos a este problema. Desde el manejo de la información utilizaremos estructuras encoding-decoding ya que la entrada y la salida son muy similares en todos los casos. Además la información de toda la entrada puede ser útil para predecir la salida.

El manejo del encoding y la conversión a una representación densa de los datos puede ayudar a eliminar la información que no es relevante, en este caso todo lo que sea ruido sonoro para los efectos de este ejercicio.

Una estructura que manejaremos será el desarrollo de un autoencoder, y mirar si el por si mismo es capaz de reconocer sólo las estructuras del audio. Si bien es posible que no sea tan eficiente puede mostrarnos algunas estructuras interesantes de los datos.

Otra estratégia será la manipulación de los datos originales con ruido que agregaremos de otras fuentes (https://www.tensorflow.org/tutorials/audio/simple_audio) e intentar predecir desde este input sucio, los archivos de audio limpios originales.

Para agilizar los procesos y aprovechar información ya entrenada utilizaremos algunas arquitecturas con transfer learning.

VI. Modelos

Vamos a utilizar una serie de modelos para esta ejecución, las arquitecturas que consideramos necesitan poder mantener una relación entre el pasado de la serie y su valor actual. Otra medida es que el producto es un problema continuo por lo que la salida debe ser continua.

En esta medida utilizaremos redes convolucionales, redes recurrentes y redes con mecanismos de atención inspirados en transformers. También como línea de base utilizamos una arquitectura de redes con capas densas sobre la serie de los datos como input, si bien estas estructuras van a tener muchos parámetros pueden manifestar la relación temporal del sonido al interactuar entre ellos..

a. Capas densas

Realizamos siete modelos autoencoders con capas densas normales:

El autoencoder naive consiste del input proveniente del espectrograma de la grabación, dos capas densas con dropout intermedio de 64 neuronas, una capa densa de la longitud de la serie y un reshape para convertirlo a las dimensiones del espectrograma.

El autoencoder dnn 1 consiste del input proveniente del espectrograma de la grabación, tres capas densas con dropout intermedio de (3000,1500,3000), una capa densa de la longitud de la serie y un reshape para convertirlo a las dimensiones del espectrograma.

El autoencoder dnn 2 consiste del input proveniente del espectrograma de la grabación, 8 capas densas de (1024,256,128,64,128,256,320), una capa densa de la longitud de la serie y un reshape para convertirlo a las dimensiones del espectrograma.

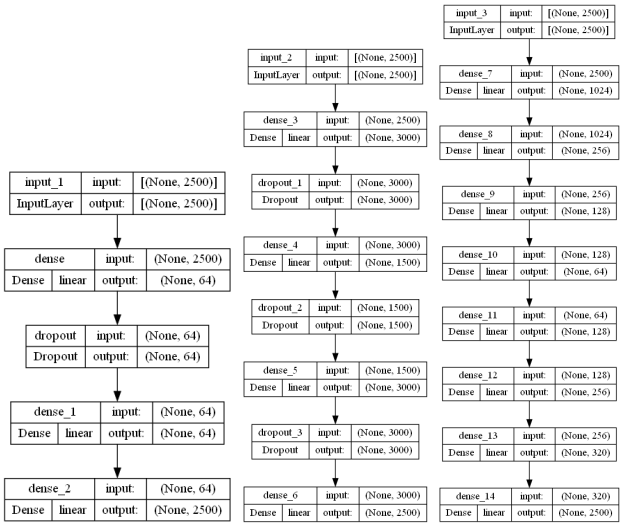


Fig 1: Arquitecturas densas naive, dnn1, dnn2

El autoencoder dnn 3 consiste del input proveniente del espectrograma de la grabación, 8 capas densas de (1024,256,128,64,128,256,320), con dropout antes y después de la capa 64 para conseguir representaciones densas más consistentes. Las anteriores capas con función de activación relu, una capa densa de la longitud de la serie y un reshape para convertirlo a las dimensiones del espectrograma.

El autoencoder dnn 4 consiste del input proveniente del espectrograma de la grabación, 3 capas densas de (3000,1500,3000), con dropout entre cada capa para conseguir representaciones densas más consistentes. Las anteriores capas con función de activación relu, una capa densa de la longitud de la serie y un reshape para convertirlo a las dimensiones del espectrograma.

El autoencoder dnn3_stacked2 y dnn4_stacked2 son representaciones de las correspondientes arquitecturas dnn3 y dnn 4 pero usando stacking del mismo autoencoder, esto para mostrar una estructura más compleja pero con el mismo estilo de capas que las anteriores.

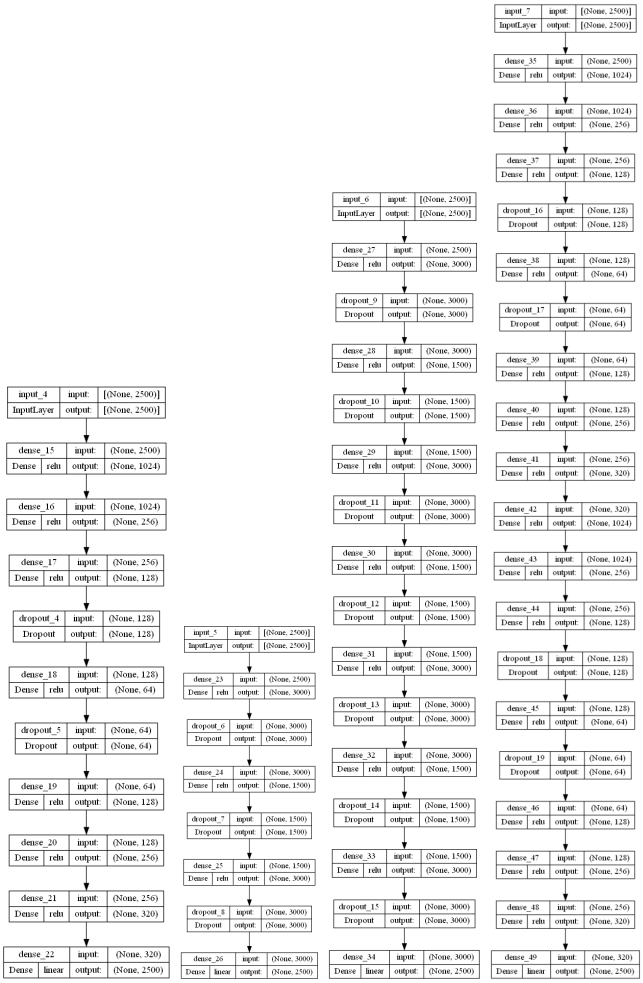


Fig 2: dnn3, dnn4, dnn4_stacked2, dnn3_stacked2

La lógica de los experimentos en esta etapa era verificar que efectos tendrían arquitecturas con más capas, y más neuronas. En este caso de procesos autoencoders un cuello de botella más pequeño puede ayudar a representar de forma más densa y por lo tanto, menos propensa a tomar valores como ruido o cosas que no sean relevantes en el proceso de entrenamiento. El trade off es que también perderá más información de la entrada y podrá ser menos sensible a omitir componentes relevantes.

b. Convolucionales

Probamos dos arquitecturas propias de redes convolucionales y 4 aprovechando transfer learning:

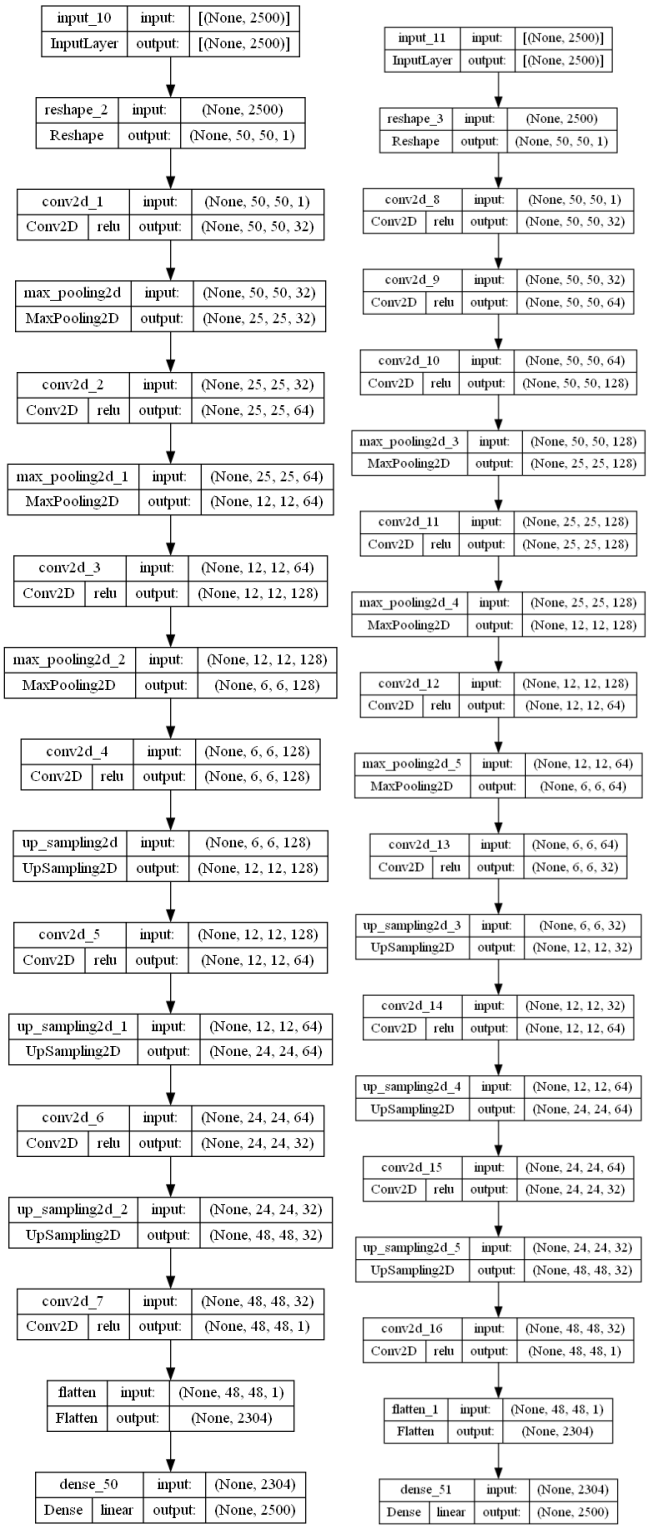


Fig 3: cnn1, cnn2

La convolucional cnn1 utiliza 7 capas convolucionales. La primera de 32 filtros y un kernel de tamaño 1,1. las siguientes dos capas de 64 y 128 filtros con capas max pooling con 2 valores de stride. Las siguientes capas de 128, 64 y 32 filtros tienen capas de Upsampling intermedias para su expansión. Todas las capas tienen padding para completar las dimensiones de la imagen y activación relu.

La convolucional cnn2 es idéntica a la cnn1 en sus primeras 7 capas (excepto que todas tienen máx pooling intermedio) y continua con 3 capas de 64 y 32 filtros con upsampling intermedios. Esperamos ver cómo se comporta una capa convolucional más compleja.

Las redes de transfer learning se eligieron con la lógica de que la representación gráfica de una canción podría tener estructuras similares a aquellas reconocidas en algunos de estos modelos. Estas estructuras básicas podrían, con entrenamiento, reconocer la información útil y desechar el ruido de grabaciones de sonido. Esta hipótesis la probaremos en entrenamiento

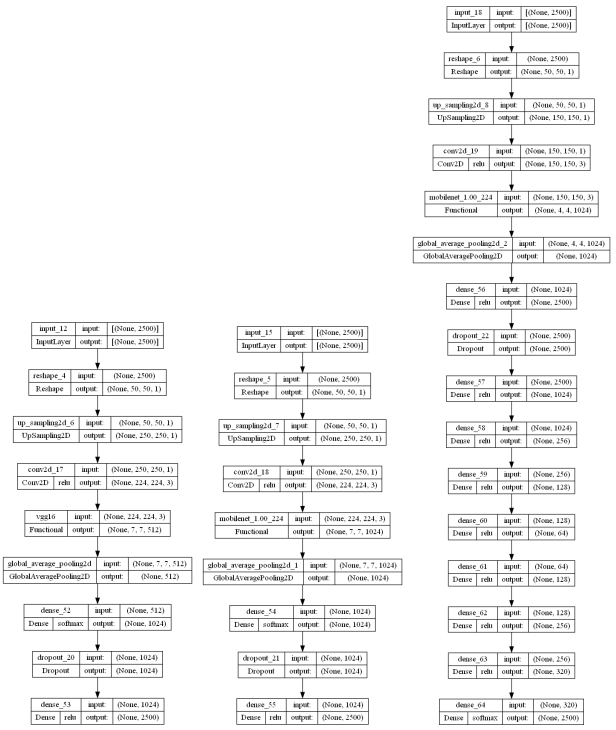


Fig 4: VGG16, MobileNet, MobileNet2

La primera red pre entrenada que usaremos es VGG16 [6] con pesos del entrenamiento en imagenet e input de 224,224 y 3. Para esto utilizamos una primera capa a convolucional que recibe esta dimensión y escala la serie por medio de upsampling para que sea compatible con la entrada. luego completamos con una capa de average pooling, una capa densa de 1024 neuronas activada con softmax y dropout y una capa densa con el número de neuronas de salida

La segunda es MobileNet [7] con pesos del entrenamiento en imagenet e input de 224,224 y 3. Para esto utilizamos una primera capa a convolucional que recibe esta dimensión y escala la serie por medio de upsampling para que sea compatible con la entrada. luego completamos con una capa de average pooling, una capa densa de 1024 neuronas activada con softmax y dropout y una capa densa con el número de neuronas de salida

La tercera es MobileNet2 con pesos del entrenamiento en imagenet e input de 150,150 y 3. Para esto utilizamos una primera capa a convolucional que recibe esta dimensión y escala la serie por medio de upsampling para que sea compatible con la entrada. Luego completamos con una capa de average pooling, una capa densa de 2500 neuronas activada con softmax y dropout. Después viene la construcción de un decoder de (1024,256,128,64,128,256,329) de capas densas con activación relu y una capa densa con las salidas con activación softmax.

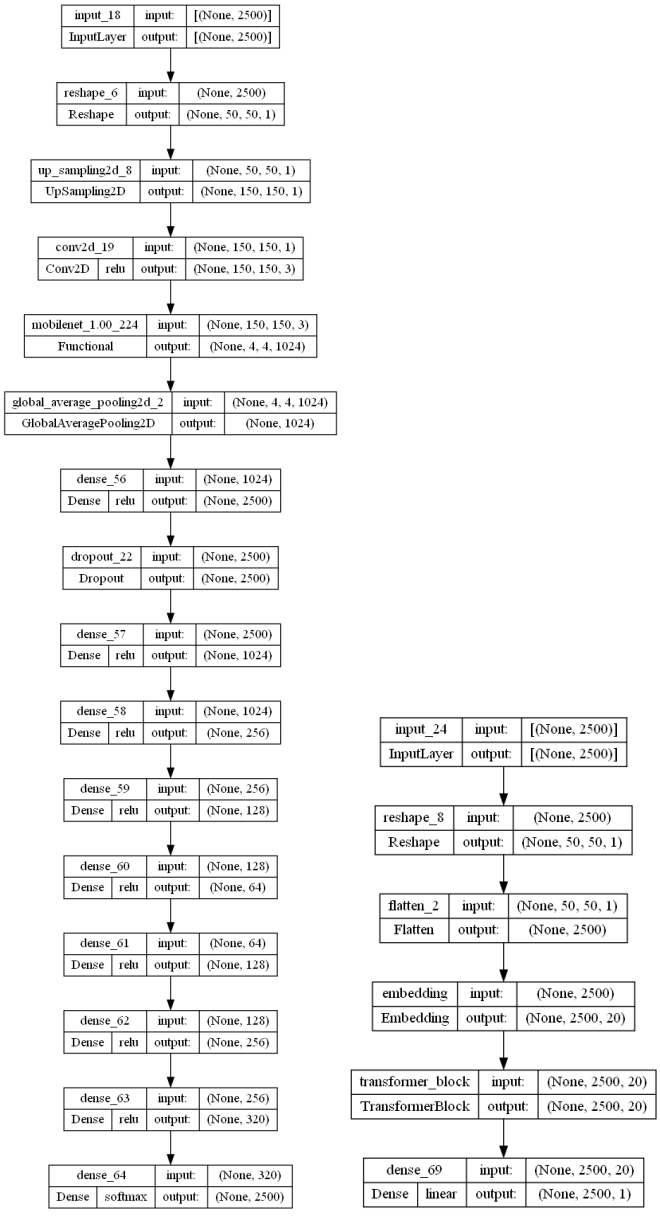


Fig 5: MobileNet 3 y modelo de atención

c. Atencion

Para concentrar parte de la información y centrarnos en las estructuras que más responden a explicar el sonido según sus antecedentes utilizamos un modelo centrado en una capa de atención con múltiples cabezales [9].

El modelo toma el espectrograma de cada canción como input, lo linealiza para tratarlo como una secuencia y lo pasa por un embedding posicional que pasa a una representación en 20 dimensiones. Posteriormente pasa por tres cabezas de atención que seleccionan con mayores pesos las estructuras más relevantes en la predicción de los datos según el contexto. Continúa pasando por dos capas densas de 10 y 20 neuronas. termina con una regularización por dropout y el input de los resultados.

Si bien tiene múltiples características de un transformer, no contiene el paso de introducir el output con un masking para el valor a predecir, por lo que no nos atrevemos a afirmar que sea un transformer.

VII. Entrenamiento

El entrenamiento en general tuvo algunas características generales en todos los modelos, una característica importante es el optimizador Adamax que utiliza un momento infinito en lugar del segundo momento. En la práctica esto permite entrenar de forma relativamente robusta a procesos ruidosos [8].

Para las arquitecturas que se hicieron desde cero, se entrenaron durante 100 épocas, para eliminar este factor de los factores experimentales y contar con el efecto de un entrenamiento largo. Sin embargo para las arquitecturas en las que se utilizó deep learning solo se entrenaron por 20 épocas debido a la capacidad de cómputo disponible, lo que resultaba en tiempos elevados limitando la cantidad de experimentos que podíamos realizar.

VIII. Resultados

Se utilizó un protocolo experimental sencillo donde los factores controlables fueron las arquitecturas y el nivel de dropout el cual variaba entre 0.1 y 0.5.

Para cada experimento se hicieron 3 corridas, con el fin de mitigar un poco el elemento de aleatoriedad dentro del entrenamiento.

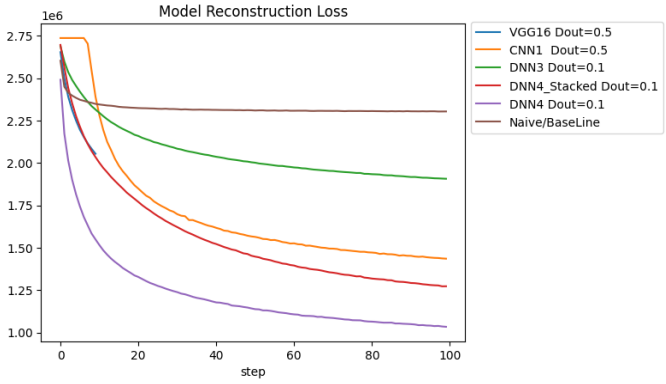
Arquitectura	Dropout	Min Loss	Max Loss	Avg Loss
dnn4	0.1	1033604	1088231.5	1056105.146
dnn4_stacked2	0.1	1272539.125	1272539.125	1272539.125
cnn1	0.5	1435283.125	1447092.125	1441187.625
dnn3	0.1	1898245.875	1906601.375	1902048.75
VGG16	0.5	2053981.625	2736507	2600000.175
dnn1	0.1	2172982.5	2180775.5	2176879
dnn3_stacked2	0.1	2200215.25	2200215.25	2200215.25
dnn2	0.5	2264374.5	2266925.25	2265649.875
dnn2	0.1	2266787	2267070	2266928.5
naive	0.1	2303524.75	2304272.5	2303898.625
dnn3	0.5	2431365.5	2442846.5	2436098.667
dnn4	0.5	2459603.5	2508397.75	2483142.167
dnn4_stacked2	0.5	2459763.25	2459763.25	2459763.25
naive	0.5	2508106.5	2508159	2508132.75
dnn1	0.5	2529752.5	2532724	2531238.25
Transformer	0.1	2662539.759	2662556.75	2662539.75
Transformer	0.05	2663109.0	2663109.0	2663109.0
dnn3_stacked2	0.5	2678950	2678950	2678950
cnn2	0.5	2736488.75	2736488.75	2736488.75
cnn1	0.1	2736489	2736492.5	2736490.75
cnn2	0.1	2736489.5	2736489.5	2736489.5
MobileNet	0.1	2736503.5	2736505.25	2736504.375
MobileNet	0.5	2736503.75	2736505.25	2736504.563
VGG16	0.1	2736504.5	2736505.25	2736504.75
MobileNet_2	0.1	2736505	2736505.25	2736505.125
MobileNet_2	0.5	2736505.75	2736508.25	2736507

Resultados por arquitectura

Arquitectura	Dropout	Pérdida de reconstrucción
dnn4	0.1	1033604.0
dnn4	0.1	1046479.9375
dnn4	0.1	1088231.5
dnn4_stacked	0.1	1272539.125
cnn1	0.5	1435283.125
cnn1	0.5	1447092.125
dnn3	0.1	1898245.875
dnn3	0.1	1901299.0
dnn3	0.1	1906601.375
VGG16	0.5	2053981.625

Top 10 mejores corridas obtenidas

Un aspecto importante a destacar en los resultados obtenidos a nivel de corrida indica que la arquitectura dnn 4 y cnn1 dominan los resultados. Otro factor a destacar es que para las arquitecturas de redes neuronales profundas un nivel de dropout es más beneficioso, mientras que para las arquitecturas con redes convolucionales es al contrario.



Enfocándonos solo en las mejores corridas de cada tipo de arquitectura podemos resaltar que la reducción en el indicador de pérdida entre la red naïve o de línea base que utilizamos es de por lo menos el doble.

Otro resultado digno de mención es el efecto contrario al deseado que obtuvimos haciendo stacks del mismo autoencoder, en donde el performance del mejor modelo se ve afectado.

IX. Analisis

Con los resultados nos damos cuenta de varias cosas, nuestra hipótesis de que modelos pre entrenados podrían descubrir cosas del espectrograma del sonido no fue tan efectiva ya que los resultados son relativamente peores que los de algunas arquitecturas entrenadas desde cero.

Otra cosa que no esperábamos fue que los modelos de atención no fueron tan eficientes en su ejecución, esto puede deberse a que tuvimos que reducirlos bastante para agilizar el entrenamiento y/o a el embedding posicional que recibía mucho poder en bruto.

Curiosamente las arquitecturas que mejor funcionaron fueron las redes densas. Esto lo explicamos un poco porque realmente estas redes están centrando todo su poder en relaciones temporales. Como los datos no son exageradamente anchos (como en tareas de nlp que tienen muchos tokens para identificar) la arquitectura puede relacionarse de forma relativamente buena algunas posiciones temporales con otras, aprendiendo de forma ágil.

X. Conclusiones

Los resultados nos muestran que es posible reducir el error de recomposición utilizando representaciones densas del sonido. Es decir, es posible reducir el ruido en grabaciones por las estratégias que desarrollamos. Aún así, la complejidad del problema y la necesidad de lo que consideramos arquitecturas más complejas, y mayor capacidad de procesamiento para llevarlo a un punto en el que cumpla el propósito de limpieza de sonido.

Referencias

[1] J. Li, W. Dai, F. Metze, S. Qu, y S. Das, «A comparison of deep learning methods for environmental sound detection», en *2017 IEEE International conference on acoustics, speech and signal processing (ICASSP)*, IEEE, 2017, pp. 126-130.

[2] H. Liu, L. Li, y J. Ma, «Rolling bearing fault diagnosis based on STFT-deep learning and sound signals», *Shock Vib.*, vol. 2016, 2016.

[3] A. Satt, S. Rozenberg, R. Hoory, y others, «Efficient emotion recognition from speech using deep learning on spectrograms.», en *Interspeech*, 2017, pp. 1089-1093.

[4] A. Huang y R. Wu, «Deep learning for music», *ArXiv Prepr. ArXiv160604930*, 2016.

[5] H. Purwins, B. Li, T. Virtanen, J. Schlüter, S.-Y. Chang, y T. Sainath, «Deep learning for audio signal processing», *IEEE J. Sel. Top. Signal Process.*, vol. 13, n.º 2, pp. 206-219, 2019.

[6] H. Qassim, A. Verma, y D. Feinzimer, «Compressed residual-VGG16 CNN model for big data places image recognition», en *2018 IEEE 8th annual computing and communication workshop and conference (CCWC)*, IEEE, 2018, pp. 169-175.

[7] W. Wang *et al.*, «A novel image classification approach via dense-MobileNet models», *Mob. Inf. Syst.*, vol. 2020, 2020.

[8] X. Zeng, Z. Zhang, y D. Wang, «AdaMax Online Training for Speech Recognition», *2016*, 2016.

[9] A. Nandan, «Text classification with Transformer». [En línea]. Disponible en: https://keras.io/examples/nlp/text_classification_with_transformer/