

Desarrollo y análisis de dispositivo para localización aviar por medio de vocalizaciones mediante el uso de Deep Learning

Julian Amette Estrada y Bianca Balzarini

Director: Gabriel Bernardo Mindlin ; Co-director: Roberto Andrés Bistel Esquivel

julianamette3@gmail.com ; balzarini.bianca@gmail.com

Resumen

Se realiza el desarrollo de un dispositivo de localización aviar por medio de vocalizaciones. Se obtiene un diseño de dispositivo y algoritmo de machine learning los cuales se prueban obteniendo resultados alentadores que mejoran métodos de minimización. Para dado dispositivo, se realiza un análisis de los errores obtenidos en simulaciones para estimar su eficiencia en pruebas de campo. El dispositivo final podría, según nuestros resultados, estimar posiciones de fuentes sonoras hasta 50 m de distancia con un error menor a 1 m en el 92 % de los casos mientras se tengan errores compatibles en delays temporales del orden de $\pm \frac{1}{11000}$ s. Se encuentra que el algoritmo es, sin embargo, robusto para errores mayores. El dispositivo final resulta: compacto, barato, con buena exactitud (según resultados parciales), de fácil colocación y sobre todo escalable para su implementación múltiple en pruebas de campo. Se diseña para posibilitar el desarrollo del dispositivo un método de corrección de desfases temporales para placas Arduino, permitiendo disminuir su desincronización relativa de 1 s a 0.0002 s al cabo de 200 s de grabación en los peores casos. Finalmente se realizan pruebas integrales que permiten corroborar la validez del sistema en su conjunto para ciertos casos particular, lo cual indicaría que su funcionamiento podría ser válido en condiciones más generales.

1. Introducción

El objetivo del presente trabajo es el desarrollo de un sistema de localización aviar por medio de vocalizaciones. El proyecto tiene como fin su utilización futura en pruebas de campo, lo cual lleva consigo un conjunto de restricciones para el diseño, tanto físico como de algoritmos. El dispositivo final debe ser de un tamaño reducido, de fácil colocación y que la misma requiera bajo conocimiento técnico. Además debe ser económico y con una exactitud en la localización lo más baja posible (alrededor de 1 m), hasta 50 m de distancia de un punto de referencia determinado. Por último, se utilizan receptores con una frecuencia de muestreo $f = 22\text{ kHz}$. Estas condiciones y características propias del problema restringen el diseño del dispositivo.

Se quiere usar un arreglo de micrófonos, cada uno en una posición \mathbf{x}_i tal que, cuando una fuente en una posición \mathbf{p} emita una señal en un tiempo t_0 , entonces se pueda localizar la misma. En la práctica, dado que las fuentes son aves, el tiempo t_0 de la emisión de la señal es desconocido. Entonces, los datos que podemos extraer de los micrófonos son los tiempos relativos de llegada de la señal (o una referencia de la misma) entre pares de receptores. La obtención de la posición a partir de esta información es conocida como localización por tiempo de diferencia de arribo (TDOA: Time Difference of arrival). El TDOA, a diferencia del tiempo de arribo (TOA, cuando se conoce t_0) es mucho más sensible a los errores. El tiempo de llegada de la señal al micrófono i está dada por (1):

$$t_i = \frac{|\mathbf{p} - \mathbf{x}_i|}{c} + t_0 \quad (1)$$

donde c es la velocidad del sonido en el medio. Las ecuaciones del problema, por ser las diferencias temporales del arribo de la señal entre micrófonos, se corresponden con (2).

$$t_i - t_j = \frac{|\mathbf{p} - \mathbf{x}_i|}{c} - \frac{|\mathbf{p} - \mathbf{x}_j|}{c} \quad (2)$$

Se podría calcular de manera analítica con distintos algoritmos la posición de la fuente a partir de la posición de los micrófonos y las diferencias temporales (también llamados delays) en caso de tener el número suficiente de ecuaciones. Estos poseen una muy mala respuesta a la presencia de errores en los delays para la estimación de la posición de las fuentes. Esto se acentúa cuando la distancia recorrida por el sonido en el tiempo de muestreo es comparable con la distancia entre micrófonos. En la sección (2) se mencionan las características particulares que se toman en cuenta para el sistema. Según el trabajo preliminar, se observa que la obtención de las posiciones de las

fuentes usando la frecuencia de muestreo $f = 22 \text{ kHz}$ mediante métodos analíticos posee un error mucho mayor al esperado para un dispositivo de tamaño portable, por lo que se decide no usar este tipo de técnicas y optar por una alternativa.

El método desarrollado en este trabajo es la utilización de algoritmos de aprendizaje profundo e inteligencia artificial para obtener las posiciones de las fuentes a partir de los delays temporales. La estrategia utilizada consiste en la sobredeterminación del problema y posterior implementación de algoritmos de machine learning que permitan modelar la posición de una fuente en función de las diferencias temporales.

1.1. Aprendizaje profundo

La inteligencia artificial representa un cambio de paradigma en el modo de enfrentar los problemas. En este caso se utiliza aprendizaje supervisado, es decir, un tipo de aprendizaje en el cual se usan como input, datos de los cuales se conoce el resultado. El objetivo es crear una red que pueda modelar las posiciones de las fuentes en función de las diferencias temporales. Resolver un problema con estos métodos requiere diseñar primero un modelo de aprendizaje con alguna estructura.

Se decide utilizar un Perceptrón multicapa (MLP por sus siglas en inglés) como método de obtención de las posiciones. Tiene sentido este modelo de red ya que no se espera una coherencia temporal entre las señales recibidas dado que un ave puede hacer ruido en una posición, luego volar en silencio y tiempo más tarde piar en una posición nueva. Un perceptrón multicapa puede tener distintas arquitecturas internas; cada una de ellas presentará distinto desempeño, cuyo estudio forma parte del trabajo. La elección de arquitecturas y número de neuronas por capa, además de las funciones de activación tienen motivos que exceden a la presente discusión. El desempeño particular en función de la profundidad de la red también es algo propio de machine learning que no se explicita en el presente informe pero se tiene en cuenta a lo largo de todo el trabajo.

Para resolver este problema, se utiliza un MLP que tiene un vector de dimensión N de entrada, donde N es el número combinatorio de la cantidad de micrófonos con dos (número total de delays). Se define su arquitectura como el número de capas ocultas que tiene y el número de neuronas en cada una de las mismas, sabiendo que la entrada tiene dimensión N y la salida 3 (por la posición x , y y z de la fuente). Se utiliza lo siguiente como notación para una arquitectura: $[c_1, c_2, \dots, c_k]$, que representa un MLP con k capas ocultas en las cuales la capa i tiene c_i neuronas, con $c_i \in \mathbb{N}$. Dentro de la arquitectura también hay otros parámetros. El otro que se tiene en cuenta aquí es la función de activación. No se especifica cual se usa dentro de la notación debido a que una vez determinado el tipo de función, todas las arquitecturas usan el mismo tipo.

Un MLP funciona, de forma simplificada, de la siguiente manera: dado un input \mathbf{X}^0 y una función de activación F , el estado de la capa \mathbf{X}^1 es el estado de activación de cada neurona X_i^1 y depende únicamente de \mathbf{X}^0 , los pesos $\mathbf{W}_{1,0}$ y el bias (o umbral de activación) b_1 , ya que es un modelo feed forward. Entonces, la activación de cada neurona X_i^m esta dada por la ecuación (3).

$$X_i^m = F \left(\sum_j W_{j,i}^m X_j^{m-1} - b_i^m \right) \quad (3)$$

Para entrenar un modelo, se parte de un conjunto de datos de entrenamiento \mathbf{E} , donde cada \mathbf{E}_i es una combinación de delays que produce una fuente particular, a la cual llamamos target \mathbf{E}_i^t . En la práctica, dado cual es el objetivo, los delays que se le presentan a la red están corruptos y en verdad son $\mathbf{E}_i + \delta\mathbf{E}_i$, donde $\delta\mathbf{E}_i$ es un arreglo de errores para cada delay, compatible con las restricciones ya mencionadas. Se define además una función costo C que determina la distancia en algún espacio elegido de manera conveniente entre el resultado que da el algoritmo y \mathbf{E}_i^t . Luego se usa back propagation (o algún algoritmo similar) para corregir los pesos $W_{i,j}^m$ y aproximar mejor el resultado. De esa manera, la red 'aprende' a pasar de un conjunto de delays a la posición de la fuente que genera los mismos. Se repite este procedimiento con el conjunto de entrenamiento tantas veces como sea necesario y luego se usa el modelo con datos nuevos y se computa su performance.

2. Desarrollo

El proyecto se divide en dos grandes partes. La primera de ellas es el desarrollo del sistema de localización, su viabilidad, el diseño de dispositivo y sus características. La segunda consiste en la construcción y testeo de los grabadores que se usarán en el dispositivo. Luego de estas dos etapas hay un período de integración donde se prueba el conjunto completo.

2.1. Sistema de localización

Se busca diseñar un arreglo de micrófonos y un algoritmo acordes a las necesidades ya previamente mencionadas. Hay un vínculo adicional, el cual consiste en mantener las simetrías del dispositivo para la semiesfera superior lo más posible (de acuerdo al número de micrófonos se podrá realizar un arreglo más o menos simétrico, pero como mínimo se busca que tenga simetría cada $\theta = \frac{\pi}{2}$). Se toma en cuenta que la frecuencia de muestreo máxima de los grabadores debía ser $f = 22 \text{ kHz}$ por restricciones técnicas y además porque es una frecuencia de muestreo típica de audio. Para asegurar la portabilidad se toma como restricción que el dispositivo debe estar contenido en una esfera de 1 m de radio. Se requiere un sistema de sincronización general para los receptores del arreglo. El diseño del dispositivo se decide a partir del uso de algoritmos de minimización (en particular, de descenso por gradiente), buscando la distribución óptima. Esto se hace bajo la premisa de que una vez encontrada una distribución que tenga una buena performance relativa con los los métodos de minimización, esta también tendrá un buen desempeño al usar Deep learning.

Se propone una colección de arquitecturas para buscar la cual, en los testeos, posee un error estadístico lo más bajo posible (teniendo errores por debajo de 1 m como objetivo), para fuentes a una distancia de hasta 50 m .

Para decidir la arquitectura adecuada, una vez decidido el arreglo de micrófonos se crean dos grupos de datos simulados: un grupo de entrenamiento y otro de testeo. En la realización de los mismos se considera que el error máximo en módulo en la estimación de las diferencias temporales entre los distintos micrófonos es $\frac{1}{11000} \text{ s}$, ya que el tiempo de llegada a cada receptor tiene como error la frecuencia de muestreo.

Se crea un grupo de datos de entrenamiento con 3000 fuentes aleatorias en una esfera de 50 m de radio desde el punto medio del dispositivo. A su vez, cada una de estas fuentes posee 10 combinaciones aleatorias distintas de errores posibles en la estimación de las diferencias temporales. Así, finalmente, se tienen 30.000 ejemplos para este conjunto de datos de entrenamiento. Este es el grupo que se utiliza para entrenar todas las arquitecturas por igual, buscando la que tenga el mejor desempeño. Por otro lado, se crea un conjunto de datos de testeo con 500.000 fuentes aleatorias con un error aleatorio dentro del rango $\pm \frac{1}{11000} \text{ s}$ para cada diferencia temporal entre micrófonos.

Una vez obtenidos los datasets, se entrena las diferentes arquitecturas y se computan sus errores con las bases de datos de testeo. Luego se comparan los resultados. Hay dos variables que se tienen en cuenta: el número de parámetros entrenables que posee la red y su error al calcular fuentes que no pertenezcan al conjunto de entrenamiento. Lo segundo es tenido en cuenta para obtener la exactitud buscada, y el número de parámetros entrenables es una cantidad a mantener lo más bajo posible; de esta manera se evita el overfitting y se tiene un modelo menos pesado y mas rápido, el cual se podría colocar, a futuro, en una placa portátil sin ocupar mucha memoria.

Una vez elegida la arquitectura con mejor desempeño en la prueba mencionada, se realiza un análisis más profundo de la misma. Primero se realiza un entrenamiento con un conjunto de datos mayor para un número relativamente alto de épocas. En segundo lugar se estudia la dependencia angular y radial de los errores de testeo. Se entrena la arquitectura por 650 épocas usando un dataset de 6 millones de ejemplos. Luego, para testear se crean 10 datasets, cada uno de ellos a radio constante, cada 5 m entre 5 m y 50 m inclusive, con $\theta \in [0, \frac{\pi}{2}]$ y $\phi \in [0, \frac{\pi}{2}]$. El sistema de referencia se encuentra centrado en el dispositivo. Cada uno de los 10 datasets se divide en 40 valores tanto en θ como en ϕ . Para cada par θ, ϕ se toman 1000 fuentes con errores aleatorios en el rango $\pm \frac{1}{11000} \text{ s}$. Es importante notar que el espacio seleccionado es solo un cuarto de la semiesfera superior, debido a que en el diseño del dispositivo se espera una relación simétrica con las demás partes de la semiesfera. La semiesfera inferior no se estudia en particular ya que, al estar el dispositivo apoyado sobre el piso, no es la de mayor interés para este problema. De todos modos, esta zona esta incluida en todo el resto de los análisis de testeo del algoritmo.

Por otra parte, se realiza un análisis de robustez de esta arquitectura. Para esto, se aumenta progresivamente el error en los datasets de testeo para ver la respuesta del algoritmo. Se repite la prueba con una red entrenada con errores del doble, es decir $\pm \frac{2}{11000} \text{ s}$, y se testea con los mismos datasets para ver las diferencias en la exactitud de los resultados.

2.2. Construcción y testeo de los grabadores

Es necesario, para el funcionamiento del algoritmo, que las grabaciones sean continuas, sin pérdidas y sincronizadas. Se probaron dos alternativas de grabadores: una placa previamente desarrollada por el laboratorio de sistemas dinámicos, la cual se construyó según los diseños de trabajos anteriores usando como procesador un MSP-430G2553, y por otro lado, una plataforma Arduino Uno. Ambas tienen un slot para conectar una tarjeta micro SD donde se graban los datos.

Para probar las placas se programan los procesadores para que graben en las tarjetas micro SD una rampa periódica generada por el mismo sistema a una frecuencia de 22 kHz . Se estudia entonces la pérdida de puntos en estas señales.

Una vez hecha la prueba de pérdida de puntos, se puede realizar la prueba de sincronía. Esta consiste en la grabación simultanea, por medio de dos placas, de una señal externa periódica. Luego se puede ver el desfasaje entre

ambas grabaciones en función del tiempo, el cuál, en el caso ideal, debería ser nulo. Esta se hace a partir de la grabación de funciones cuadradas y medición del desfasaje para cada uno de los pulsos.

2.3. Integración

Para realizar la integración del algoritmo de posicionamiento con la construcción de los grabadores, se realizan pruebas midiendo diferencias de tiempo de arribo y luego se intenta estimar características de la posición de la fuente.

En primer lugar, se ubican dos grabadores a 1.2 m de distancia. Con 2 micrófonos, las condiciones del problema analíticamente alcanzan sólo para calcular la dirección de incidencia de la fuente sonora en 2 dimensiones. Entonces, a partir de los delays y sin utilizar machine learning, sino usando la aproximación de campo lejano, se calculan los ángulos de incidencia de fuentes ubicadas a menos de 5 m del centro de la configuración de micrófonos.

Se denota como D al delay temporal entre ambos micrófonos, es decir, la diferencia del tiempo de arribo de la señal entre ellos. Si se llama α al ángulo entre la fuente y la línea de los micrófonos, entonces, usando la aproximación de campo lejano se tiene que: $D = \frac{1.2m}{c} \cos \alpha$. De esta ecuación se despeja el ángulo de incidencia.

Por otra parte, usando 3 grabadores, analíticamente alcanzan los grados de libertad para recuperar la posición en dos dimensiones de la fuente. Se colocan 3 micrófonos en las posiciones: $(0.6, 0)\text{ m}$, $(-0.6, 0)\text{ m}$ y $(0, 1.05)\text{ m}$; Luego se graban distintas fuentes sonoras nuevamente ubicadas a no más de 5 m del centro de la configuración de micrófonos, y en el mismo plano que ellos. De las grabaciones se extraen los delays y luego se utilizan como input en un MLP previamente entrenado para estas condiciones. Esta prueba sirve para testear integralmente al sistema y la concordancia en el funcionamiento entre el algoritmo de inteligencia artificial y los grabadores.

3. Resultados

3.1. Sistema de localización

En primer lugar, es necesario definir todo lo que concierne al sistema de localización. Es decir, la disposición de los micrófonos y la arquitectura del perceptrón multicapa tal que los errores de posicionamiento estén dentro del rango deseado, siempre teniendo en cuenta las restricciones que impone el problema.

3.1.1. Elección de configuración

Para decidir el arreglo de micrófonos, se analiza la relación entre cantidad de micrófonos, precio, portabilidad y errores. Usando un algoritmo de minimización de descenso por gradiente (el cual se desarrolló previamente pero se omite en este informe por su extensión, ver el apéndice A) se concluye que la mejor solución es utilizar dos dispositivos de 5 micrófonos cada uno, independientes entre sí (para ver el diseño final del dispositivo listo para el uso ver el apéndice B). De esta manera, se tienen dos dispositivos que funcionan en conjunto, cada uno portable por si mismo. Cada uno de ellos tiene micrófonos localizados, usando de referencia el centro del dispositivo, en:

$$\mathbf{x}_1 = (1, 0, 0)\text{ m} ; \mathbf{x}_2 = (-1, 0, 0)\text{ m} ; \mathbf{x}_3 = (0, 1, 0)\text{ m} ; \mathbf{x}_4 = (0, -1, 0)\text{ m} ; \mathbf{x}_5 = (0, 0, 1)\text{ m} \quad (4)$$

Se separan los mismos una distancia de 30 m entre sus ejes y se los toma alineados. Sobre este arreglo de micrófonos se realiza la generación de los datasets calculando, para cada una de las fuentes, los tiempos de arribo relativos entre receptores. Esta configuración posee, desde el centro de los dos dispositivos, simetría de $\frac{\pi}{2}$ en la semiesfera superior.

3.1.2. Elección de arquitectura

Una vez definida la configuración de los micrófonos, se proponen las siguientes arquitecturas: [256], [512], [64, 32], [64, 64], [128, 64], [128, 128], [128, 64, 32] , [256, 128], [32, 32, 32], [64, 64, 64], [32,32,32,32], [64,64,64,64] y [128,64,32,16]. Estas son un total de 13 arquitecturas distintas. Se quiere elegir la que sea más adecuada para el presente problema. Todas ellas poseen en las capas internas la tangente hiperbólica como función de activación y lineal en la capa de salida. Además se usa como función costo el error cuadrático medio. El método de minimización es RMSprop con un learning rate de 10^{-4} .

En la figura (1) se grafican las comparaciones entre los resultados del testeo con datasets de 500.000 fuentes luego de haber entrenado todas las arquitecturas con el dataset de 30.000 muestras durante 1500 épocas. Cada una de las arquitecturas es un punto en un gráfico que posee dos ejes: uno en el que se ve la mediana de los errores sobre los datos de testeo, y otro en el que se ve el número de parámetros entrenables que posee cada uno de los modelos.

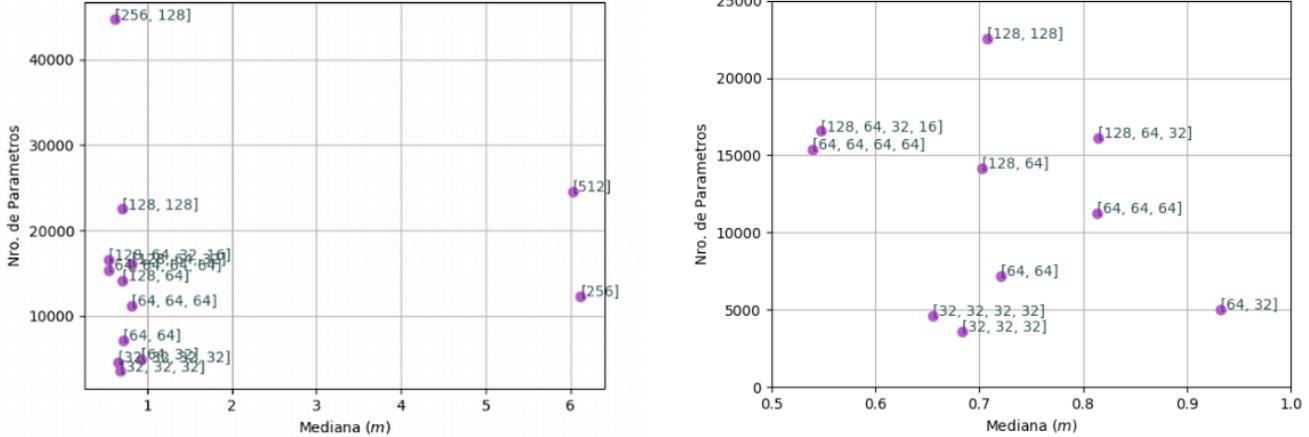


Figura 1: Comparación de las arquitecturas. Se muestra, para todas las arquitecturas analizadas, el número de parámetros en función de la mediana de los errores. Todas fueron entrenadas durante 1500 épocas con un dataset de 30.000 ejemplos, y luego testeadas con un dataset de 500.000 ejemplos. A la derecha se ve un acercamiento de la figura de la izquierda en la zona de menor número de parámetros y menor mediana.

Como se explicó anteriormente, lo óptimo es obtener tanto la menor mediana como el menor número de parámetros posible. Se observó que las arquitecturas más profundas poseían mejores resultados, obteniendo medianas más bajas acompañadas de un número de parámetros relativamente chico, en comparación a las arquitecturas con pocas capas ocultas y gran número de neuronas en ellas. Para obtener medianas menores a 1 m se necesitaron arquitecturas con un mínimo de dos capas ocultas en ellas. El modelo seleccionado fue el de [64,64,64,64], teniendo (sin contar los bias) 15.360 parámetros entrenables y el mejor desempeño relativo en el entrenamiento.

3.1.3. Resultados de entrenamiento y testeo

Una vez seleccionada la arquitectura de mejor desempeño, se la entrena con un dataset más robusto para tener una medida de que tan buenos resultados es capaz de proporcionar. Esto se hace durante 650 épocas, con un dataset de entrenamiento de 6.000.000 de ejemplos. El aumento en el número de ejemplos que ve la red durante su entrenamiento evita el overfitting y promueve que este pueda mapear de mejor manera los valores en función de los delays temporales. Luego se la prueba con un dataset de 1.000.000 de ejemplos. Se obtiene una media de los errores de 0.47 m y una mediana de 0.34 m . La distribución de errores de esta arquitectura se encuentra en la figura (2). Se puede ver que la misma es una distribución de cola larga por lo que posee casos excepcionales con un error alto. Sin embargo, se encuentra que el 92 % de los errores están por debajo de 1 m y el 99 % por debajo de 2.56 m . De esta manera se concluye que la cola tiene un número de puntos mucho menor al resto de la distribución, siendo estos resultados positivos para el objetivo buscado.

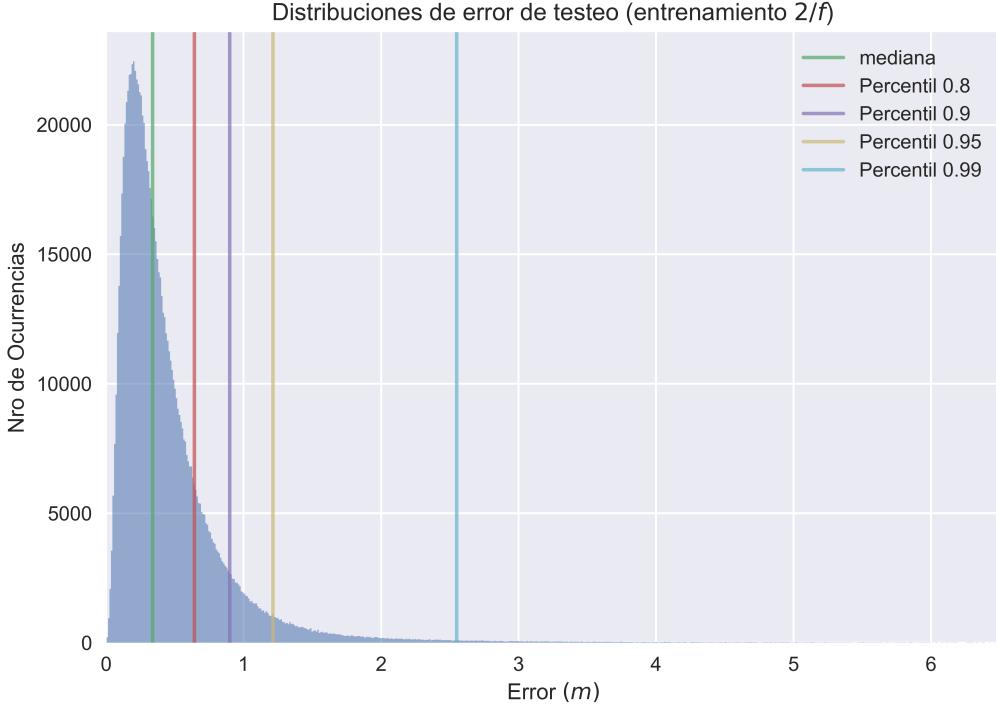


Figura 2: Histograma de los errores de testeo para la arquitectura $[64,64,64,64]$ entrenada durante 650 épocas con un dataset de 6.000.000 ejemplos y testeada con uno de 1.000.000, ambos con errores aleatorios de $\pm \frac{1}{11000} s$. Se marcan en lineas verticales los percentiles que contienen a todos los valores de error de testeo con valor menor.

Es posible también el estudio de la importancia de cada uno de los 45 delays temporales (cada diferencia de tiempo de arriba entre pares de los 10 micrófonos). Se encuentra que todos los delays son importantes, con un conjunto de ellos como los principales para la determinación de las posiciones. Para ver un análisis más detallado de esto ver el apéndice C.

Análisis angular y radial

Para entender la distribución angular de los errores, se puede hacer un análisis como el descrito en la sección 2.1. Es decir, testear el modelo tomando conjuntos de datos de testeo a radios fijos ($5 m, 10 m, \dots, 50 m$) desde el centro del dispositivo, y dividiendo cada uno de esos espacios en grillas de 40×40 (40 valores de θ y 40 de ϕ). Además, solo se considera la semiesfera superior, ya que la inferior no es tan relevante al estar el dispositivo sobre el piso. De la semiesfera superior se analiza solo un cuarto, dado que por la simetría del dispositivo, los 3 cuartos restantes son equivalentes.

Para cada par θ, ϕ , se toman 1000 ejemplos, cada uno con un error aleatorio en el rango $\pm \frac{1}{11000} s$. Para visualizar los resultados, se grafican las 40×40 medianas de error para cada radio. Esto se muestra en la figura 3 para radio $5 m$ y $45 m$. Se colocan distintos valores como límites de la barra de color para poder tener resolución en los errores. El resto de las figuras se puede ver en el apéndice D.

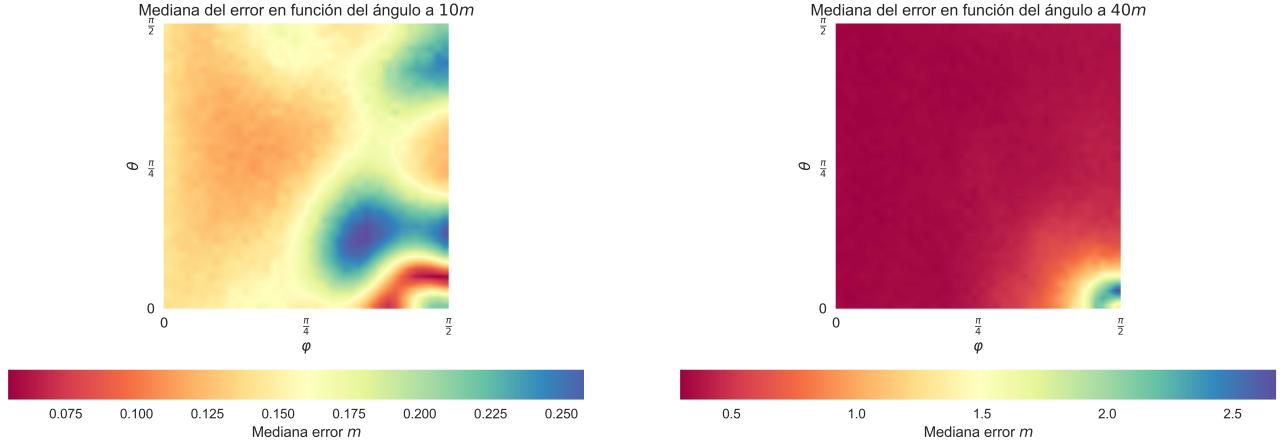


Figura 3: Mediana de errores de testeo para zonas de radio constante, ubicadas en un cuarto de la semiesfera superior, divididas en 40 valores de θ y 40 de ϕ .

Mediante el análisis en función de los radios se ve que para radios chicos no se encuentran zonas donde el error sea considerable. A partir de los 15 m se encuentra que comienza a existir una zona con error relativamente grande. Para ver que representa esta zona, hay que tener en cuenta que las fuentes se crean de la siguiente manera:

$$(x, y, z) = R(\cos(\theta) \sin(\phi), \sin(\theta) \sin(\phi), \cos(\phi)) \quad (5)$$

donde R es el radio de las fuentes de testeo. Por lo tanto, la zona de alto error se encuentra centrada en el eje \hat{x} . Ya que ambos dispositivos se encuentran sobre este mismo eje, estas son aquellas que están en la linea que atraviesa el centro de ambos dispositivos. Coincidir con estar a una distancia grande de uno de los dispositivos lo que genera que los aportes que puede hacer este para el posicionamiento se vean severamente afectados, además que presenta muy alta simetría en las diferencias temporales para ambos aparatos (lo cual también genera problemas en la estimación de la posición).

Sin embargo, en toda la zona intermedia entre ambos dispositivos el error se encuentran dentro de los límites estipulados, por lo tanto se pueden realizar mediciones muy exactas allí. Desde un punto de vista práctico no afecta la implementación del sistema en sí. Conociendo la distribución de errores, dada una zona de interés es posible colocar los dispositivos para maximizar su eficacia en la localización.

Se puede calcular para cada uno de los radios cual es el quantil 0.9 de los errores, es decir, el valor para el cuál el 90 % de los puntos de testeo tienen error por debajo del mismo. Segregándolo para los casos en los que se toma en cuenta la zona de alto error y los que no se puede ver en la figura 4 los resultados.

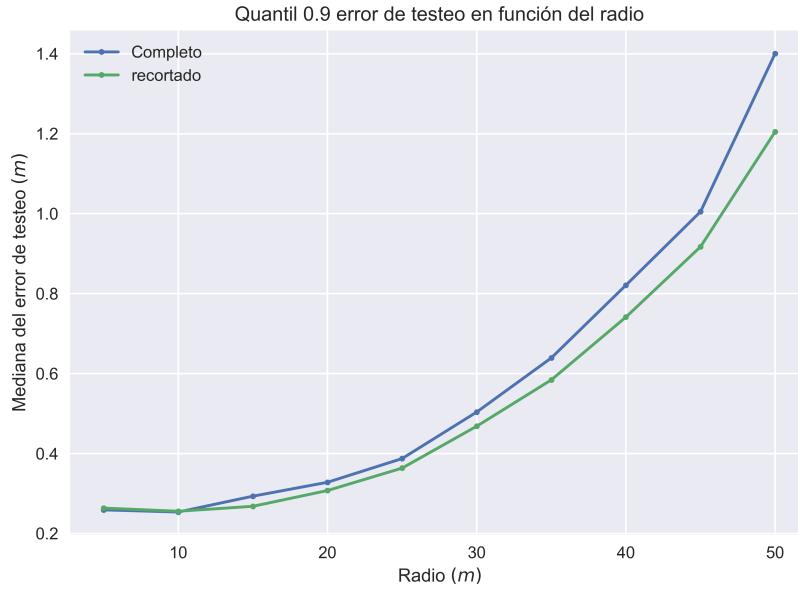


Figura 4: Cuantil 0.9 de los errores de testeo en función del radio, incluyendo y excluyendo la zona de alto error que se ve en la figura 3.

Es notable que hay una mejora significativa de los errores al eliminar la zona de alto error que se ve amplificada para radios mayores. Se llega a aproximadamente 20 cm de diferencia entre cuantiles 0.9 para 50 m en el caso en el cual la zona no es tenida en cuenta, lo que supone una mejoría sustancial.

Seleccionando el espacio que no contiene el error alto, es posible graficar diferentes valores de los cuantiles en función del radio. Esto permite obtener un gráfico que muestra la performance esperada del dispositivo en función del radio, fuera de la zona de conflictos. Esta información se encuentra en la figura 5

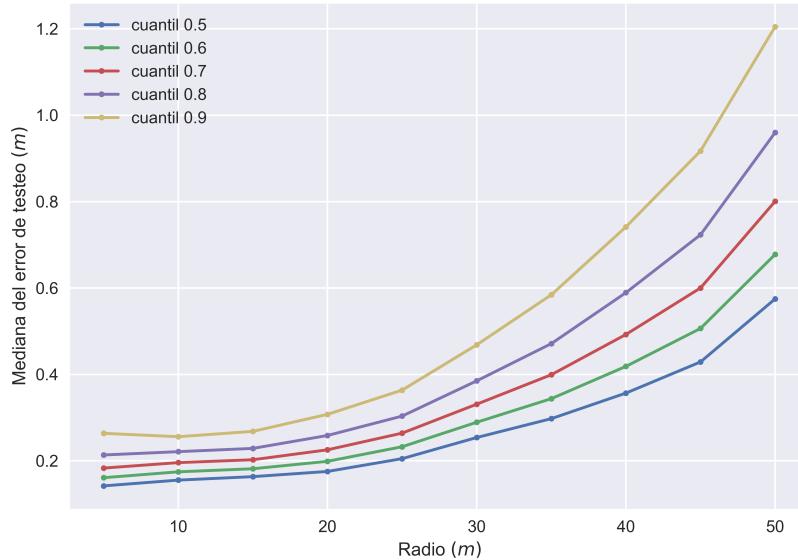


Figura 5: Distintos cuantiles del error de testeo en función del radio excluyendo la zona de alto error que se ve en la figura 3.

Otra característica a estudiar del sistema es la relación entre el error en módulo en la posición de la fuente predicha por el algoritmo y el error en módulo del vector formado por los 45 delays que le fue asignado aleatoriamente a la fuente. Es esperable que cuanto mayor sea el error en los delays, mayor sea el error en la posición. En la figura 6 se puede ver un scattering de los valores del error en función del módulo del error de los delays. Esta figura contiene $40 \times 40 \times 1000$ fuentes, para el caso particular de radio fijo 45 m . Para los otros radios, los resultados son análogos. No se muestran todos los resultados para facilitar la visualización de los datos. Además, se divide el gráfico en la zona con errores altos y la zona fuera de esta.

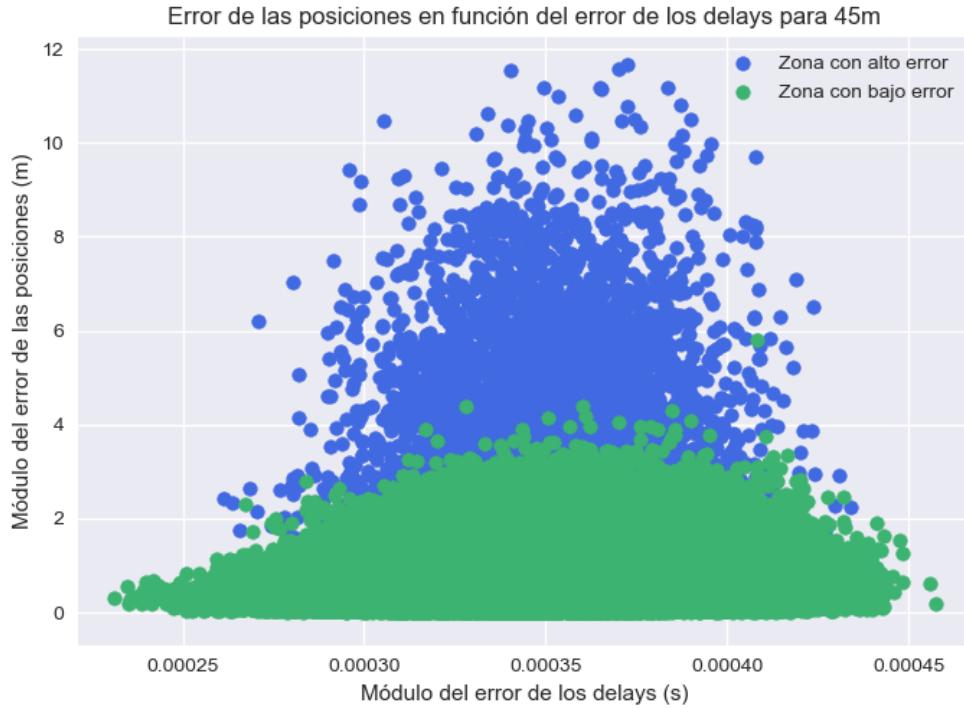


Figura 6: Error en la posición en función del error en los delays para cada una de las $40 \times 40 \times 1000$ fuentes, para el caso de radio 45 m . En azul se muestran las fuentes correspondientes a la zona con mayor error, y en verde las que corresponden a la zona de menor error.

Se puede ver que las fuentes con un error mayor en los delays tienen una tendencia marcada a tener errores más grandes en las posiciones. Más allá de sus valores estadísticos, los máximos errores se sitúan alrededor de 12 m mientras que en la zona de bajo error, fuera de un punto considerado outlier, todos se concentran por debajo de los 6 m . Si bien esta relación no es tan obvia, sí es consistente con lo que se esperaba encontrar. Si se deseara hacer un análisis mas profundo, se puede notar una tendencia a aumentar los errores en la localización al aumentar los valores del modulo de error en los delays. Puede ser poco evidente en el gráfico pero es debido a una cuestión estadística: hay muy poca cantidad de puntos con errores extremos en el conjunto de delays.

Análisis de Robustez

Para analizar que tan robusto es el algoritmo de deep learning desarrollado, se realiza un conjunto de pruebas donde se testea su desempeño para condiciones más diversas. En primer lugar, se entrena a la red usando el dataset de entrenamiento con errores de $\pm 2/f$, donde f es la frecuencia de muestreo ya mencionada. Luego se testea la red con distintos errores en el dataset de testeo: $\pm 2/f$, $\pm 4/f$, $\pm 6/f$, $\pm 8/f$. Si el algoritmo es robusto, al aumentar el error de testeo, el error en las posiciones debería aumentar pero no de forma excesiva. Se esperaría para un algoritmo óptimo que tenga un desempeño aceptable para condiciones fuera de las cuales fue entrenado. Se repite el mismo procedimiento pero entrenando con error $4/f$. Los histogramas de los errores obtenidos en las posiciones en ambos casos se encuentran en la figura 7.

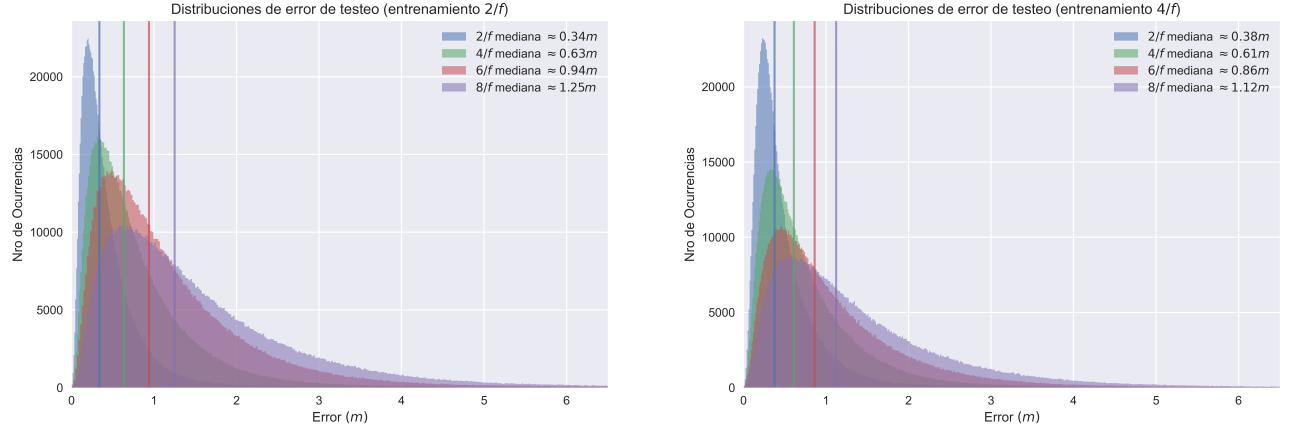


Figura 7: Derecha: Histograma de los errores obtenidos para la arquitectura [64,64,64,64] entrenada con error $\pm 4/f$, para distintos errores en el dataset de testeо. Izquierda: Histograma de los errores obtenidos para la arquitectura [64,64,64,64] entrenada con error $\pm 2/f$, para distintos errores en el dataset de testeо. En ambos casos se muestran las medianas de los errores como lineas verticales.

Como se ve en la figura 7 la mediana crece lentamente a medida que se agranda el error en el dataset de testeо. Además, la cola de la distribución va alargándose pero no de forma brusca. Todo esto es un indicio de robustez en el algoritmo y señala la posibilidad de obtener buenos resultados tanto para los errores esperados, como errores mayores. Se puede ver que, para el entrenamiento con errores de $4/f$ el comportamiento es el mismo. Es notable que entrenando con el error mayor, se obtiene un desempeño general mejor para un espectro mas amplio de errores, a excepción del rango de error particular en el que se entrenó la otra red.

Podemos ver esto en detalle lo mencionado, dadas las dos redes entrenadas con errores $\pm 2/f$ y $\pm 4/f$; y luego testeadas con error $\pm 2/f$ (mínimo error esperado para la frecuencia de muestreo). Las distribuciones obtenidas se muestran en la figura 8.

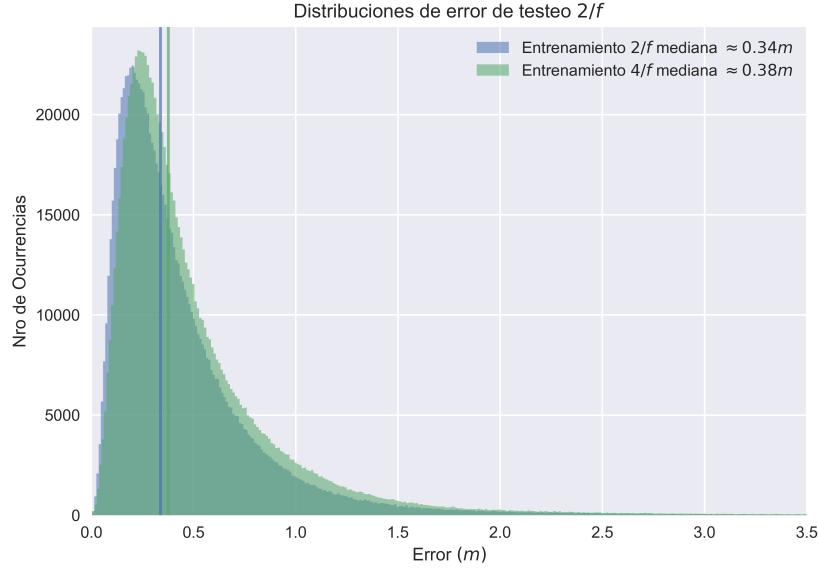


Figura 8: Histograma de los errores obtenidos para la arquitectura [64,64,64,64] entrenada con error $\pm 2/f$ y luego con error $\pm 4/f$, en ambos casos testeada con error $2/f$. En ambos casos se indica la mediana de los errores obtenidos.

En la figura, puede notarse que el entrenamiento con error $2/f$ posee una distribución de errores con un soporte mas chico que la de errores $4/f$. En casos de aplicación, logrando un error de $2/f$, el desempeño de la red entrenada

con ese mismo error sería el óptimo por dar un error general mejor, más allá de que las medianas de ambas sean similares. Esto se debe a que en el caso que tiene menor error de entrenamiento, el histograma se encuentra mas concentrado en la zona de modulo de error de testeo baja.

3.2. Construcción y testeo de los grabadores

Para que el sistema funcione correctamente, son necesarias dos condiciones: que los grabadores puedan tomar datos continuamente y que lo hagan de forma sincrónica. Si esto no se cumple, las diferencias temporales entre grabadores no serían fiables con lo cuál no sería posible localizar las fuente sonoras con un error razonable.

3.2.1. Corrección de sobreescritura

Se quiere ver si existe una pérdida de puntos al realizar grabaciones con los dos tipos de placas previamente mencionadas en el desarrollo. Dada la frecuencia de muestreo utilizada y el hecho de que las tarjetas SD poseen una tasa de transferencia de datos propia además de un protocolo que introduce un retardo de manera no periódica, entonces pueden surgir pérdidas de puntos dentro del grabado de los datos. Realizando los pasos explicados en el desarrollo con la placa en el MSP-430G2553, enviando una rampa periódica y usando una frecuencia de muestreo de 22 kHz , se encuentra en todos los casos, para todas las tarjetas SD probadas, resultados como los que se ven en la figura 9. Estos resultados no dependen de la clase de la tarjeta, la cual indica la velocidad de transferencia que tiene hacia la memoria. Se usan múltiples tipos de clase de tarjetas sin éxito, dado que siempre se encuentran pérdidas de puntos. Sin embargo, como es esperable, a medida que la clase de la tarjeta es superior, esta pierde menos puntos.

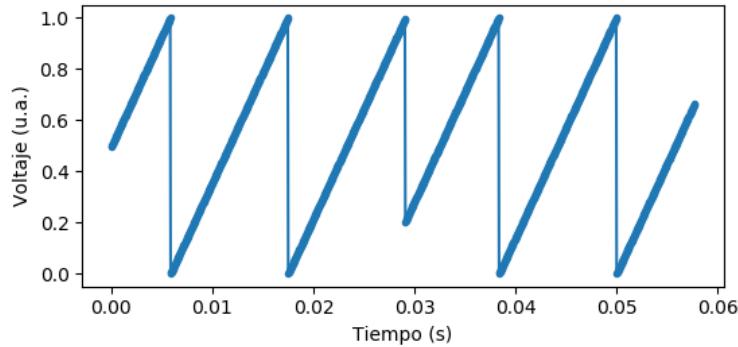


Figura 9: Medición usando una MSP-430G2553 con una tarjeta micro SD clase 10. La señal de entrada es una rampa. Se observa una pérdida de puntos entre 0.02 s y 0.03 s .

Las pérdidas se observan independientemente de si las tarjetas están previamente formateadas o no. Además, no son periódicas ni uniformes, lo cual las hace no modelables y dependen de cada tarjeta particular. Esto significa que la tasa de adquisición de datos es mayor que la tasa de guardado de los mismos. El procesador posee un buffer, cuyo objetivo es acumular los puntos previo a su grabación. Dadas las condiciones de medición, este se llena y comienza a sobregrabar puntos lo que conlleva directamente a la pérdida de puntos vista en las grabaciones.

Se puede agrandar el buffer existente para disminuir la pérdida de datos. Sin embargo, esto no logra evitar la falla, ni siquiera cuando el buffer es tan grande como la memoria lo permite. Es por eso que este tipo de procesador no se considera adecuado para ser utilizado en el dispositivo.

El segundo tipo de plataforma utilizada es Arduino Uno, la cual posee una memoria RAM de 2 kB , aproximadamente 4 veces mayor a la del MSP-430G2553 previamente mencionado. Además, por su arquitectura, en caso de que el buffer se complete es posible contar cuántos elementos se pierden y en qué lugar temporal.

Si bien la pérdida de datos sigue existiendo y por lo tanto la señal está corrupta; tener conocimiento de la posición en las grabaciones donde se perdieron puntos además del número de datos faltantes posibilita que las mismas sean completadas luego. Esto es una parte básica del esquema de grabaciones que se ha de utilizar para que estas placas puedan usarse en el sistema.

3.2.2. Pruebas de sincronía

Para corroborar que la sincronía es efectiva luego de completar las grabaciones como se menciona en la sección anterior, se realiza el procedimiento descrito en el desarrollo. La misma únicamente se realiza con la plataforma

Arduino debido a que, por la imposibilidad de corregir los errores de sobreescritura, el MSP-430G2553 queda descartado. Se envía una señal cuadrada periódica externa a un par de placas simultáneamente. Luego, debido a que cada placa tiene una latencia en el comienzo se alinean las grabaciones con el primer pulso recibido. Si ahora se grafica el desfase entre cada par de cuadradas de ambas grabaciones en función del tiempo, se obtiene lo que se ve en la figura 10.

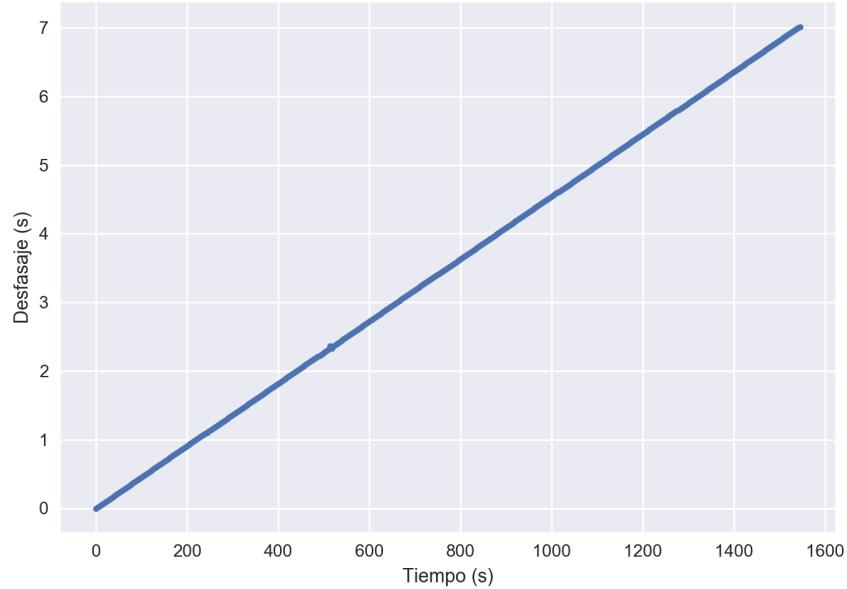


Figura 10: Desfase en función del tiempo de las mediciones simultáneas alineadas por el primer pulso recibido de una señal periódica en dos Arduinos en mediciones a 22 kHz .

Lo que se ve en la figura 10 muestra que no hay un desfase constante, en cuyo caso, debido a la alineación debería ser idénticamente nulo. Se puede ver entonces que el mismo es variable en el tiempo. En particular, en la medición ilustrada, al cabo de 200 s se tiene un desfase entre las Arduinos de alrededor de 1 s . Y luego de 1600 s , alrededor de 7 s .

Debido a que la forma de los desfases en función del tiempo sigue una relación lineal muy marcada, es posible corregir las mediciones mediante un ajuste. Al saber que los desfases entre ambos grabadores crecen linealmente y al conocer la tasa (o la pendiente) de este crecimiento, es posible partir de un delay indicado por los grabadores y corregirlo para obtener el delay real. Llamando D_{med} al delay extraído de las grabaciones luego de ser alineadas, P a la pendiente del desfase en función del tiempo, D_{real} al desfase real que existe en la llegada del pulso externo entre las fuentes (en este caso debería ser siempre 0), y t al tiempo desde el pulso de alineación, entonces se cumple lo siguiente:

$$D_{real} = D_{med} - P \cdot t \quad (6)$$

Una vez definido el procedimiento para la corrección de los delays temporales, queda determinar el valor de la pendiente, y bajo qué condiciones permanece constante para un conjunto particular de grabadores. Se tienen disponibles 4 placas Arduino (nomencladas como a , b , c y d), lo primero que se analiza es si la pendiente depende del par de Arduinos que se esté utilizando. Para esto, se repite la medición que se hizo para realizar la figura 9 con todas las combinaciones posibles de pares de Arduinos. Para cada par se realizan varias repeticiones del mismo experimento. En cada prueba se extrae la pendiente mediante un ajuste lineal. En la figura 11 se grafican todos los valores de las pendientes obtenidas, usando el mismo color para las mediciones que corresponden al mismo par de Arduinos. Allí también se muestra un acercamiento a las mediciones correspondientes al par de Arduinos $a - c$.

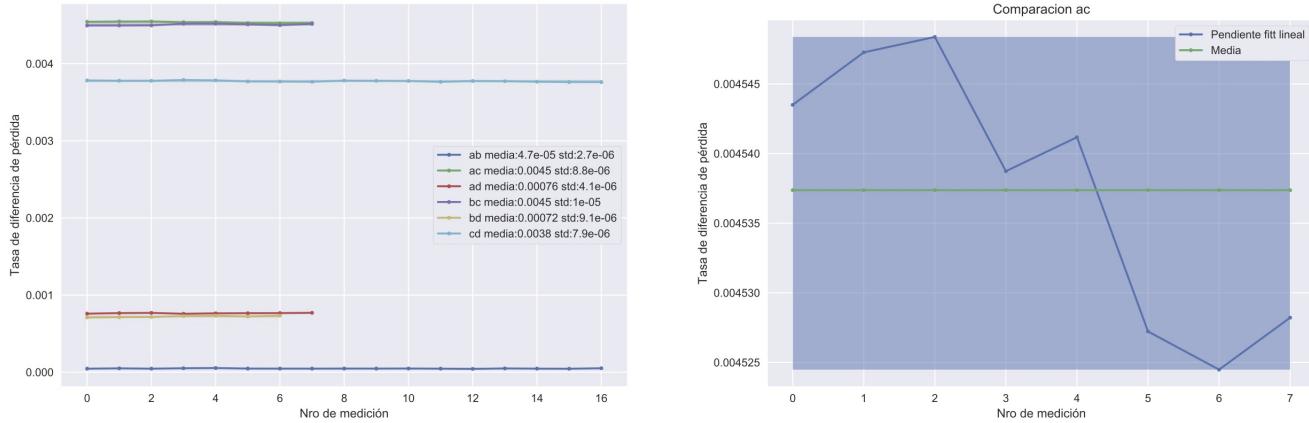


Figura 11: Izquierda: Pendientes obtenidas para cada par de Arduinos, repitiendo el experimento varias veces para cada par. Derecha: Acercamiento a las mediciones correspondientes al par de Arduinos $a - c$.

Como se ve en el gráfico de la izquierda de la figura 11, para un par de Arduinos dado, la pendiente obtenida posee cierta consistencia. Siempre que se calcula para un par particular se encuentra cerca de cierto rango y esta diferenciado de otros pares. Este hecho señala que el origen de la deriva en el delay temporal es que cada una de las placas posee un tiempo de retardo propio debido a procesos internos el cual no se encuentra computado en la corrección de sobreescritura. Esto implicaría en principio que cualquier medición hecha con un mismo par, se podría corregir siempre usando la misma pendiente. Sin embargo, como se muestra en el gráfico de la derecha de la figura anterior, al ver las pendientes obtenidas con el par $a - c$ se observan fluctuaciones alrededor de la pendiente media. Lo mismo ocurre para el resto de los pares de Arduinos. Las variaciones presentes, debido a la alta frecuencia de medición no permite realizar una corrección única para cualquier grabación con un par de placas particular. Para corroborar esto, se realizan nuevas mediciones para el par $a - c$ y se calcula la media de las pendientes obtenidas. Luego, se corrigen estas mediciones con la pendiente media después de alinear con el primero de los pulsos y se vuelve a graficar el desfase en función del tiempo. Se muestran en la figura 12 los resultados obtenidos.

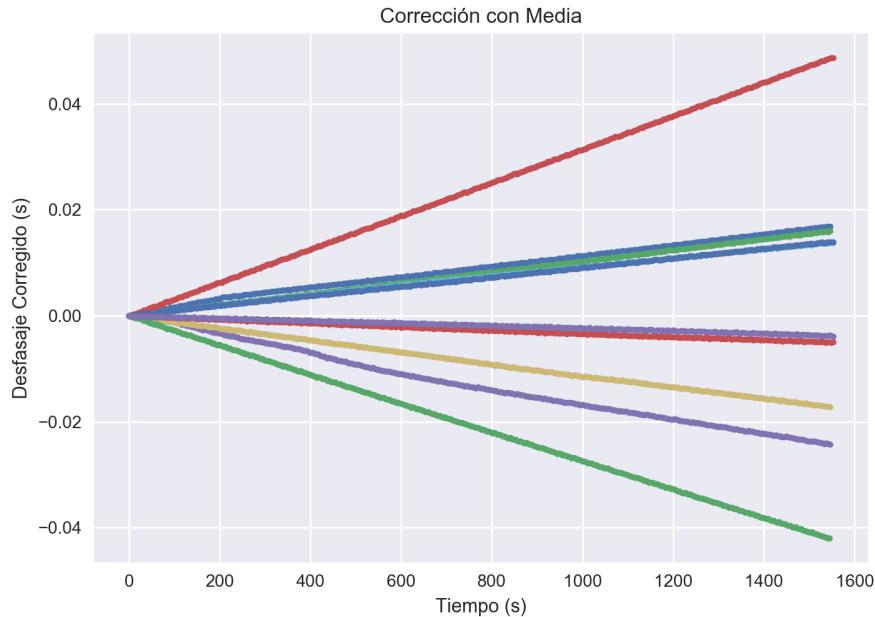


Figura 12: Desfases en función del tiempo para grabaciones alineadas con el primer pulso recibido para varias mediciones hechas con el par de Arduinos $a - c$ luego de corregir los desfases con la pendiente media.

En una situación ideal, en la figura 12 todas las rectas serían horizontales e identicamente nulas, lo cual indicaría sincronía luego de la corrección. Esto no es así, pero las pendientes de las rectas disminuyen significativamente con respecto a lo obtenido cuando no se realiza ninguna corrección. Para ver esto, se puede comparar con lo obtenido en la figura 10. En esta figura, el desfase luego de 200 s era de 1 s, mientras que con la corrección con la media, el desfase máximo luego de 200 s es de alrededor de 0.01 s. Esto es una gran mejora, sin embargo sigue si ser suficiente, ya que al ser la frecuencia de muestreo de $f = 22 \text{ kHz}$, 0.01 s representa alrededor de 200 puntos de error. Es notable entonces ver que las pendientes del desfase entre pares de arduinos depende de las condiciones de cada grabación en particular. Se especula que podría depender de la tensión que recibe cada Arduino, temperatura y factores internos de cada placa.

Se concluye que corregir con una pendiente fija a los desfases en función del tiempo conlleva un error que no es aceptable para las condiciones deseadas del problema. Se propone un método distinto de corrección. El mismo consiste en corregir cada punto con la pendiente de la recta de delay en función del tiempo que le corresponde. En la práctica esto se realiza superponiendo pulsos periódicos enviados externamente directamente conectados a cada grabador. Para cada par de pulsos se calcula la pendiente local, y todos los delays que estén posicionados entre ambas cuadradas se corregirán con ella. Esto permite evitar cometer un error en la corrección por las condiciones particulares de cada tiempo en cada medición. Esto es lo que se nombra como *corrección local*. Si se repite el gráfico de la figura 12 pero usando la corrección local, se obtiene la figura 13 para los primeros 200 s. En la misma la corrección se realiza únicamente con dos pulsos enviados con un espaciado de alrededor de 200 s

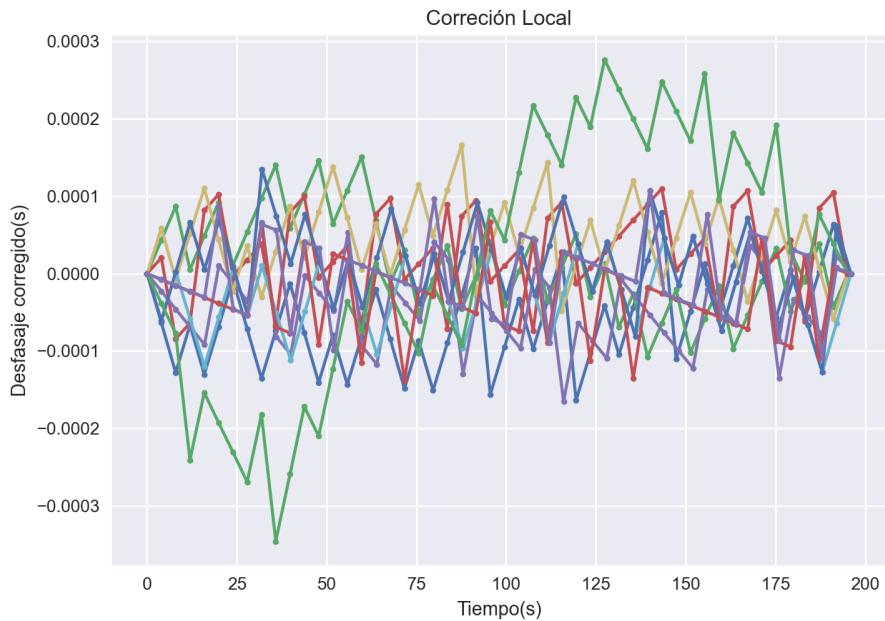


Figura 13: Desfases en función del tiempo para varias mediciones hechas con el par de Arduinos $a - c$ luego de corregir las mediciones con la pendiente local. En este caso se enviaron dos cuadradas espaciadas 200 s.

Como se ve en la figura 13, con la corrección local se logra al cabo de 200 s un desfase máximo de alrededor de 0.0002 s (en comparación a los 0.01 s de la corrección con la media). Esto equivale a aproximadamente 4 puntos en el peor de los casos y menos de 1 en muchos otros casos lo cual es un resultado muy satisfactorio dado que el error es ahora del orden de la inversa de la frecuencia de muestreo. Con este resultado se puede decir que se recupera la sincronía en las grabaciones. Se encuentra ahora un método apto para la corrección de mediciones y sincronía para ser utilizado en el sistema.

3.3. Integración del sistema

Una primera prueba para corroborar que la corrección de las grabaciones efectivamente funciona, consiste en grabar una fuente sonora en 2 dimensiones usando 2 micrófonos. Como se describe en más detalle en la sección 2.3, esta configuración permite estimar el ángulo de incidencia de la fuente. Se tiene una fuente colocada a 22.6° medido desde la linea que une ambos grabadores y centrado en uno de ellos. El ángulo calculado con este método es 21.7°.

Estos valores son muy similares considerando que además se está bajo la aproximación de campo lejano (la cuál no es enteramente valida en las condiciones del problema, ya que la distancia intergrabador es comparable con la distancia a la fuente). Esto muestra que la corrección de delays es válida y permite sobreponerse a los problemas de sincronización.

Como prueba final, para integrar la correcciones desarrolladas sobre las grabaciones y el trabajo hecho sobre el algoritmo, se realizan grabaciones en 2 dimensiones usando 3 micrófonos, lo cual es suficiente para estimar la localización de las fuentes. El detalle de este experimento está en la sección 2.3. Los errores obtenidos en las posiciones de las fuentes están entre 0.5 m y 1.2 m . Estos resultados son muy satisfactorios ya que se condicen en orden de magnitud con los errores de entrenamiento obtenidos para el MLP (error de aproximadamente 0.6 m para las últimas épocas). Para comparar relativamente el desempeño, se obtiene a partir de los delays temporales la localización de las fuentes usando de métodos de minimización. Se encuentra que estos presentan errores mucho mayores y variables. Es importante notar que el número de micrófonos usado para esta prueba es el mínimo que se necesita en la definición analítica del problema por lo que, el hecho de que el orden de magnitud de los errores fuera similar es un muy buen resultado. En el diseño total del dispositivo (para localización 3D) se usan 10 grabadores, más del doble de los necesarios analíticamente. Sabemos que la relación entre el número de micrófonos necesarios y los utilizados afecta fuertemente el desempeño, siendo que cuanto mayor sea la sobredeterminación del problema mejor será el resultado.

4. Conclusiones

Se concluye que la utilización de sistemas de deep learning es válida para este tipo de problemas, adaptándose mejor a los errores que los sistemas existentes. El desempeño del algoritmo depende fuertemente de las condiciones que se tengan en el problema. Tomando una frecuencia de muestro de 22 kHz para los micrófonos, se logra diseñar una distribución de 10 micrófonos en total, en dos dispositivos independientes e iguales de 5 micrófonos, que junto con el algoritmo puede cumplir con las especificaciones necesarias de acuerdo con las simulaciones realizadas. Estos dispositivos pueden, a futuro, usarse en conjunto de a un número mayor para lograr una localización exitosa en regiones de gran tamaño.

Se obtiene, usando una arquitectura de [64,64,64,64], un mediana en los errores de localización en datasets sintéticos de 0.47 m , teniendo el 99% de las fuentes localizadas con un error menor a 2.56 m . Estos valores pueden mejorarse con nuevos entrenamientos y desarrollos. Además el sistema presenta robustez frente a un aumento en los errores, por lo que aunque en pruebas de campo se encontraran errores mayores, no se esperaría una falla en el desempeño demasiado grande hasta errores en los delays de $\pm \frac{8}{22000}\text{ s}$.

Se concluye que, usando una plataforma Arduino Uno, empleando una frecuencia de 22 kHz , y realizando una corrección local, es posible realizar grabaciones en donde la perdida de la sincronía es consistente con los errores mínimos esperados. Con la corrección, se obtienen disminuciones experimentales del error en el defasaje entre grabaciones desde 1 s al cabo de 200 s , hasta 0.0002 s .

Se logra unificar el trabajo hecho en los grabadores con el de machine learning. Se realizan pruebas de concepto en donde se logra localizar una fuente sonora con error compatible con el esperado.

5. Futuro

Esta sección esta dedicada a las cosas que fuimos pensando e ideas que surgieron y no se llevaron a término durante el desarrollo del trabajo pero que creemos pueden ser útiles a futuro.

5.1. Corrección velocidad del sonido

Es importante notar que, a pesar de que la red se entrena con una velocidad del sonido particular, es posible usar el mismo entrenamiento para otra. Sabiendo que un delay temporal generado por una fuente esta dado por:

$$d = \frac{\|\Delta x\|}{c} \quad (7)$$

Donde $\|\Delta x\|$ es la diferencia de camino recorrido por el sonido para llegar a cada uno de los dos micrófonos y c es la velocidad del sonido en una situación particular. Si tenemos un entrenamiento de la red para esta velocidad c pero nuestra medición tiene una velocidad del sonido \bar{c} sabiendo que:

$$\bar{d} = \frac{\|\Delta x\|}{\bar{c}} \quad (8)$$

entonces podemos concluir que, si las condiciones del experimento hubieran tenido la velocidad del sonido original de entrenamiento c entonces, el delay temporal hubiera sido:

$$d = \frac{\bar{d}\bar{c}}{c} \quad (9)$$

Es importante notar que, en la práctica, debido a la existencia de errores, \bar{d} en verdad será $\bar{d} + \delta\bar{d}$. Por lo que:

$$d = (\bar{d} + \delta\bar{d})\frac{\bar{c}}{c} = d_{real} + (\delta\bar{d})\frac{\bar{c}}{c} \quad (10)$$

Donde d es el delay que se obtiene transformando la velocidad del sonido y d_{real} es el que se hubiera obtenido si originalmente no hubiera habido error. Es importante ver que el error temporal teórico **no depende de la velocidad del sonido** sino de la frecuencia de muestreo. Con lo cual, entrenando con una velocidad del sonido relativamente alta, el error efectivo luego de la transformación en caso de tener mediciones a menor temperatura es menor. En nuestro trabajo se usó como velocidad del sonido $350 \frac{m}{s}$, correspondiéndose con la que existe en un ambiente a $32C$ lo cuál es relativamente alto. Se obtiene entonces que para toda temperatura menor el desempeño del algoritmo es mejor.

5.2. Datos de entrenamiento

En este trabajo se usó como entrenamiento para el algoritmo puntos ubicados aleatoriamente con una distribución uniforme en una esfera de $50m$ de radio alrededor del dispositivo. En la práctica, se podría restringir la esfera de entrenamiento para un área de interés particular, pudiendo suponer esto una mejora para el rendimiento del algoritmo. Un ejemplo paradigmático podría ser entrenarlo en la semiesfera superior si el dispositivo se encuentra en el suelo.

5.3. Datos de entrenamiento

En caso de tener errores más altos que lo deseado, sería posible usar machine learning para poder superar estos problemas. Para eso, hay dos opciones que podrían ser útiles. Se presentan a continuación los paradigmas para esto. Ambos se basan en el supuesto de que no cualquier relación entre los delays temporales de entrada es compatibles con uno que podría ocurrir en la realidad, sino que formarían un manifold en la dimensión total de los delays.

5.3.1. Perceptrón multicapa

Se podría entrenar un preceptuaron multicapa para dar una noción de la confianza de cierto juego de delays temporales. Esto permitiría estimar cuan confiable es a su vez el resultado de la red, evitando outliers que pudieran afectar las mediciones.

A su vez, se podrían proponer variaciones aleatorias sobre los delays medidos buscando un juego de diferencias temporales cercano al original pero que posea una confianza mayor.

Para hacer esto, dada una arquitectura del MLP se debería realizar un entrenamiento supervisado, categorizando cada juego de delays como uno posible o no. Para ello se podría tomar como input un juego de delays que tuviera módulo de diferencia con los errores reales de δ . Luego, normalizando con $\max(\delta)$ se obtendría para cada uno de los delays un $\bar{\delta}$ como etiqueta. De esta forma 0 sería un juego de delays sin error y 1 uno con error máximo. Idealmente este error máximo debería estar fuera de lo aceptado por la red. Sería necesario estudiar previamente el desempeño de la red en función de δ de una forma mas completa para poder elegir el rango para el entrenamiento.

5.3.2. Autoencoder

La segunda posibilidad sería utilizar un autoencoder para quitar la corrupción de los delays. Para ellos se tendría que usar como entrenamiento un juego de delays corruptos y usar como target los reales. De ese modo se podría tener un nuevo juego de delays a partir de los originales que tengan un error total menor. Esto podría formar parte del pipeline de funcionamiento del sistema en caso de que el rendimiento fuera bueno y los errores típicos encontrados experimentalmente lo ameriten.

6. Automatización de medición de delays temporales

Para la medición a partir de las señales de los delays temporales se pensó como posibilidad el uso de cross-correlation. Para ello se necesitaría buscar una sección de interés en una señal particular y luego encontrar su posición

análoga para todos los demás micrófonos.

7. Señal de referencia a distancia

Debido a que los dispositivos presentados en el diseño son unidades independientes físicamente, sería necesario para que tuvieran una referencia temporal conjunta que ambos estuvieran cableados lo cual puede no ser práctico para su uso en el campo. Para ello se pensó en agregar a cada uno de los dispositivos un módulo wifi dedicado a comunicarse con los demás marcando para la señal de referencia. La suposición clave en este caso es el hecho de que la velocidad típica de envío de señales wifi es mucho más rápida que la del sonido y podría considerarse casi instantánea.

8. Códigos

Para ver los códigos que se usaron tanto para el desarrollo del algoritmo (creación de datasets, entrenamiento y testeo de arquitecturas) como para el manejo de las señales de los grabadores, ir al siguiente github: <https://github.com/julianamette/laboratorio7>

9. Agradecimientos

Se agradece a Gustavo Maurin por la colaboración en el diseño del dispositivo, pensando en su uso y futura instalación en el campo.

Se agradece a todo el laboratorio de sistemas dinámicos, en especial a Gabo quien nos dió el espacio y confió en nosotros para realizar este proyecto; y a Roberto quién no solo se ocupó de realizar los experimentos que diseñamos en este contexto tan particular que es finalizar laboratorio 7 en medio de una pandemia (COVID-19), sino que nos acompañó en todo el trayecto (al cual no le faltaron problemas).

Agradecemos también a nuestrxs amigxs, cuya influencia siempre resulta positiva para la realización de este tipo de proyectos.

A. Apéndice: Decisión de disposición de micrófonos

A continuación se presentarán brevemente las opciones de diseño del dispositivo. Usando un algoritmo de descenso por gradiente para la localización basada en TDOA, se explorarán los errores asociados a distintas configuraciones de micrófonos. Estudiando las dependencias de estos errores con los diferentes parámetros del dispositivo (cantidad de micrófonos y disposición de los mismos), se decide cuáles son las configuraciones más favorables. Sin embargo, también se deben considerar algunas restricciones impuestas por las características del problema, como son la portabilidad del dispositivo y su precio.

A.1. Algoritmo de descenso por gradiente

Dados n micrófonos, cada uno de ellos en una posición \mathbf{x}_i , la fuente de la señal a detectar en \mathbf{p} , y c la velocidad del sonido en el medio; entonces la diferencia de tiempo de arribo de la señal entre un micrófono \mathbf{x}_i y otro \mathbf{x}_j esta dada por:

$$\|\mathbf{x}_i - \mathbf{p}\| - \|\mathbf{x}_j - \mathbf{p}\| = c \cdot \Delta t_{i,j} \quad (11)$$

donde $\Delta t_{i,j} = t_i - t_j$.

El algoritmo consiste en minimizar la función costo C :

$$C(\mathbf{p}) = \sum_i \sum_{j>i} (\|\mathbf{x}_i - \mathbf{p}\| - \|\mathbf{x}_j - \mathbf{p}\| - c \cdot \Delta t_{i,j})^2 \quad (12)$$

Se utiliza esa función para que el algoritmo sea lo menos sensible posible a cambios particulares en los pares de micrófonos.

El método de minimización utilizado es descenso por gradiente. Este es un método iterativo. Dada una condición inicial \mathbf{X}_0 se calcula el estado \mathbf{X}_i de la minimización como:

$$\mathbf{X}_i = \mathbf{X}_{i-1} - lr \cdot \nabla C(\mathbf{p}) \quad (13)$$

donde lr es el coeficiente de aprendizaje.

El algoritmo consiste en dos partes. Primero se utiliza $lr = 2$ durante 2 iteraciones y luego se toma un coeficiente de aprendizaje adaptativo dado por:

$$lr_i = \frac{|(\mathbf{X}_i - \mathbf{X}_{i-1})^T [\nabla C(\mathbf{X}_i) - \nabla C(\mathbf{X}_{i-1})]|}{\|\nabla C(\mathbf{X}_i) - \nabla C(\mathbf{X}_{i-1})\|^2} \quad (14)$$

Esto se repite hasta llegar a un total de 300 iteraciones.

A través de análisis preliminares con este método, no se consiguieron resultados aceptables (entiéndase por aceptable, alrededor de un metro de error en mediana) para un solo dispositivo de tamaño portable. Por lo tanto, el diseño del prototipo consiste en dos dispositivos, cada uno con un arreglo de micrófonos propio: se llaman \mathbf{x}_i^1 a los micrófonos del dispositivo 1 y \mathbf{x}_i^2 a los micrófonos del dispositivo 2. El centro de cada uno de los dispositivos se encuentra en las posiciones \mathbf{r}_1 y \mathbf{r}_2 . Se ejecuta entonces el algoritmo mencionado anteriormente para cada uno de los arreglos por separado y se normaliza el vector obtenido con la minimización de la función C . Estos vectores v_1 y v_2 indican las direcciones en que, según el algoritmo, se encuentra la fuente con respecto al dispositivo 1 y 2 respectivamente. Una vez realizado este cálculo se buscan los puntos más cercanos entre las rectas (escritas de forma paramétrica):

$$\mathbf{L}_1(\alpha) = \mathbf{r}_1 + \alpha \cdot \mathbf{v}_1 \quad (15)$$

$$\mathbf{L}_2(\beta) = \mathbf{r}_2 + \beta \cdot \mathbf{v}_2 \quad (16)$$

La recta \mathbf{L}_1 une el centro del primer dispositivo con el punto donde es mínima la función costo de este mismo dispositivo. \mathbf{L}_2 se construye de forma análoga pero para el segundo dispositivo. Se quieren los puntos más cercanos entre ambas rectas (uno por cada recta) ya que, idealmente la fuente se ubicaría exactamente en la intersección entre ellas. Dado que existen errores experimentales, esto no ocurre. Sin embargo, se espera que el punto medio de la recta que une estos dos puntos sea una buena aproximación a la fuente real. Para encontrar estos puntos, se debe minimizar el módulo al cuadrado de la diferencia entre ambas rectas. Es decir que se quiere encontrar valores de α y β tal que $|\mathbf{L}_1(\alpha) - \mathbf{L}_2(\beta)|^2$ sea mínimo. Para ello, primero se escribe esta expresión explícitamente, luego se la deriva separadamente con respecto a α y a β , y por último se pide que estas derivadas sean nulas. Es decir que se calcula lo siguiente:

$$\frac{\partial}{\partial \alpha} (|\mathbf{L}_1(\alpha) - \mathbf{L}_2(\beta)|^2) = 0 \quad (17)$$

$$\frac{\partial}{\partial \beta} (|\mathbf{L}_1(\alpha) - \mathbf{L}_2(\beta)|^2) = 0 \quad (18)$$

Estas dos expresiones pueden escribirse de forma matricial. Resolviendo este sistema lineal, se consiguen los valores de α y β deseados. Reemplazándolos en las expresiones de \mathbf{L}_1 y \mathbf{L}_2 , se obtienen los puntos más cercanos entre las rectas. Por último, se toma el punto medio entre de ambas como la posición de la fuente.

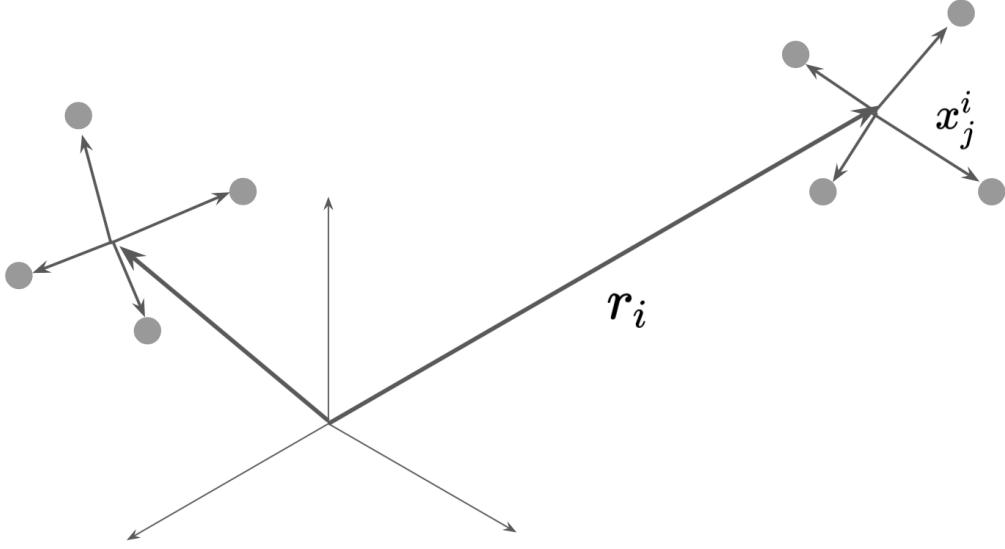


Figura 14: Esquema del dispositivo

A.2. Propuestas de diseño y errores

Con el objetivo de analizar como afectaban los errores experimentales a la posición encontrada de la fuente (mediante el algoritmo de descenso por gradiente) en comparación con la posición real, se realizaron varias simulaciones. Los errores experimentales considerados fueron los de la frecuencia de muestreo, la cual se tomó como 22 kHz . Entonces, cada tiempo t_i medido tenía algún error dentro del intervalo $\pm 1/22000\text{ s}$. Para hacer el estudio de errores se toma un error aleatorio dentro del intervalo mencionado, con probabilidad uniforme.

En cada simulación se fijo la distancia entre ambos dispositivos y la configuración de cada uno (se analizaron distintas configuraciones de micrófonos para los dispositivos individuales). Se simularon varias fuentes dentro de un volumen esférico de radio 50 m centrado en el punto medio entre ambos dispositivos. Cada fuente se ubicó en un lugar aleatorio dentro de este volumen. Se calcularon los tiempos de llegada del sonido a cada micrófono (t_i), utilizando como velocidad del sonido $350\frac{\text{m}}{\text{s}}$, lo cual corresponde a aproximadamente 32°C de temperatura. Con estos datos, se buscó la posición de cada una de las fuentes por separado, como se indica en la sección A.1. Todo esto se hizo con el objetivo de analizar la estadística de los errores en función del tamaño de cada dispositivo, su configuración y la distancia entre ambos.

A.3. Gráficos comparativos

Una vez hechas todas las simulaciones, se buscaron relaciones entre los errores y el diseño del aparato de medición (tamaño de los dispositivos y su separación). En la sección A.4 se detallan todas las configuraciones estudiadas.

En primer lugar se estudió la dependencia de los errores con la separación entre dispositivos. Fijando los dispositivos con 100 cm de radio y usando la configuración 007 (ver sección A.4), en la figura 15 se muestra como varían los porcentajes de fuentes con error menor a 2 m y menor a 1 m en función de la separación.

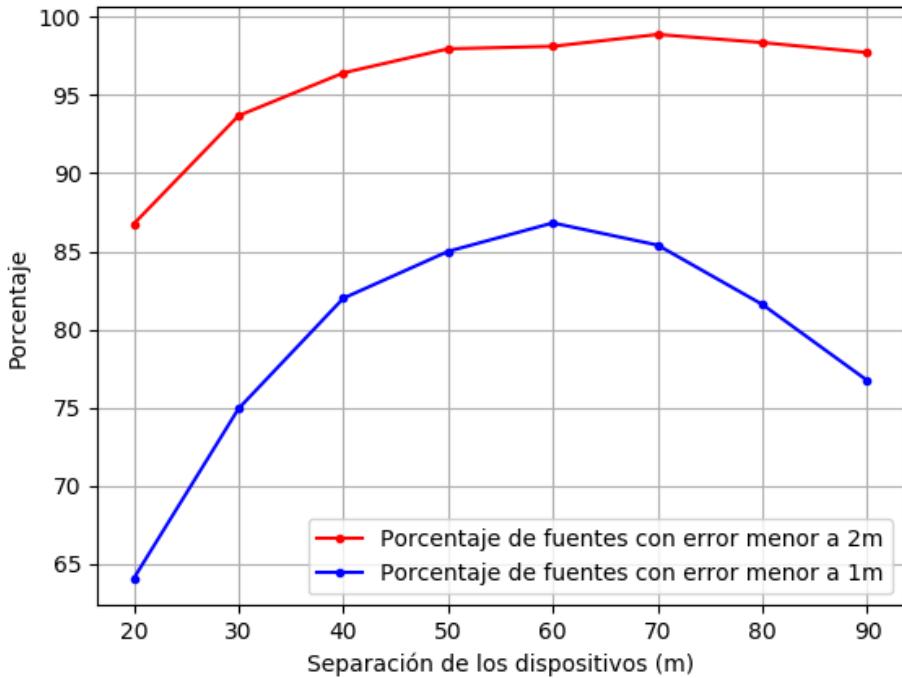


Figura 15: Simulación con dispositivos de 100 cm de radio en la configuración 007, utilizando 5000 fuentes. Se muestra la dependencia de los errores con la separación de los dispositivos.

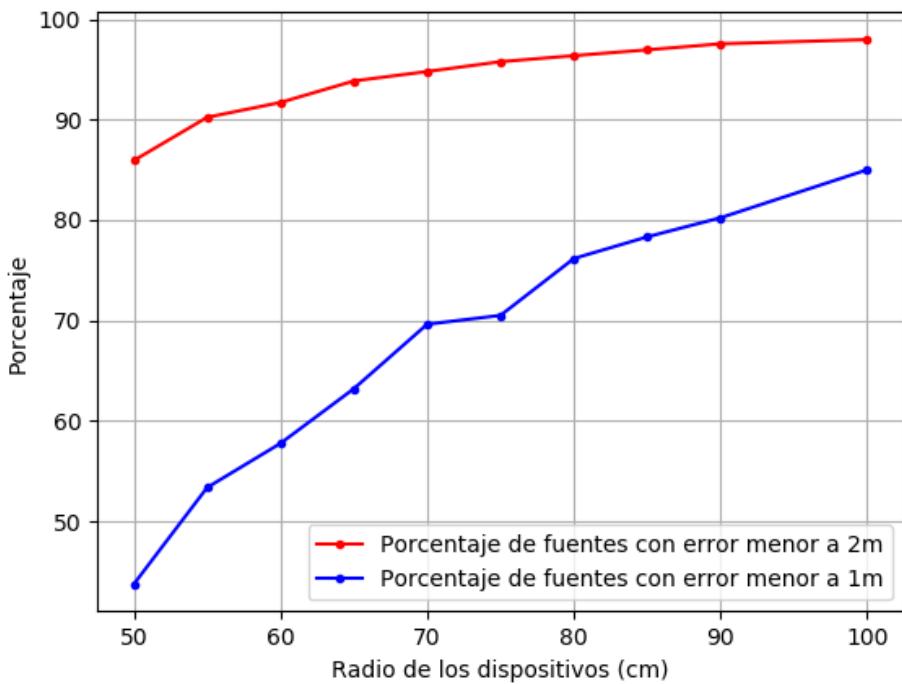


Figura 16: Simulación con dispositivos a 50 metros de separación en la configuración 007, utilizando 5000 fuentes. Se muestra la dependencia de los errores con el radio de los dispositivos.

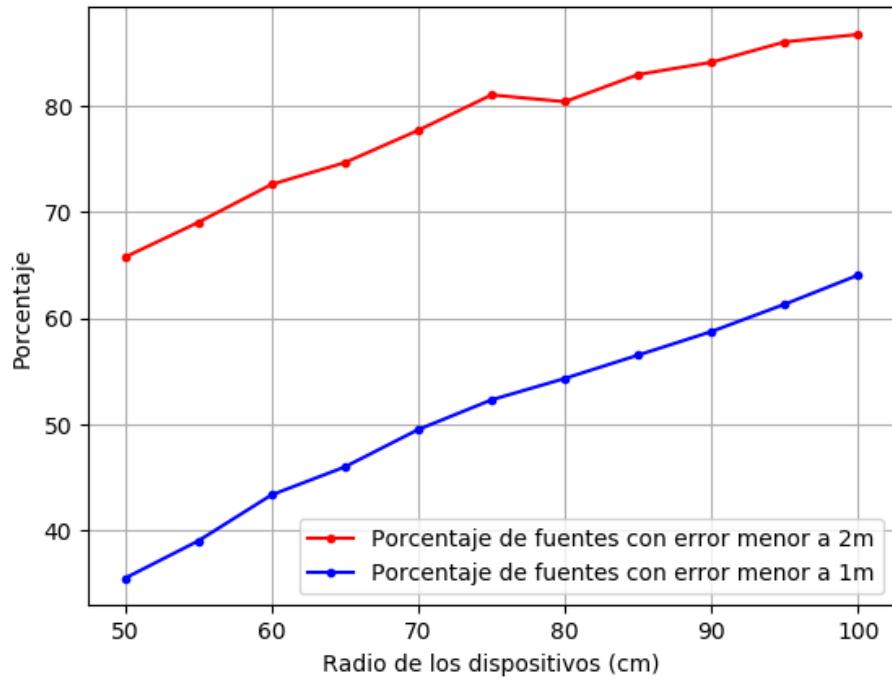


Figura 17: Simulación con dispositivos a 20 metros de separación en la configuración 007, utilizando 5000 fuentes. Se muestra la dependencia de los errores con el radio de los dispositivos.

Como se ve en la figura 15, al aumentar la separación entre dispositivos el error disminuye, hasta llegar a una separación crítica, a partir de la cual el error vuelve a empeorar. Dado un tamaño de dispositivo se encuentra entonces una distancia inter-dispositivo en la cual el error es mínimo.

Con respecto a la dependencia de los errores con el radio de los dispositivos, para el caso de 50 m y 20 m de distancia inter-dispositivo, los resultados se muestran en las figuras 16 y 17 respectivamente. En ambos casos se ve que a mayor radio de los dispositivos, el error disminuye. No se estudiaron radios mayores a 100 cm ya que este se considera como el límite máximo de tamaño deseable para mantener la portabilidad.

A.3.1. Comparación de configuraciones

Se compararon los errores de cada una de las configuraciones. En la figura 18 se muestran los errores obtenidos variando el radio de los dispositivo mientras la separación entre ellos permanece constante (20 m) para la configuración 006, la 007 y la 008.

Es claro al ver la figura 18 que, para una separación entre dispositivos de 20 m, la configuración 008 es la que posee los mejores resultados en todo el rango. Sin embargo, esta configuración implica tener dos dispositivos de 6 micrófonos cada uno, es decir, 12 micrófonos en total. Esta cantidad es muy elevada debido a los costos que implica. Es por esto que se descarta la configuración 008 y se elige la 007. Esta última corresponde a la siguiente configuración con mejores resultados.

En conclusión, mediante un análisis de todas las variables involucradas en el diseño del dispositivo (cantidad de micrófonos, disposición de los mismo, tamaño de cada dispositivo, y separación entre los mismos) usando el algoritmo de descenso por gradiente, se concluye que dos dispositivos de radio 1 m en la configuración 007 es la mejor opción para los requisitos del problema planteado.

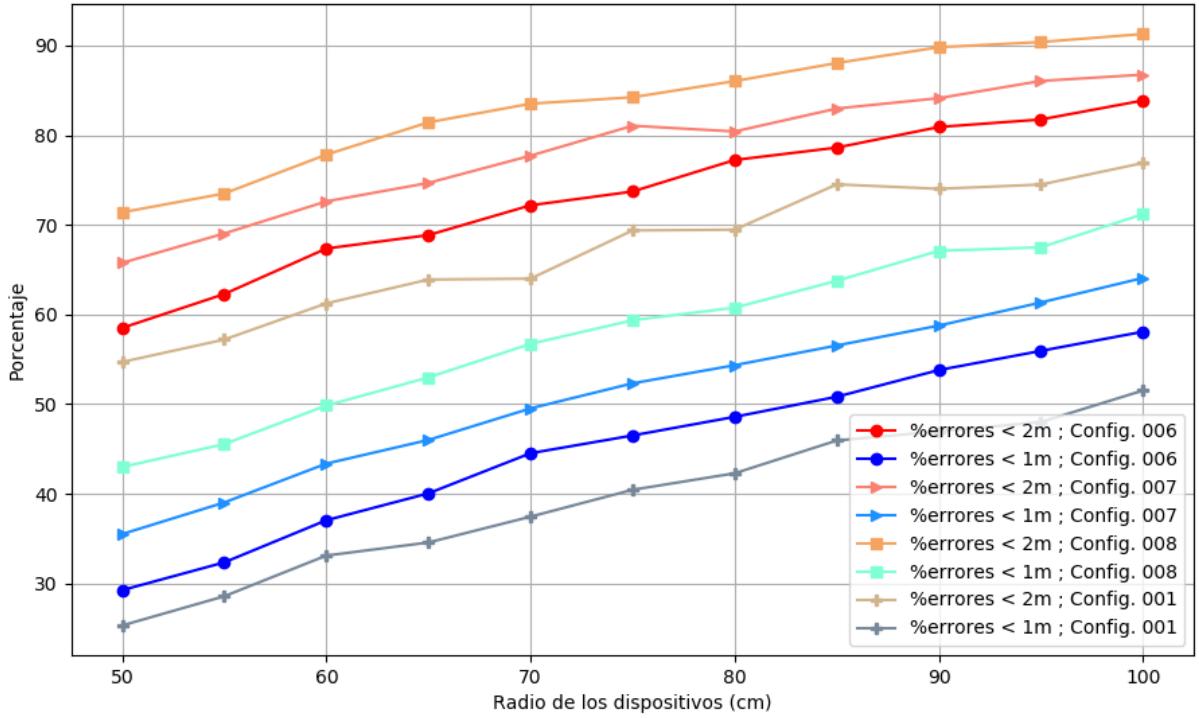


Figura 18: Comparación de los errores (tanto el porcentaje de fuentes con error menor a 2 m, como el de fuentes con error menor a 1 m) para distintas configuraciones. Simulaciones con dispositivos a 20 metros de separación, utilizando 5000 fuentes.

A.4. Configuraciones

A.4.1. Configuración 001

La configuración 001 tiene 4 micrófonos. Dado un radio del dispositivo r_d , las coordenadas (dadas en esféricas) de los micrófonos son las siguientes:

Número Micrófono	θ	ϕ	r
1	0	$\frac{\pi}{2}$	r_d
2	$\frac{2\pi}{3}$	$\frac{\pi}{2}$	r_d
3	$\frac{2\pi}{3}$	$\frac{\pi}{2}$	r_d
4	-	-	0

A.4.2. Configuración 006

La configuración 006 tiene 5 micrófonos. Dado un radio del dispositivo r_d , las coordenadas (dadas en esféricas) de los micrófonos son las siguientes:

Número Micrófono	θ	ϕ	r
1	0	$\frac{\pi}{2}$	r_d
2	$\frac{2\pi}{3}$	$\frac{\pi}{2}$	r_d
3	$\frac{2\pi}{3}$	$\frac{\pi}{2}$	r_d
4	-	-	0
5	-	0	r_d

A.4.3. Configuración 007

La configuración 007 tiene 5 micrófonos. Dado un radio del dispositivo r_d , las coordenadas (dadas en esféricas) de los micrófonos son las siguientes:

Número Micrófono	θ	ϕ	r
1	0	$\frac{\pi}{2}$	r_d
2	$\frac{\pi}{2}$	$\frac{\pi}{2}$	r_d
3	π	$\frac{\pi}{2}$	r_d
4	$-\frac{\pi}{2}$	$\frac{\pi}{2}$	r_d
5	-	0	r_d

A.4.4. Configuración 008

La configuración 008 tiene 6 micrófonos. Dado un radio del dispositivo r_d , las coordenadas (dadas en esféricas) de los micrófonos son las siguientes:

Número Micrófono	θ	ϕ	r
1	0	$\frac{\pi}{2}$	r_d
2	$\frac{\pi}{2}$	$\frac{\pi}{2}$	r_d
3	π	$\frac{\pi}{2}$	r_d
4	$-\frac{\pi}{2}$	$\frac{\pi}{2}$	r_d
5	-	0	r_d
6	-	π	r_d

B. Apéndice: Diseño del dispositivo

El diseño del dispositivo donde se colocarán cada uno de los dos grupos de 5 micrófonos, fue realizado en colaboración con el diseñador industrial Gustavo Maurin. El mismo puede verse en la figura 19. Es un dispositivo desplegable con estacas en la base, lo cual lo hace fácil de portar y armar. Cuatro de los cinco micrófonos se ubican en los brazos inferiores, y el quinto en la punta. Además, el dispositivo cuenta con un nivel para alineararlo en el momento de la instalación.

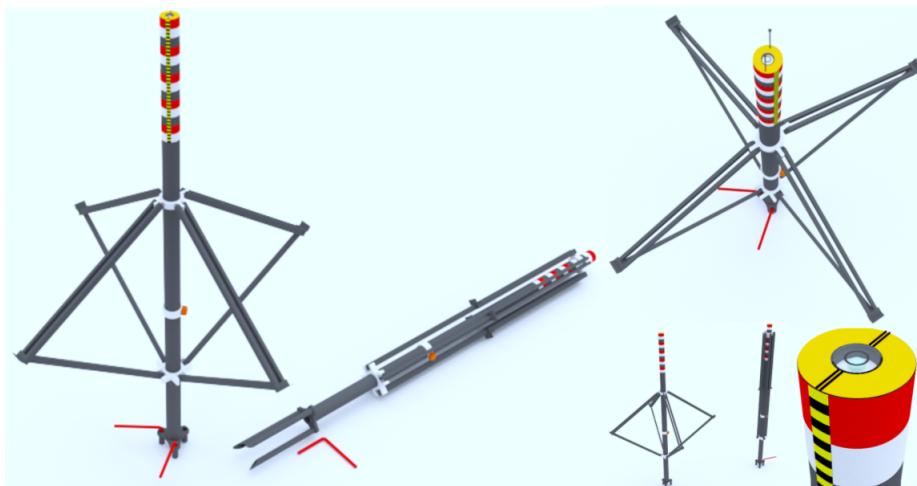


Figura 19: Diseño del dispositivo para portar un grupo de 5 micrófonos.

C. Apéndice: Importancia relativa de los pesos de la primera capa del MLP

Dado que el diseño posee 10 micrófonos, se tiene un total de 45 delays temporales entre receptores. Para analizar la importancia relativa de cada uno de ellos, se pueden ver los pesos de la primera capa oculta $W_{j,i}^0$. Estos se en forma de matriz en forma de matriz. La fila j representa el peso que tiene la neurona j con cada uno de los delays i . Graficando esta matriz en módulo, se obtiene lo que se puede ver en la figura (20). Se muestra, sin pérdida de la generalidad, el caso particular de la arquitectura [64,64,64,64]. Se observa que, sin importar la arquitectura, los resultados eran los mismos. Existe la presencia de franjas verticales con valores de bajo módulo, lo cual indicaría que esos delays son 'poco importantes' para las neuronas de entrada y sus valores son despreciados frente a otros. Los delays que de bajo módulo siempre son los mismos, sin importar el entrenamiento.

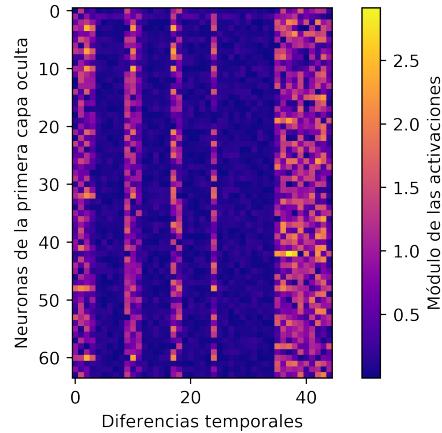


Figura 20: Matriz del módulo de las activaciones correspondiente a la arquitectura [64,64,64,64] entrenada durante 4500 épocas con un dataset de 30.000 ejemplos.

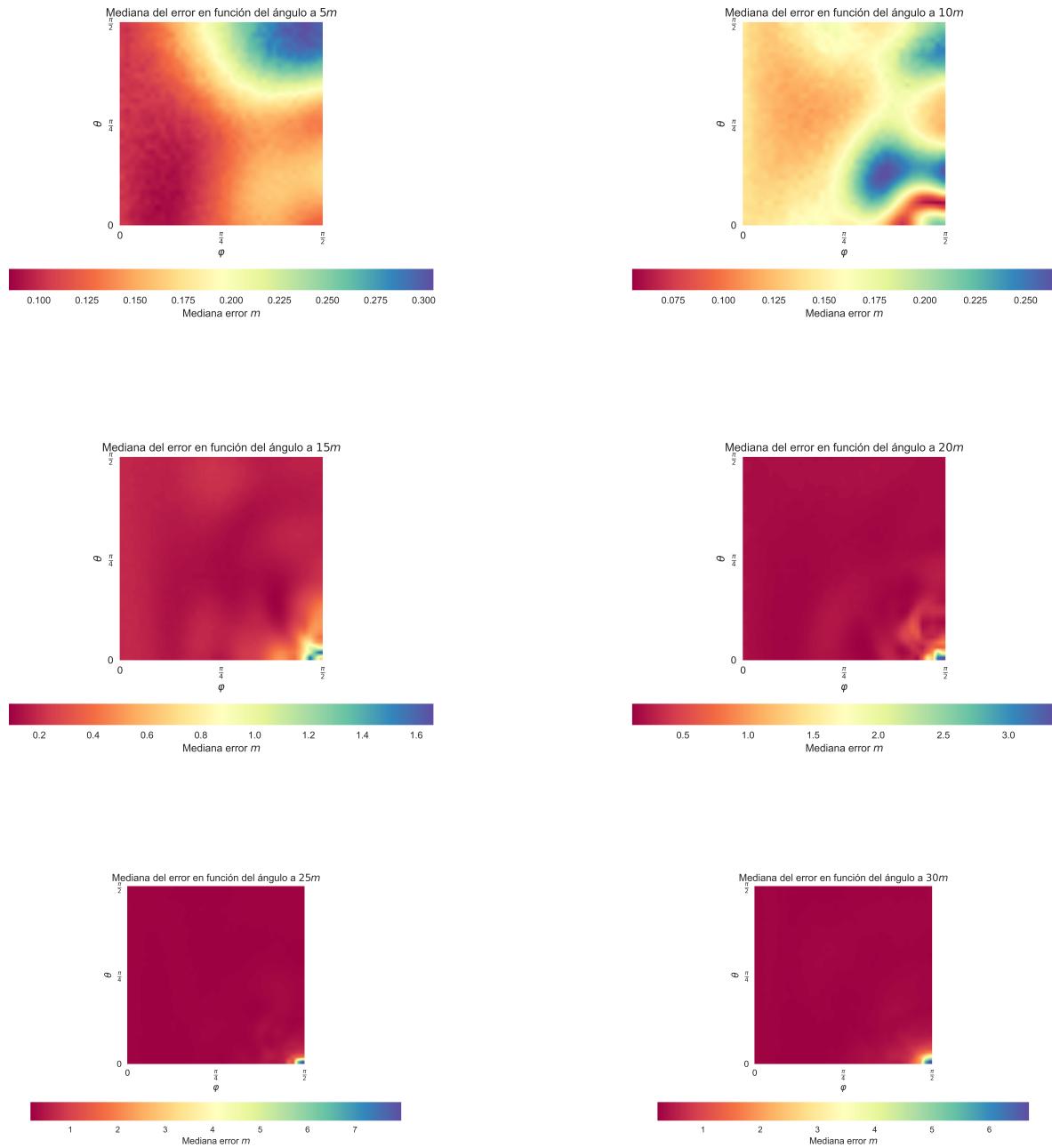
Se prueba entonces quitar los delays aparentemente despreciables y entrenar el perceptron para ver si hay un cambio en el desempeño. Si mejoran los resultados de testeo, se podría concluir que estos no son necesarios para determinar el problema. Al realizar esto se obtiene lo que se muestra en el cuadro 1.

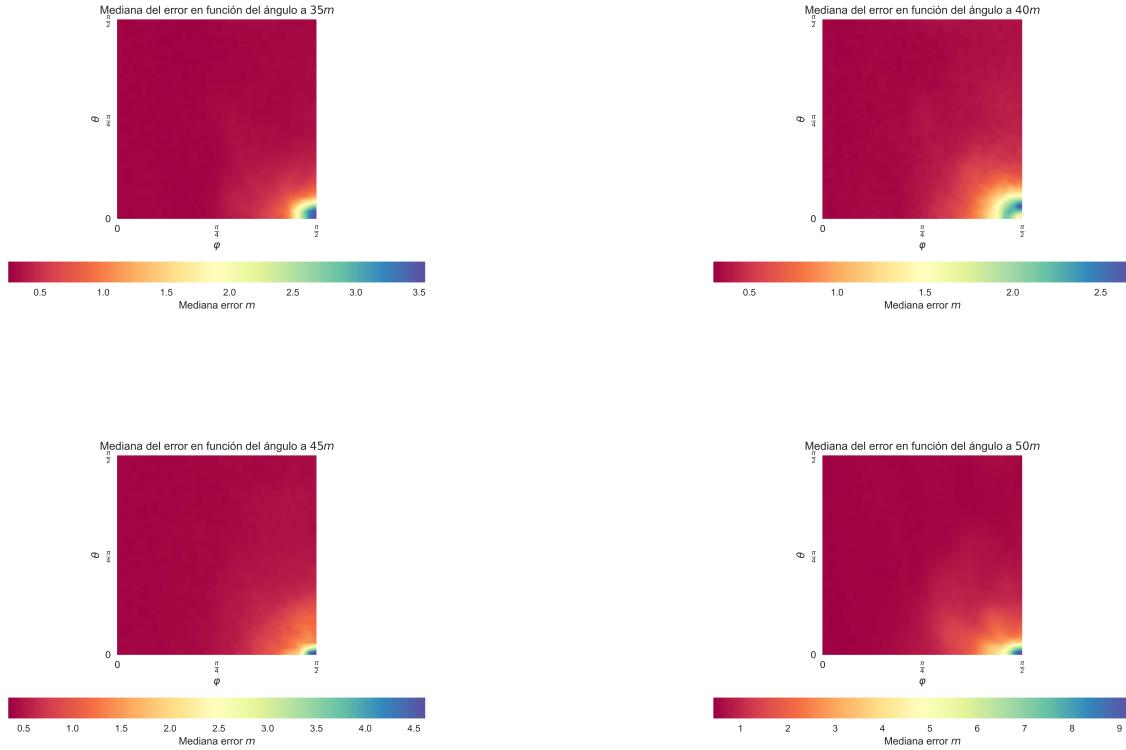
	Usando todos los delays	Sacando delays
Media	0.92 m	1.01 m
Mediana	0.61 m	0.68 m
% < 1,5 m	87.00	84.47

Cuadro 1: Estudio de los errores de testeo para la arquitectura [64, 64, 64, 64]. En ambos casos se usaron las arquitecturas entrenadas luego de 1500 épocas con el mismo dataset, y como testeo otro de 500.000 ejemplos.

Se observa que quitar delays temporales empeora la performance del algoritmo, con lo cual se concluye que las diferencias temporales de valores altos en la matriz de pesos son los que se utilizan para aproximar la posición a primer orden, y los que tienen pesos menores se usan como correcciones para hacer el sistema más eficiente y exacto.

D. Apéndice: Análisis angular de los errores del MLP





Referencias

- [1] Joseph H. DiBiase, Harvey F. Silverman, Michael S. Brandstein, *Robust localization in Reverberant Rooms*, Brown University, Providence RI, USA , Harvard University, Cambridge MA, USA.
- [2] Lubna Farhi, *Dynamic location estimation by Kalman Filter*, Engineering University, Karachi, Pakistan.
- [3] J. P. Dedieu, M. Shub, *Newton's method for overdetermined systems of equations*, 19 de Mayo de 1999.
- [4] Steven J. Spencer, *Closed-form analytical solutions of the time difference of arrival source location problem for minimal element monitoring arrays*, Luchas Heights Research Laboratory CSIRO Process Science and Engineering, Locked Bag 2005, Kirrawee, New South Wales 2232, Australia (2010).