UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Deconvolución acústica basada en filtros adaptativos y redes neuronales regresivas

Trabajo de graduación presentado por Carlos Manuel López Flores para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

2021

Deconvolución acústica basada en filtros adaptativos y redes neuronales regresivas

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Deconvolución acústica basada en filtros adaptativos y redes neuronales regresivas

Trabajo de graduación presentado por Carlos Manuel López Flores para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

2021

Vo.Bo.:

(f) _____

MSc. Miguel Enrique Zea Arenales

.

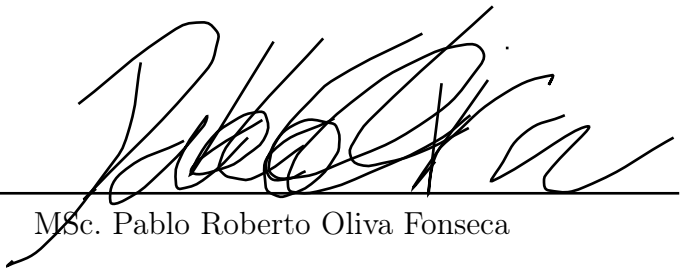
Tribunal Examinador:

(f) _____

MSc. Miguel Enrique Zea Arenales

(f) _____

MSc. Carlos Alberto Esquit Hernández



(f) _____

MSc. Pablo Roberto Oliva Fonseca

Fecha de aprobación: Guatemala, 22 de enero de 2021.

Prefacio

Primeramente agradezco a Dios, por darme paz en momentos de adversidad, por su infinito amor y por permitirme ver un nuevo amanecer cada día.

Luego a mis padres, Carlos López y Alma Flores, por sus consejos e instrucción a lo largo de mi vida. Por su trabajo y sacrificio, sin los cuales no hubiera podido llegar hasta este punto.

A mi asesor, MSc. Miguel Zea, por su constante guía y dedicación en cada una de las etapas de este proyecto.

A esta casa de estudios y demás catedráticos que me formaron a lo largo de esta carrera.

A la tía Sheny, tío Giovanni y Diego, por darme un hogar los primeros años de esta etapa universitaria.

A mis demás familiares y amigos, por inspirarme a ser una mejor persona cada día y quienes quiero un montón.

Índice

Prefacio v

Lista de figuras xv

Lista de cuadros xviii

Resumen xix

Abstract xxi

1. Introducción 1

2. Antecedentes 3
2.1. Primera fase . . . 3
2.1.1. Respuesta impulsional . . . 3
2.1.2. Identificación de sistemas . . . 3
2.2. Software comercial . . . 4
2.2.1. ERA 4 Reverb Remover de Accusonus . . . 4
2.2.2. Dialogue De-reverb de iZotope . . . 4
2.3. Intentos previos con redes neuronales . . . 4
2.3.1. Red neuronal TDNN . . . 4
2.3.2. Red neuronal CMAC . . . 5

3. Justificación 7

4. Objetivos 9
4.1. Objetivo general . . . 9
4.2. Objetivos específicos . . . 9

5. Alcance 11

6. Marco teórico 13
6.1. Variables aleatorias . . . 13
6.2. Valor esperado E{x} . . . 13

6.3. Procesos estocásticos . . . 14
6.4. Autocorrelación y correlación cruzada . . . 14
6.5. Filtros adaptativos . . . 14
6.5.1. Algoritmo LMS estándar . . . 15
6.5.2. Algoritmo LMS normalizado . . . 15
6.5.3. Algoritmo RLS convencional . . . 16
6.6. Redes neuronales artificiales (RNA) . . . 17
6.6.1. Teorema de aproximación universal . . . 18
6.6.2. Funciones de activación . . . 18
6.6.3. *Backpropagation* . . . 19
6.6.4. Función de costo . . . 20
6.6.5. Descenso por gradiente . . . 20
6.6.6. Redes neuronales regresivas . . . 21
6.6.7. TDNN (*Time delay neural network*) . . . 21
6.6.8. Redes neuronales *Focused Gamma* . . . 21
6.6.9. CMAC (*Cerebellar Model Articulation Controller*) . . . 22

7. Diseño experimental **25**
7.1. Equipo . . . 25
7.2. Respuesta en frecuencia . . . 25
7.3. Geometría y características de la habitación . . . 26

8. Metodología **29**
8.1. Grabación de las señales de audio . . . 29
8.2. Selección de las señales . . . 30
8.3. Implementación de los filtros adaptativos . . . 31
8.4. Implementación de las redes neuronales . . . 32

9. Filtros adaptativos **35**
9.1. LMS estándar . . . 35
9.1.1. Pruebas con señales determinísticas . . . 35
9.1.2. Pruebas con *clips* musicales . . . 40
9.2. LMS normalizado . . . 46
9.2.1. Pruebas con señales determinísticas . . . 46
9.2.2. Pruebas con *clips* musicales . . . 51
9.3. RLS convencional . . . 58
9.3.1. Pruebas con señales determinísticas . . . 58
9.3.2. Pruebas con *clips* musicales . . . 63
9.4. Discusión de resultados . . . 70
9.4.1. Evolución del ECM en el filtrado de señales determinísticas . . . 70
9.4.2. Evolución en la magnitud de $\mathbf{w}$ en el filtrado de señales determinísticas 70
9.4.3. Filtrado de los *clips* musicales . . . 71
9.4.4. Robustez del RLS y el LMS . . . 71
9.4.5. Orden del filtro y complejidad computacional . . . 72
9.4.6. Significado de la variación en $\mathbf{w}$ . . . 72

10. Redes neuronales **73**
10.1. TDNN . . . 73
10.1.1. Red neuronal por pista . . . 73
10.1.2. Red neuronal por lista de reproducción . . . 89
10.2. *Focused Gamma* . . . 94
10.2.1. Red neuronal por pista . . . 94
10.2.2. Red neuronal por lista de reproducción . . . 104
10.3. Discusión de resultados . . . 109
10.3.1. Investigación anterior . . . 109
10.3.2. Desempeño de las diferentes funciones de activación . . . 109
10.3.3. TDNN contra la *Focused Gamma* . . . 110
10.3.4. Redes con las pistas musicales . . . 110
10.3.5. Ruido . . . 111
10.3.6. Sobre el aprendizaje de las redes . . . 111
10.3.7. Redes neuronales contra los filtros adaptativos . . . 111
10.3.8. Posibles alternativas . . . 112

11. Conclusiones **113**

12. Recomendaciones **115**

13. Bibliografía **117**

14. Anexos **119**

15. Glosario **121**

Lista de figuras

1. Deconvolución obtenida por la TDNN para la señal de prueba $x(t)$. 5
2. Deconvolución de ruido blanco empleando una CMAC. 5
3. Deconvolución de un string de bits aleatorios empleando una CMAC. 5

4. Esquema general de un filtro adaptativo FIR. 15
5. Esquema de una red neuronal sencilla. 17
6. Esquema básico de una neurona artificial. 17
7. Diferentes funciones de activación. 19
8. Diagrama que muestra la relación funcional entre los nodos X, Y y Z. 20
9. Descenso gradiente ejemplificado en una función cuadrática. Al ser esta convexa, se garantiza un mínimo global[16]. 21
10. Red neuronal TDNN. 22
11. Red neuronal *Focused Gamma*. 22
12. Unidad recursiva $G(z)$ de la red neuronal *Focused Gamma*. 22
13. Red neuronal CMAC. 23

14. Respuesta en frecuencia del sistema. 26
15. Vista de planta de la habitación de prueba. 27

16. Diagrama simplificado del sistema de grabación. 29
17. Diagrama que muestra el proceso de filtrado. 31

18. ECM del filtrado de sinusoides por LMS estándar. 35
19. Magnitud del vector $\mathbf{w}$ del filtrado de sinusoides por LMS estándar. 36
20. ECM del filtrado de ondas cuadradas por LMS estándar. 36
21. Magnitud del vector $\mathbf{w}$ del filtrado de ondas cuadradas por LMS estándar. . . 36
22. ECM del filtrado de ondas diente de sierra por LMS estándar. 37
23. Magnitud del vector $\mathbf{w}$ del filtrado de ondas diente de sierra por LMS estándar. 37
24. Resultados del filtrado de un sinusoide de amplitud modulada por LMS estándar. 37
25. Resultados del filtrado de un sinusoide de frecuencia modulada por LMS estándar. 38
26. Resultados del filtrado de un sinusoide de amplitud y frecuencia modulada por LMS estándar. 38

27. Resultados del filtrado de un barrido sinusoidal (de 20 Hz a 10 kHz) por LMS estándar. 39
28. Señales del filtrado de clips de un solo instrumento por LMS estándar. 40
29. ECM del filtrado de clips de un solo instrumento por LMS estándar. 41
30. Magnitud del vector **w** del filtrado de clips de un solo instrumento por LMS estándar 41
31. Espectrogramas del filtrado de clips de un solo instrumento por LMS estándar. 42
32. Señales del filtrado de clips de varios instrumentos por LMS estándar. 43
33. ECM del filtrado de clips de varios instrumentos por LMS estándar. 44
34. Magnitud del vector **w** del filtrado de clips de varios instrumentos por LMS estándar. 44
35. Espectrogramas del filtrado de clips de varios instrumentos por LMS estándar. 45
36. ECM del filtrado de sinusoides por LMS normalizado. 46
37. Magnitud del vector **w** del filtrado de sinusoides por LMS normalizado. 46
38. ECM del filtrado de ondas cuadradas por LMS normalizado. 47
39. Magnitud del vector **w** del filtrado de ondas cuadradas por LMS normalizado. 47
40. ECM del filtrado de ondas diente de sierra por LMS normalizado. 47
41. Magnitud del vector **w** del filtrado de ondas diente de sierra por LMS normalizado. 48
42. Resultados del filtrado de un sinusoide de amplitud modulada por LMS normalizado. 48
43. Resultados del filtrado de un sinusoide de frecuencia modulada por LMS normalizado. 48
44. Resultados del filtrado de un sinusoide de amplitud y frecuencia modulada por LMS normalizado. 49
45. Resultados del filtrado de un barrido sinusoidal (de 20 Hz a 10 kHz) por LMS normalizado. 49
46. Señales del filtrado de clips de un solo instrumento por LMS normalizado. . . 51
47. ECM del filtrado de clips de un solo instrumento por LMS normalizado. . . . 52
48. Magnitud del vector **w** del filtrado de clips de un solo instrumento por LMS normalizado 52
49. Espectrogramas del filtrado de clips de un solo instrumento por LMS normalizado. 53
50. Señales del filtrado de clips de varios instrumentos por LMS normalizado. . . 54
51. ECM del filtrado de clips de varios instrumentos por LMS normalizado. . . . 55
52. Magnitud del vector **w** del filtrado de clips de varios instrumentos por LMS normalizado. 56
53. Espectrogramas del filtrado de clips de varios instrumentos por LMS normalizado. 57
54. ECM del filtrado de sinusoides por RLS convencional. 58
55. Magnitud del vector **w** del filtrado de sinusoides por RLS convencional. . . . 59
56. ECM del filtrado de ondas cuadradas por RLS convencional. 59
57. Magnitud del vector **w** del filtrado de ondas cuadradas por RLS convencional. 59
58. ECM del filtrado de ondas diente de sierra por RLS convencional. 60
59. Magnitud del vector **w** del filtrado de ondas diente de sierra por RLS convencional. 60
60. Resultados del filtrado de un sinusoide de amplitud modulada por RLS convencional. 60

61. Resultados del filtrado de un sinusoide de frecuencia modulada por RLS convencional. 61
62. Resultados del filtrado de un sinusoide de amplitud y frecuencia modulada por RLS convencional. 61
63. Resultados del filtrado de un barrido sinusoidal (de 20 Hz a 10 kHz) por RLS convencional. 62
64. Señales del filtrado de clips de un solo instrumento por RLS convencional. . . 63
65. ECM del filtrado de clips de un solo instrumento por RLS convencional. . . . 64
66. Magnitud del vector **w** del filtrado de clips de un solo instrumento por RLS convencional 64
67. Espectrogramas del filtrado de clips de un solo instrumento por RLS convencional. 65
68. Señales del filtrado de clips de varios instrumentos por RLS convencional. . . 66
69. ECM del filtrado de clips de varios instrumentos por RLS convencional. . . . 67
70. Magnitud del vector **w** del filtrado de clips de varios instrumentos por RLS convencional. 68
71. Espectrogramas del filtrado de clips de varios instrumentos por RLS convencional. 69

72. Costo por batch del entrenamiento de redes TDNN para sinusoides. 73
73. Señales generadas por las redes TDNN para sinusoides. 74
74. Espectrogramas generados por las redes TDNN para sinusoides. 74
75. Costo por batch del entrenamiento de redes TDNN para ondas cuadradas. . . 74
76. Señales generadas por las redes TDNN para ondas cuadradas. 74
77. Espectrogramas generados por las redes TDNN para ondas cuadradas. 75
78. Costo por batch del entrenamiento de redes TDNN para ondas diente de sierra. 75
79. Señales generadas por las redes TDNN para ondas diente de sierra. 75
80. Espectrogramas generados por las redes TDNN para ondas diente de sierra. . 76
81. Costo por batch del entrenamiento de redes TDNN para las señales AM, FM y AM-FM. 76
82. Salidas generadas por las redes TDNN para las señales AM, FM y AM-FM. . 76
83. Espectrogramas generados por las redes TDNN para las señales AM, FM y AM-FM. 77
84. Costo por batch del entrenamiento de redes TDNN para la señal chirp. 77
85. Salidas generadas por las redes TDNN para la señal chirp. 77
86. Espectrogramas generados por las redes TDNN para la señal chirp. 78
87. Costo por batch del entrenamiento de redes TDNN para sinusoides. 79
88. Señales generadas por las redes TDNN para sinusoides. 79
89. Espectrogramas generados por las redes TDNN para sinusoides. 79
90. Costo por batch del entrenamiento de redes TDNN para ondas cuadradas. . . 80
91. Señales generadas por las redes TDNN para ondas cuadradas. 80
92. Espectrogramas generados por las redes TDNN para ondas cuadradas. 80
93. Costo por batch del entrenamiento de redes TDNN para ondas diente de sierra. 80
94. Señales generadas por las redes TDNN para ondas diente de sierra. 81
95. Espectrogramas generados por las redes TDNN para ondas diente de sierra. . 81
96. Costo por batch del entrenamiento de redes TDNN para las señales AM, FM y AM-FM. 81
97. Salidas generadas por las redes TDNN para las señales AM, FM y AM-FM. . 81

98. Espectrogramas generados por las redes TDNN para las señales AM, FM y AM-FM. 82
99. Costo por batch del entrenamiento de redes TDNN para la señal chirp. 82
100. Salidas generadas por las redes TDNN para la señal chirp. 82
101. Espectrogramas generados por las redes TDNN para la señal chirp. 83
102. Costo por batch del entrenamiento de redes TDNN para sinusoides. 84
103. Señales generadas por las redes TDNN para sinusoides. 84
104. Espectrogramas generados por las redes TDNN para sinusoides. 84
105. Costo por batch del entrenamiento de redes TDNN para ondas cuadradas. . . 85
106. Señales generadas por las redes TDNN para ondas cuadradas. 85
107. Espectrogramas generados por las redes TDNN para ondas cuadradas. 85
108. Costo por batch del entrenamiento de redes TDNN para ondas diente de sierra. 85
109. Señales generadas por las redes TDNN para ondas diente de sierra. 86
110. Espectrogramas generados por las redes TDNN para ondas diente de sierra. . 86
111. Costo por batch del entrenamiento de redes TDNN para las señales AM, FM y AM-FM. 86
112. Salidas generadas por las redes TDNN para las señales AM, FM y AM-FM. . 86
113. Espectrogramas generados por las redes TDNN para las señales AM, FM y AM-FM. 87
114. Costo por batch del entrenamiento de redes TDNN para la señal chirp. 87
115. Salidas generadas por las redes TDNN para la señal chirp. 87
116. Espectrogramas generados por las redes TDNN para la señal chirp. 88
117. Evolución del costo por batch para la red TDNN con activación sigmoide. . . 89
118. Evolución del costo por batch para la red TDNN con activación tanh. 89
119. Evolución del costo por batch para la red TDNN con activación sinusoidal. . . 89
120. Señales del filtrado de clips de un solo instrumento por la TDNN con activación sinusoidal. 90
121. Señales del filtrado de clips de varios instrumentos por la TDNN con activación sinusoidal. 91
122. Espectrogramas del filtrado de clips de un solo instrumento por TDNN con activación sinusoidal. 92
123. Espectrogramas del filtrado de clips de varios instrumentos por TDNN con activación sinusoidal. 93
124. Costo por batch del entrenamiento de redes gamma para sinusoides. 94
125. Señales generadas por las redes gamma para sinusoides. 94
126. Espectrogramas generados por las redes gamma para sinusoides. 95
127. Costo por batch del entrenamiento de redes gamma para ondas cuadradas. . . 95
128. Señales generadas por las redes gamma para ondas cuadradas. 95
129. Espectrogramas generados por las redes gamma para ondas cuadradas. 96
130. Costo por batch del entrenamiento de redes gamma para ondas diente de sierra. 96
131. Señales generadas por las redes gamma para ondas diente de sierra. 96
132. Espectrogramas generados por las redes gamma para ondas diente de sierra. . 97
133. Costo por batch del entrenamiento de redes gamma para sinusoides. 98
134. Señales generadas por las redes gamma para sinusoides. 98
135. Espectrogramas generados por las redes gamma para sinusoides. 98
136. Costo por batch del entrenamiento de redes gamma para ondas cuadradas. . . 99
137. Señales generadas por las redes gamma para ondas cuadradas. 99
138. Espectrogramas generados por las redes gamma para ondas cuadradas. 99

139. Costo por batch del entrenamiento de redes gamma para ondas diente de sierra. 99
140. Señales generadas por las redes gamma para ondas diente de sierra. 100
141. Espectrogramas generados por las redes gamma para ondas diente de sierra. . 100
142. Costo por batch del entrenamiento de redes gamma para sinusoides. 101
143. Señales generadas por las redes gamma para sinusoides. 101
144. Espectrogramas generados por las redes gamma para sinusoides. 101
145. Costo por batch del entrenamiento de redes gamma para ondas cuadradas. . . 102
146. Señales generadas por las redes gamma para ondas cuadradas. 102
147. Espectrogramas generados por las redes gamma para ondas cuadradas. 102
148. Costo por batch del entrenamiento de redes gamma para ondas diente de sierra.102
149. Señales generadas por las redes gamma para ondas diente de sierra. 103
150. Espectrogramas generados por las redes gamma para ondas diente de sierra. . 103
151. Evolución del costo por batch para la red gamma con activación sigmoide. . . 104
152. Evolución del costo por batch para la red gamma con activación tanh. 104
153. Evolución del costo por batch para la red gamma con activación sinusoidal. . 104
154. Señales del filtrado de clips de un solo instrumento por la gamma con activación sinusoidal. 105
155. Señales del filtrado de clips de varios instrumentos por la gamma con activación sinusoidal. 106
156. Espectrogramas del filtrado de clips de un solo instrumento por gamma con activación sinusoidal. 107
157. Espectrogramas del filtrado de clips de varios instrumentos por gamma con activación sinusoidal. 108

Lista de cuadros

1. Diferentes señales determinísticas empleadas y los respectivos parámetro(s) empleados. . . . 30
2. Diferentes clips musicales empleados. El tiempo t hace referencia a los primeros segundos utilizados de la canción en donde se escuchan los instrumentos listados. . . . 31
3. Parámetros y orden de cada filtro al ser empleado con las distintas señales de prueba. . . . 31
4. Parámetros constantes durante el entrenamiento de las diferentes redes neuronales. . . . 33

5. ECM finales y máximas magnitudes de $\mathbf{w}$ alcanzados por el LMS estándar para señales determinísticas. . . . 39
6. ECM finales y máximas magnitudes de $\mathbf{w}$ alcanzados por el LMS estándar para los clips musicales. . . . 46
7. ECM finales y máximas magnitudes de $\mathbf{w}$ alcanzados por el LMS normalizado para señales determinísticas. . . . 50
8. ECM finales y máximas magnitudes de $\mathbf{w}$ alcanzados por el LMS normalizado para los clips musicales. . . . 58
9. ECM finales y máximas magnitudes de $\mathbf{w}$ alcanzados por el RLS convencional para señales determinísticas. . . . 62
10. ECM finales y máximas magnitudes de $\mathbf{w}$ alcanzados por el RLS convencional para los clips musicales. . . . 70

11. Errores promedios de la señales determinísticas por las redes TDNN con activación sigmoide. . . . 78
12. Errores promedios de la señales determinísticas por las redes TDNN con activación tanh. . . . 83
13. Errores promedios de la señales determinísticas por las redes TDNN con activación sinusoidal. . . . 88
14. Errores promedios de los clips musicales por las redes TDNN con activación sinusoidal. . . . 94
15. Errores promedios de la señales determinísticas por las redes gamma con activación sigmoide. . . . 97

16. Errores promedios de la señales determinísticas por las redes gamma con activación tanh. 100
17. Errores promedios de la señales determinísticas por las redes gamma con activación sinusoidal. 103
18. Errores promedios de los clips musicales por las redes gamma con activación sinusoidal. 109

Resumen

En este trabajo se implementaron diferentes redes neuronales y se compararon contra varios filtros adaptativos con el objetivo de eliminar efectos de convolución de un cuarto no tratado acústicamente. Ambos algoritmos se pusieron a prueba con un set de señales de audio de diferente grado de complejidad, tanto señales determinísticas como clips musicales. En la primera parte se implementaron los filtros adaptativos LMS estándar, LMS normalizado y RLS estándar. En la segunda parte se probaron las redes TDNN y *Focused Gamma*, cada una con las funciones de activación sigmoide, tanh y sinusoidal. Dichas arquitecturas fueron entrenadas con *backpropagation*. Se encontró que el mejor de los filtros fue el RLS estándar. En cuanto a las redes neuronales, estas lograron disminuir en cierto grado el ruido pero no lograron la fidelidad en audio del filtro antes mencionado.

Abstract

In this work, different neural network architectures were implemented and later compared with adaptive filters. This was done with the goal of eliminating the convolutive effects of a room that has not been acoustically treated. Both algorithms were tested by different audio signals which differ in complexity. Such signals included deterministic ones and musical clips. In the first section, the implemented adaptive filters were: standard LMS, normalized LMS and conventional RLS. In the second part, the neural networks TDNN and Focused Gamma were combined each one with three activation functions: sigmoid, tanh and sine. All neural networks were trained under the backpropagation algorithm. The results showed that conventional RLS produced the best performance of the three filters. On the other hand, the neural networks diminished background noise to a certain degree but they didn't accomplish the audio fidelity by the aforementioned adaptive filter.

CAPÍTULO 1

Introducción

El proyecto que a continuación se presenta propone generar deconvolución acústica mediante la combinación de filtros adaptativos y redes neuronales. Este proyecto beneficiará a músicos amateurs que producen desde casa, para que así logren una calidad profesional de audio a un costo asequible. Se continuará a partir de la primera fase ya desarrollada, en la cual se probaron diferentes algoritmos de deconvolución. Se evaluará el desempeño de los diferentes filtros adaptativos y se analizará el comportamiento de los mismos ante señales deterministas y estocásticas. Una vez realizadas las primera pruebas se propone poner a prueba las redes TDNN (Time Delay Neuronal) y *Focused Gamma* ante las mismas señales.

CAPÍTULO 2

Antecedentes

2.1. Primera fase

En la primera fase de este proyecto se experimentó con diversas métodologías de deconvolución tales como las de respuesta impulsional, identificación de sistemas y algoritmos LMS. Varios de estos se apoyaron de funciones propias de Matlab para la obtención de resultados. Los algoritmos LMS puestos a prueba fueron el convencional y el de Volterra. En efecto, fueron estos dos últimos los que generaron mejores resultados en contraste con los primeros métodos.

2.1.1. Respuesta impulsional

Estos métodos se basan en una respuesta al impulso $h[n]$, la cual caracteriza el ambiente de grabación. Esta respuesta impulsional, en convolución con una señal $x[n]$, genera como resultado $y[n]$. Lo que se realizó fue obtener esta última señal, la cual consiste del audio original modificado por el efecto acústico del cuarto. Después se procedió a generar $h[n]$ al grabar dentro del recinto un sonido impulsional, tal como un aplauso. Una estrategia utilizada fue implementar la función de Matlab *deconv*, teniendo como parámetros las señales grabadas $y[n]$ y $h[n]$. La segunda estrategia consistió en pasar las señales al dominio de la frecuencia y obtener la salida mediante una simple división $X[Z] = Y[Z]/H[Z]$.

2.1.2. Identificación de sistemas

Estas pruebas se realizaron con la herramienta de identificación de sistemas de Matlab. Para esta función se necesita tanto de la entrada $x[n]$ como de la salida $y[n]$ para así obtener la función de transferencia deseada. Esta función de identificación además da al usuario diferentes parámetros adicionales tales como el porcentaje de la aproximación y la cantidad de polos y ceros. En este caso después de haber obtenido la función de transferencia, lo

siguiente fue obtener el inverso de la misma. De la expresión invertida se extrajeron los coeficientes y por último se insertaron como parámetros en la función *filter* de Matlab.

2.2. Software comercial

2.2.1. ERA 4 Reverb Remover de Accusonus

Reverb Remover de Accusonus es un plug-in que logra eliminar el efecto de reverberación acústica. Este plug-in estima el perfil de reverberación, para luego permitir al usuario definir la magnitud de este efecto sobre la señal de audio a procesar. La interfaz da al usuario la opción de seleccionar en qué parte del espectro se desea aplicar el plug-in. Permite definir frecuencias bajas, medias, altas o todo el espectro de frecuencias presente en el audio. Este también ofrece un ajuste en la ganancia, ya que la desreverberación sobre la señal original tiende a disminuir la amplitud de la misma [1].

2.2.2. Dialogue De-reverb de iZotope

Dialogue De-reverb es un módulo de la suite RX 7 de iZotope que consigue reducir o eliminar la reverberación no deseada de clips de audio que contengan diálogo. El módulo no necesita de un perfil de reverberación antes de empezar el procesamiento. Como alternativa, este se vale de un algoritmo de machine learning que separa el diálogo de las componentes que presentan reverberación. Se promociona como una solución para rescatar pistas de audio y evitar el uso de técnicas como el ADR, por ejemplo [2].

2.3. Intentos previos con redes neuronales

2.3.1. Red neuronal TDNN

En 1994, Chang et al. [3] propusieron el uso de una red neuronal TDNN (Time delay neural network) para el filtrado inverso de señales acústicas. El sistema compuesto en análisis fue el cuarto más la bocina, para el cual se realizaron simulaciones (RK4) empleando el modelo dinámico del mismo. Se entrenaron dos redes, una para el modelo en sí y otra para el inverso. La señal de prueba fue la suma del par de sinusoides $x(t) = 0.3(\sin(1.28\omega t)) + 0.5(\cos(3.93\omega t))$, con $\omega = 200\pi$ (Figura 1). Se concluyó que tanto las distorsiones lineales como no lineales pueden reducirse en un orden de magnitud.

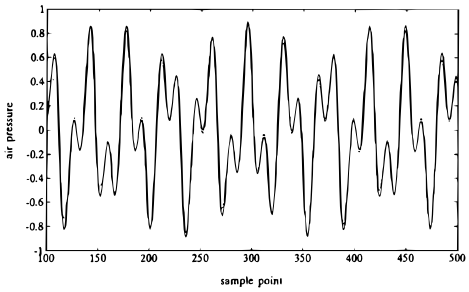


Figura 1: Deconvolución obtenida por la TDNN para la señal de prueba $x(t)$.

2.3.2. Red neuronal CMAC

Para la solución del problema de deconvolución y filtrado inverso no lineal, Glanz y Miller [4] plantearon el uso de una red neuronal CMAC (*Cerebellar Model Arithmetic Computer*). Iniciaron con pruebas de un sistema lineal, cuya respuesta impulsional correspondió a un sinusoide con decaimiento exponencial. Dentro de las pruebas se incluyó recuperar las señales originales, las cuales fueron ruido blanco (Figura 2) y una secuencia binaria aleatoria (Figura 3). La prueba final fue con el sistema no lineal $y[n] = \arctan(x[n]) + B(\arctan(x[n-1]))$ para el cual probaron diferentes valores de B. En este otro caso, la salida de la red CMAC fue similar a la deseada mas no idéntica.

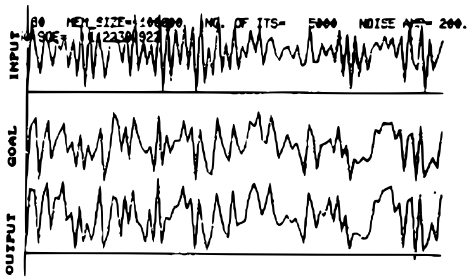


Figura 2: Deconvolución de ruido blanco empleando una CMAC.

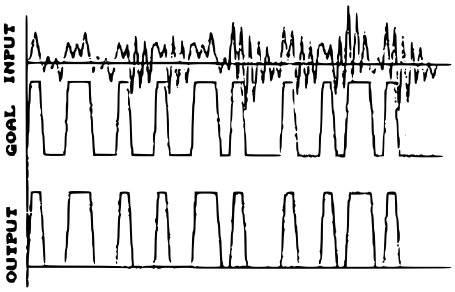


Figura 3: Deconvolución de un string de bits aleatorios empleando una CMAC.

CAPÍTULO 3

Justificación

La industria musical ha ido evolucionando, dando la oportunidad a más personas de componer y compartir su música. Antes los artistas debían firmar con alguna disquera para poder ser reconocido por las masas. Ya hoy con el avance tecnológico es mucho más fácil crear composiciones y darse a conocer. Ahora es posible crear desde estudios virtuales, emplear controladores MIDI, interfaces de audio, experimentar con diferentes plug-ins. Y para ser escuchado existen distintas plataformas tales como Spotify, SoundCluoud, YouTube, por mencionar algunas. Todo este entorno es un nicho para un grupo de músicos en específico, aquellos que graban y producen desde casa.

A pesar de la nueva accesibilidad de poder ser un músico de casa, se tiene sus limitaciones con respecto a la calidad del audio que se puede obtener. Uno de estos inconvenientes es trabajar en un cuarto que no ha sido tratado acústicamente. Este recinto introduce efectos indeseados tales como reflexiones y reverberación. Además, se debe tomar en cuenta el equipo de grabación que el músico emplea. Si este es de gama baja, este sumará distorsión al audio. Todos estos efectos representan las desventajas de ser un músico desde casa.

Este trabajo posibilitaría a músicos amateur lograr producción de audio de alta calidad invirtiendo el mínimo presupuesto. Este proyecto posibilitaría al músico de casa entregar audio tal como pretendía ser escuchado, eliminando todas las imperfecciones indeseadas que introducen las variables anteriores. Se planea realizarlo mediante la combinación de algoritmos, filtros adaptables y aprendizaje de máquina (*Machine Learning*). Así el músico de casa solo tendría que invertir en una pieza de software para entregar audio con calidad de estudio.

CAPÍTULO 4

Objetivos

4.1. Objetivo general

Usar filtros adaptativos y redes neuronales regresivas con el fin de eliminar los efectos de convolución no deseados que introducen variables externas, tales como un cuarto no tratado acústicamente o el equipo de grabación.

4.2. Objetivos específicos

- Seleccionar la variedad de filtro adaptativo que mejor se acople a la aplicación.
- Definir las señales de audio que servirán de entrenamiento, a modo que representen una amplia cantidad de casos distintos.
- Evaluar el desempeño de la combinación de un filtro adaptativo y una red neuronal.

CAPÍTULO 5

Alcance

Encontrar un algoritmo de deconvolución acústica que aprenda un set coeficientes generales para el filtrado de señales de audio. Mediante el uso de aprendizaje de máquina, buscar prescindir de señales originales durante el filtrado y solo emplearlas durante el entrenamiento. En específico, probar con distintas redes neuronales de una única capa intermedia (arquitecturas shallow). En caso de no obtener el filtrado esperado, se procederá a explicar los resultados generados, mas no se plantearán arquitecturas profundas u otros métodos de extracción de características.

Debido a la pandemia del COVID-19 del año 2020, existieron limitantes en cuanto a la experimentación que se realizó. Se inició con el limitado acceso a lugares, por lo que únicamente se usó el un dormitorio para las pruebas de grabación. La otra restricción estuvo en cuanto al equipo, el cual se usó el que el experimentador ya tenía disponible. En un escenario diferente, hubiera sido permisible el uso del mobiliario con el que contaba la universidad. También hubiera existido la oportunidad de hacer uso de un equipo más especializado de audio y el acceso a un cuarto tratado acústicamente.

CAPÍTULO 6

Marco teórico

6.1. Variables aleatorias

Una variable aleatoria, ya sea discreta o continua, es aquella que se genera a partir de un evento que depende del azar. Tómese por ejemplo la acción de tirar un dado, en el cual cada cara posee 1/6 de probabilidad de ser obtenida. Cada uno de los seis posibles resultados que se puede llegar a obtener es un evento elemental. Cada evento pertenece a un conjunto mayor que es conocido como el espacio muestral Ω, al cual le es asignado una probabilidad de uno ($\Pr\{\Omega\} = 1$). Por lo tanto, una variable aleatoria se define como aquella que se mapea de este espacio a algún número real o complejo [5].

6.2. Valor esperado E{x}

En procesos que involucran variables aleatorias, muchas veces se prefiere solo conocer el valor esperado en vez de una caracterización estadística completa. Se tiene de nuevo el ejemplo del dado, en el cual la media muestral está dada por la ecuación (1). Si se tiró el dado un suficiente número de veces (N_T lo suficientemente grande), n_k/N_T se puede aproximar a la probabilidad que cada cara tiene de salir $Pr\{x = k\}$. Tomando en cuenta lo anterior, se obtiene la aproximación mostrada en la ecuación (2). De forma general, la fórmula para la media o valor esperado para el caso discreto se muestra en la ecuación (3) [5].

$$\langle x \rangle_{N_T} = \frac{n_1 + 2n_2 + 3n_3 + 4n_4 + 5n_5 + 6n_6}{N_T}, \tag{1}$$

$$\langle x \rangle_{N_r} \approx \sum_{k=1}^{6} k \Pr\{x = k\}, \tag{2}$$

$$E\{x\} = \sum_k \alpha_k \Pr\{x = \alpha_k\}. \tag{3}$$

6.3. Procesos estocásticos

Un proceso estocástico es la extensión del concepto de variables aleatorias, el cual se define como una secuencia de este tipo de variables. Lo anterior significa que para cada evento del espacio de muestreo Ω, existe un valor respectivo $x[n]$. Cada variable aleatoria en un determinado momento n puede tener su propias funciones de distribución y densidad de probabilidad, o bien, ser las mismas para todo instante. De forma análoga al valor esperado para variables aleatorias, el de un proceso estocástico se pude obtener de la media del proceso $m_x[n] = E\{x[n]\}$ [5].

6.4. Autocorrelación y correlación cruzada

En procesos estocásticos se definen medidas de importancia tales como la autocorrelación y correlación cruzada. Estas medidas hablan sobre el grado de dependencia lineal que existe entre dos variables aleatorias. Se habla de autocorrelación cuando ambas variables son generadas por el mismo proceso estocástico (ecuación (4)). Por otro lado, si las variables son generadas por procesos diferentes, entonces se habla de correlación cruzada (ecuación (5))[5].

$$r_x(k,l) = E\{x(k)x^*(l)\}, \tag{4}$$

$$r_{xy}(k,l) = E\{x(k)y^*(l)\}. \tag{5}$$

6.5. Filtros adaptativos

Un filtro adaptativo es un filtro que toma una entrada estocástica $x[n]$ y va ajustando sus propios parámetros con el objetivo de obtener una señal de salida $y[n]$ lo más parecida a una señal deseada $d[n]$. Un filtro de este tipo se representa por medio de la ecuación (6), la cual expresa que cada muestra atrasada es multiplicada por un w_k correspondiente. Estos coeficientes w_k, conocidos como pesos, son los que el filtro va modificando para adaptarse. Este objetivo se logra a través de una señal de error $e[n]$, la cual dicta qué tanto deben cambiar los pesos.

$$y[n] = \sum_{k=0}^{N-1} w_k x[n-k]. \tag{6}$$

Un primer planteamiento propone usar el error cuadrático medio (valor esperado) como la medida de error a optimizar. La selección anterior se debe a la naturaleza de las señales involucradas. En este caso $x[n]$ se describe como una secuencia de variables aleatorias y, como consecuencia, $y[n]$ y $e[n]$ entran en la misma categoría. Lo anterior conduce a emplear el operador de valor esperado (ecuación (7)), el cual admite valores que exhiben este comportamiento. La forma cuadrática de la función es conveniente ya que al poseer un mínimo global es posible encontrar valores w_k óptimos [6].

$$\xi = E\{e^2\}. \tag{7}$$

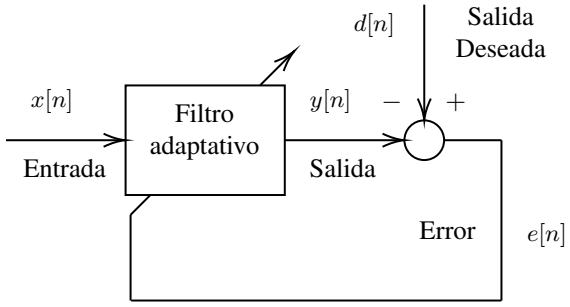


Figura 4: Esquema general de un filtro adaptativo FIR.

6.5.1. Algoritmo LMS estándar

El algoritmo Least-Mean Square (LMS) se presenta como un método para encontrar el vector de pesos $\mathbf{w}[n]$ con el cálculo de un gradiente aproximado de la función ξ. Obtener de forma directa el gradiente requiere conocer diferentes parámetros estadísticos, lo cual resulta computacionalmente caro. Debido a lo anterior se prefiere usar únicamente el error cuadrático e^2, con el cual se obtiene un estimando del gradiente $\hat{\nabla}[n]$. Adicionalmente se define una constante arbitraria β, la cual establece la tasa de adaptación [6].

$$w[n+1] = w[n] - \beta\hat{\nabla}[n]$$

6.5.2. Algoritmo LMS normalizado

El algoritmo LMS normalizado presenta una ventaja sobre la versión anterior. El problema con la versión estándar es la sensibilidad que posee a la magnitud de la señal de entrada $\mathbf{x}[n]$. Esta respuesta no deseada se puede evitar al sustituir la tasa de adaptación β por una que realice el escalamiento necesario al momento de actualizar el vector de pesos $\mathbf{w}[n]$. La nueva tasa (ecuación (8)) escala la amplitud de la entrada en este paso, necesario si el orden de magnitud de esta señal se encuentra muy por encima o por debajo del de la señal deseada $\mathbf{d}[n]$ [6].

$$\beta_{\text{NLMS}} = \frac{1}{\sum_{n=0}^{(N-1)} \boldsymbol{x}^2[n]}. \tag{8}$$

6.5.3. Algoritmo RLS convencional

El algoritmo RLS es otra alternativa para encontrar los vectores de peso del filtro adaptativo. La función objetivo determinística a minimizar por el RLS se presenta en la ecuación (9). Esta función depende de $\varepsilon[n]$, el cual se conoce como el error *a posteriori*; esto difiere del algoritmo LMS que emplea el error *a priori*. Adicionalmente se introduce un factor de memoria λ, el cual se puede ajustar en un rango de $0 << \lambda \leq 1$. Al diferenciar con respecto a $\mathbf{w}[k]$ e igualando a cero (ecuación (10)) es posible obtener el vector de pesos óptimos (ecuación 11). El vector de pesos se nota que depende de $\mathbf{R}_D[k]$ y $\mathbf{p}_D[k]$, que corresponden a la matriz de correlación de la señal de entrada y a la correlación cruzada entre la entrada y la señal deseada, respectivamente (ambas determinísticas) [7].

$$\mathcal{E}^d[n] = \sum_{i=0}^{n} \lambda^{n-i} \varepsilon^2[i] = \sum_{i=0}^{n} \lambda^{n-i} \left(d[i] - \mathbf{x}^T[i]\mathbf{w}[n]\right)^2, \tag{9}$$

$$\frac{\partial \mathcal{E}^d[n]}{\partial \mathbf{w}[n]} = -2 \sum_{i=0}^{n} \lambda^{n-i} \mathbf{x}[i] \left(d[i] - \mathbf{x}^T[i]\mathbf{w}[n]\right) = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \tag{10}$$

$$\begin{aligned} \mathbf{w}[n] &= \left(\sum_{i=0}^{n} \lambda^{n-i} \mathbf{x}[i] \mathbf{x}^T[i] \right)^{-1} \sum_{i=0}^{n} \lambda^{n-i} \mathbf{x}[i] d[i] \\ &= \mathbf{R}_D^{-1}[n] \mathbf{p}_D[n] \end{aligned}. \tag{11}$$

El costo computacional de calcular la inversa de $\mathbf{R}_D[n]$ puede ser reducido. La inversión directa de esta matriz posee una complejidad de $O[N^3]$. Lo anterior se puede reducir si se emplea el lema de inversión de matrices (ecuación (12)), donde $\mathbf{A}$ y $\mathbf{C}$ son no singulares. Sustituyendo por $\mathbf{A} = \lambda \mathbf{R}_D$, $\mathbf{B} = \mathbf{D}^T = \mathbf{x}[n]$ y $\mathbf{C} = 1$ se obtiene la expresión mostrada en la ecuación (13) [7].

$$[\mathbf{A} + \mathbf{BCD}]^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B} \left[\mathbf{DA}^{-1}\mathbf{B} + \mathbf{C}^{-1}\right]^{-1} \mathbf{DA}^{-1}, \tag{12}$$

$$\mathbf{S}_D[n] = \mathbf{R}_D^{-1}[n] = \frac{1}{\lambda} \left(\mathbf{S}_D[n-1] - \frac{\mathbf{S}_D[n-1]\mathbf{x}[n]\mathbf{x}^T[n]\mathbf{S}_D[n-1]}{\lambda + \mathbf{x}^T[n]\mathbf{S}_D[n-1]\mathbf{x}[n]} \right). \tag{13}$$

6.6. Redes neuronales artificiales (RNA)

El nombre red neuronal no es coincidencia, ya que esta imita la forma en la que las neuronas biológicas se activan, comunican y organizan. Una célula nerviosa recibe una señal eléctrica por las dendritas y, a través del axón, la transmite hasta sus terminales. El que una neurona se dispare o no dependerá de la intensidad del estímulo recibido. Esto significa que la señal de la entrada debe llegar a cierto umbral para que la neurona se logre activar. La transmisión a través de la neurona resulta en una señal eléctrica en sus terminales, lo cual estimula las dendritas de una o varias neuronas posteriores. De forma análoga, una red neuronal artificial toma este ejemplo para organizarse (Figura 5) [8].

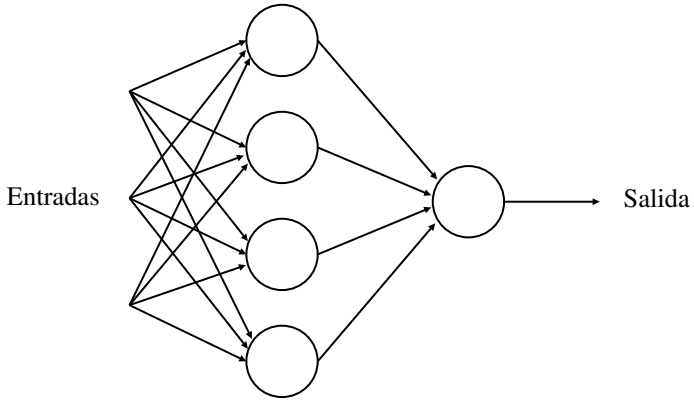


Figura 5: Esquema de una red neuronal sencilla.

Siguiendo este comportamiento es como se modela una neurona artificial para que opere de forma similar. La entrada se representa como un vector $\mathbf{x}$ que, después de ponderada por la neurona, genera una salida (Figura 6). A este vector $\mathbf{x}$ corresponde un vector de pesos $\mathbf{w}$ y un factor de sesgo $\mathbf{b}$ (ecuación (14)). Este resultado luego pasa a la función de activación, la sigmoide por ejemplo (ecuación (15)), que genera un valor cercano a 1 entre más positivo sea z y 0 para valores más negativos. Lo anterior coincide con el comportamiento a emular, ya que una salida más cercana a 1 corresponde a un estímulo más fuerte en la entrada [9].

$$z = \mathbf{w}^T\mathbf{x} + b. \tag{14}$$

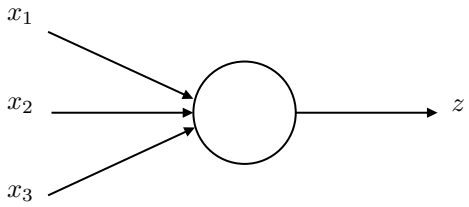


Figura 6: Esquema básico de una neurona artificial.

$$\sigma(z) \equiv \frac{1}{1 + e^{-z}}. \tag{15}$$

6.6.1. Teorema de aproximación universal

El teorema de aproximación universal garantiza la existencia de una red neuronal capaz de calcular una función $f(x)$, bajo ciertas consideraciones. Este teorema aplica tanto para funciones SISO y MIMO, y aplica incluso a arquitecturas sencillas como las redes neuronales con una sola capa oculta. En cuanto a las consideraciones, la primera tiene que ver con la exactitud. Para una función $f(x)$, la salida de la red $g(x)$ no va a ser igual, sino que va a estar dentro cierto margen ($|f(x)-g(x)| < \epsilon$). El otro punto a tomar en cuenta es que la red podrá calcular la función $f(x)$ si esta es continua. A pesar de esto, en el caso de ser discontinua, una aproximación continua es aceptable y no representa limitantes de implementación en la práctica. El teorema se resume a que las redes neuronales de al menos una capa oculta pueden aproximar una función $f(x)$ a una exactitud deseada ϵ [9].

6.6.2. Funciones de activación

Tangente hiperbólica

La tangente hiperbólica, es otra opción popular además del sigmoide antes introducido, que presenta ventajas y desventajas. A diferencia del sigmoide, tanh está centrada en 0, por lo que se le facilita modelar data que oscile entre valores positivos y negativos. Esta función comparte la misma desventaja que la sigmoide, que es el desvanecimiento de gradiente. Este se exhibe con valores muy positivos o negativos en la entrada con los cuales las predicciones no cambian significativamente. Este problema hace el aprendizaje sea lento o que la red simplemente deje de aprender [10].

RBF (*Radial Basis Functions*)

La característica principal de las RBF es que son simétricas con respecto a un eje vertical. Una red neuronal RBF, con una sola capa intermedia, se puede describir por la ecuación (16). Inmediatamente se nota que la función depende de la diferencia $\|\mathbf{x} - \mathbf{c}_i\|$. Esto quiere decir que entre menor sea esta diferencia (data más cercana al centro), mayor será la salida. Un ejemplo de RBF comúnmente utilizada es la gaussiana, cuya función se muestra en la ecuación (17) [11].

$$y = w_0 + \sum_{i=1}^{n_h} w_i f\left(\|\mathbf{x} - \mathbf{c}_i\|\right), \tag{16}$$

$$f(x) = e^{\frac{-(x-c)^2}{r^2}}. \tag{17}$$

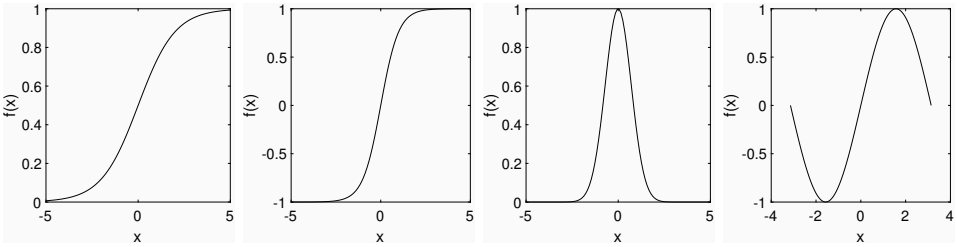
Las redes neuronales RBF encuentran una función arbitraria mediante la combinación lineal de funciones radiales. Las redes RBF son en esencia redes de funciones kernel que limitan la salida a una región local en el espacio de vectores de entrada [11]. De aquí se hace mención a la regresión kernel, la cual emplea el mismo tipo de funciones para realizar el

mapeo de una entrada a una salida. De igual forma que en la redes, en la regresión la salida se conforma de una suma ponderada donde cada valor kernel es multiplicado por un peso w [12].

Funciones periódicas

El uso señales periódicas como funciones de activación ha resultado en una mayor velocidad de aprendizaje, mayor capacidad computacional y generalización que el de las funciones monótonas. Este desempeño superior es observable cuando se habla de la superposición de hiperplanos en el espacio. Ya que una función monótona sólo divide el espacio en dos áreas distinguibles. Cuando varios hiperplanos son combinados, se pueden lograr clasificaciones no lineales complejas. En contraste, una sola función sinusoidal dividiría al espacio en un número infinito de regiones. Se ha recalcado, además, que emplear funciones periódicas es equivalente a construir una serie de Fourier, en la cual es posible el ajuste de las frecuencias [13].

Resumen de las funciones de activación



(a) Sigmoide. (b) Tanh. (c) Gaussiana. (d) Sinusoidal.

Figura 7: Diferentes funciones de activación.

6.6.3. *Backpropagation*

Backpropagation es un algoritmo que permite obtener el gradiente de alguna función con respecto a uno o varios parámetros de interés. *Backpropagation* se presenta como una alternativa eficiente para el cálculo de las derivadas parciales. Este es comparable con una simple derivación *forward*. Véase el diagrama que se muestra en la Figura 8 para comparar ambos métodos de derivación. Con el método *forward*, el cambio en X se tendría que dirigir por nueve caminos diferentes y luego sumar la contribuciones que de cada camino (ecuación (18)). Por otro lado, *backpropagation* toma derivaciones parciales de atrás para adelante, considerando mejor cómo cada nodo afecta la salida. Por lo tanto, lo anterior se resume a que *backpropagation* da la derivada de una salida con respecto a todas las entradas y la derivación *forward* da las derivadas de todos las salidas con respecto a una entrada [14].

$$\frac{\partial Z}{\partial X} = \alpha\delta + \alpha\epsilon + \alpha\zeta + \beta\delta + \beta\epsilon + \beta\zeta + \gamma\delta + \gamma\epsilon + \gamma\zeta. \tag{18}$$

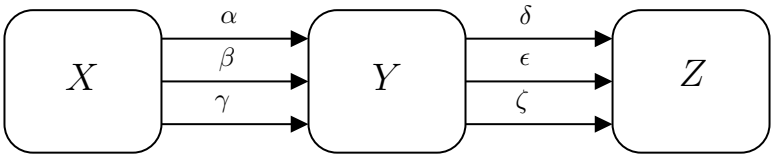


Figura 8: Diagrama que muestra la relación funcional entre los nodos X, Y y Z.

6.6.4. Función de costo

Se define una función de costo J, la cual le indica a la RNA qué tanto difiere la predicción realizada del valor verdadero. En las primeras iteraciones de entrenamiento este costo será alto, ya que la red a penas está empezando a aprender. Conforme se realicen iteraciones los parámetros se irán actualizando y la red cada vez acertará a valores más cercanos de los reales. Se establece por lo tanto la función de pérdida $\mathcal{L}$, la cual indica esta diferencia para una sola muestra. El promedio de la función $\mathcal{L}$ para todas las muestras se define como la función de costo J (ecuación 19).

$$J = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}\left(a^{(i)}, y^{(i)}\right). \tag{19}$$

6.6.5. Descenso por gradiente

Una vez ya definida una función costo J lo que queda es encontrar los pesos $\mathbf{w}$ y sesgos $\mathbf{b}$ que minimizan el valor de esta. Para asegurarse de que se encontrará un mínimo global la función debe cumplir con la condición de ser convexa. La búsqueda del mínimo se realiza en la dirección de máximo descenso, lo cual se encuentra con el gradiente de la función J con respecto a cada uno de los parámetros. Véase el caso, unidimensional, por ejemplo (Figura 9), donde el problema se reduce a encontrar la derivada para definir en qué dirección desplazarse. En este caso, una pendiente positiva significa un desplazamiento hacia la izquierda. Después de n pasos, se encontrará el mínimo global. Por último se introduce la tasa de aprendizaje α, la cual definirá el tamaño de cada uno de los pasos dados [15].

Algoritmo 1: Entrenamiento para una red neuronal artificial

```
Resultado: Red neuronal artificial entrenada
definir k iteraciones;
w y b se inicializan aleatoriamente;
mientras i ≤ k hacer
    aplicar forward propagation;
    obtener el resultado de J;
    aplicar backward propagation;
    actualizar parámetros;
fin
```

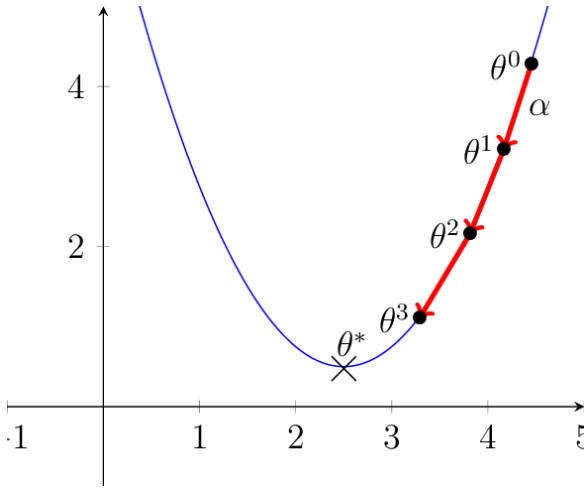


Figura 9: Descenso gradiente ejemplificado en una función cuadrática. Al ser esta convexa, se garantiza un mínimo global[16].

6.6.6. Redes neuronales regresivas

Las redes neuronales regresivas son un grupo especial de las redes neuronales artificiales. Estas se diferencian por presentar una salida unidimensional con un valor real. La entrada puede ser de una única variable, o bien, multivaribale. Esta siguen presentando las funciones de activación en las capas ocultas, pero en el nodo de salida está la diferencia. La salida no presenta función de activación y solo representa la suma del resultado de la capa anterior, incluyendo los respectivos pesos y sesgos. Para la función de costo, el error cuadrático medio es la que comúnmente se utiliza (ecuación 20) [15].

$$J\left(\theta_0, \theta_1\right) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta\left(x_i\right) - y_i\right)^2 . \tag{20}$$

6.6.7. TDNN (*Time delay neural network*)

Las TDNN son un tipo de redes que al momento de entrenarse por descenso por gradiente, se obtiene un filtro no lineal óptimo [11]. Esta red es de tipo *feedforward*, por lo que no hay retroalimentación en la red. Esta arquitectura se puede interpretar como una combinación no lineal estática de filtros adaptativos FIR. Tal como se observa en la Figura 10, la capa de entrada está conformada por una línea de dalays unitarios. Para el entrenamiento de esta red se puede usar algoritmo de *backpropagation* estándar.

6.6.8. Redes neuronales *Focused Gamma*

La red *Focused Gamma* (Figura 11), a diferencia de la TDNN, adicionalmente implementa un mecanismo de memoria a corto plazo en la entrada [17]. Como se observa en la figura, las unidades $G(z)$ son las encargadas de aplicar dicho lazo de retroalimentación. La

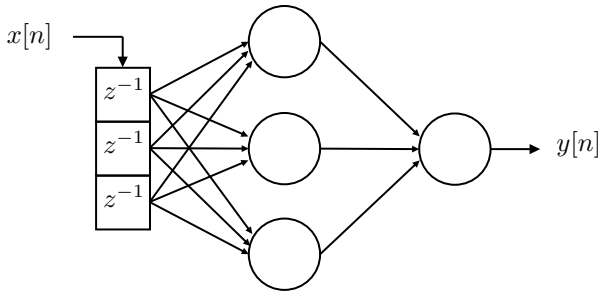


Figura 10: Red neuronal TDNN.

memoria se ajusta con la variación de μ. En dicho caso, se identifica que la red TDNN es un caso especial de esta red cuando $\mu = 1$ (sin memoria). Esta red es un modelo recurrente, pero debido a la topología, esta se puede seguir entrenando usando *backpropagation*. El detalle está en la adaptación μ, en la que sí es necesario aplicar un algoritmo de aprendizaje recurrente.

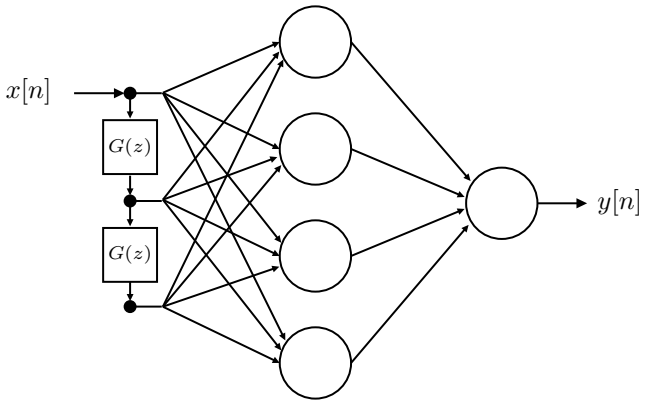


Figura 11: Red neuronal *Focused Gamma*.

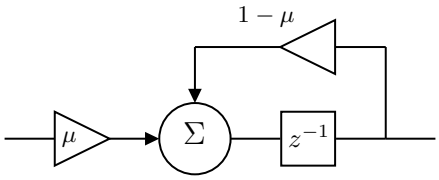


Figura 12: Unidad recursiva $G(z)$ de la red neuronal *Focused Gamma*.

6.6.9. CMAC (*Cerebellar Model Articulation Controller*)

La red neuronal CMAC (figura 13) es una arquitectura que consigue el objetivo de generalización gracias a una serie de mapeos. El primer mapeo es no lineal y ocurre entre el espacio de los vectores de entrada discretos a los vectores de asociación ($\mathbf{u} \rightarrow \mathbf{a}$). Los vectores de asociación son vectores binarios en los que solo ciertos bits C son activados. Entre estos dos espacios debe existir correspondencia de cercanía. Esto significa que dos vectores de entrada $\mathbf{u}$ tendrán más bits C en común que un par de vectores estén más alejados entre sí.

El mapeo de los vectores de asociación a la salida ($\mathbf{a} \rightarrow y$) es donde se multiplica por los pesos w, los cuales pueden ser entrenados por LMS [18].

$$y(\mathbf{u}) \;=\; \mathbf{a}(\mathbf{u})^T \mathbf{w}$$

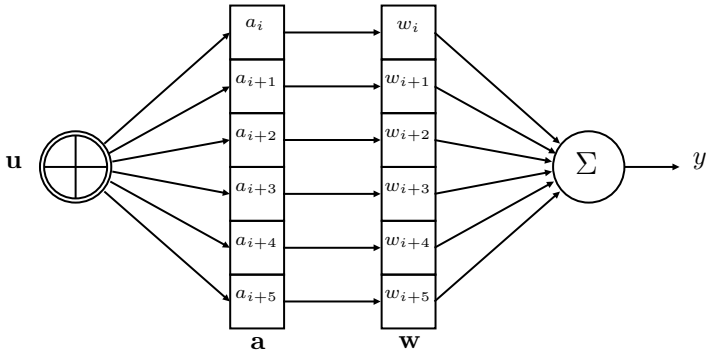


Figura 13: Red neuronal CMAC.

CAPÍTULO 7

Diseño experimental

7.1. Equipo

Las pruebas realizadas se llevaron a cabo con la bocina estéreo iHM9 de *iHome* [19] y con el micrófono de una computadora portátil *ASUS X555L* [20]. En las pruebas realizadas, la bocina se colocó justo a la derecha del computador. Este aparato, originalmente alimentado por baterías, pasó a ser alimentado por una fuente DC que generó 5.0V y 550 mA. Por parte del equipo de grabación, el micrófono se ubicaba por arriba de la pantalla, justo a la par de la cámara.

7.2. Respuesta en frecuencia

Se midió la respuesta en frecuencia de la bocina y el micrófono en conjunto por medio de un barrido espectral de una señal sinusoidal. En la medición a dicha respuesta en frecuencia, se probaron tonos bajos, medios y altos [21]. Con una frecuencia muestreo establecida en 48kHz, se empezó barriendo la frecuencia del sinusoide en aumentos 50 Hz, desde 0Hz hasta 24kHz. Se realizaron un total de tres corridas y luego se promedió. El resultado fue la gráfica del máximo valor de intensidad obtenido conforme se fue variando la frecuencia en el sistema, la cual se muestra en la Figura 14.

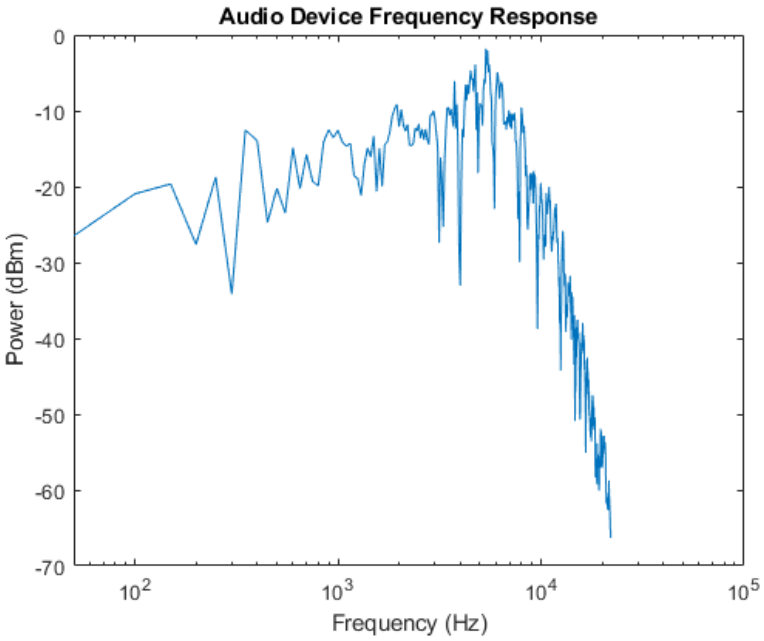


Figura 14: Respuesta en frecuencia del sistema.

7.3. Geometría y características de la habitación

El cuarto en el que se realizaron la pruebas de audio fue un dormitorio convencional. En la Figura 15 se muestra un plano de la habitación, con dimensiones generales. El recinto presentó una altura constante de 238.50 cm. Dentro de este cuarto se posicionó la laptop cerca de la esquina inferior izquierda. Está se encontraba sobre una mesa de madera, a 72cm por encima del piso. Cabe mencionar además que la pose de la computadora se mantuvo a 90 grados de apertura.

La recámara se termina de caracterizar por los diferentes elementos dentro y materiales de las superficies. Los objetos más voluminosos fueron la cama del centro y un sillón de tela que se encontraba a la izquierda de esta. Los demás muebles eran de madera o de MDF con recubrimiento de plástico. Las paredes eran de block con revestimiento y un acabado liso. Por otra parte, el techo se distinguió por tener un acabado granulado. El piso era cerámico y en el ventanal colgaban cortinas de tela gruesa.

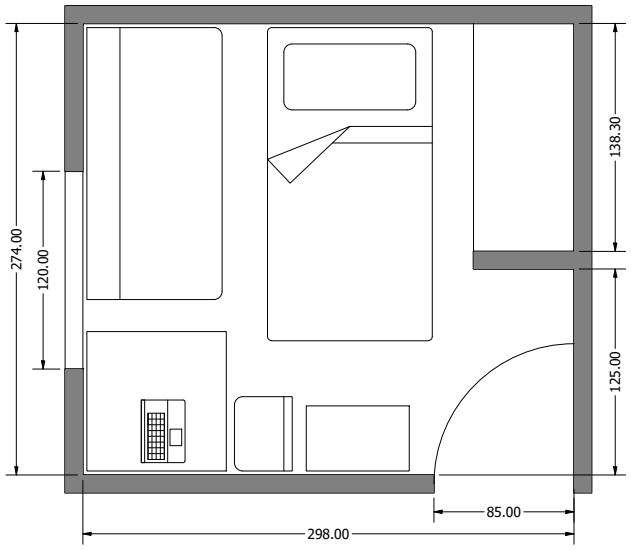


Figura 15: Vista de planta de la habitación de prueba.

CAPÍTULO 8

Metodología

8.1. Grabación de las señales de audio

Lo que se realizó fue reproducir el audio a través de los parlantes de la computadora y grabar con el micrófono del mismo equipo. La situación simplificada se muestra en la Figura 16. La señal $d[n]$ es la deseada y fue la que se reprodujo en las bocinas. Esta señal de audio fue afectada por la reverberación del recinto, así como por el ruido introducido por los dispositivos de grabación. La señal obtenida por el micrófono es $x[n]$, la cual corresponde a la señal de audio perturbada.

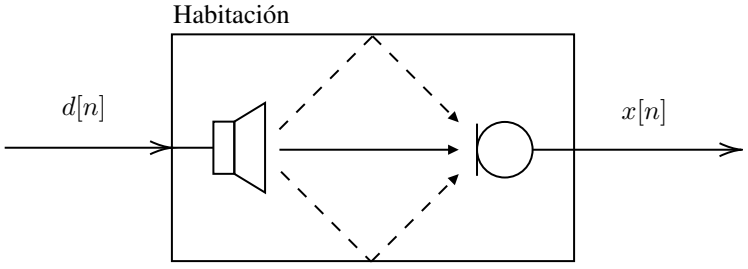


Figura 16: Diagrama simplificado del sistema de grabación.

Se veló para que la reproducción de la señal deseada fuera de la más alta fidelidad posible, así que se escogieron pistas de formato *lossless*. Se usó ya sea el formato WAV, que asegura la máxima calidad de audio, o de CODEC eficiente, como FLAC [22].

Se procuró que las condiciones del cuarto fueran las mismas ideales, para que las señales grabadas no fueran comprometidas por alguna variable ajena al proceso. La grabación se realizó con la ventana y puerta cerradas para que un ruido del exterior no afectara el proceso. Además, solo una persona estuvo en la habitación durante la captura de datos.

El programa utilizado para la reproducción y grabación de la señales fue *Matlab*. Antes de la grabación se aseguró que otros programas no interfieran en la obtención de datos. Por ejemplo, al administrador de sonido *Realtek HD* se le desactivaron el supresor de ruidos y la cancelación de eco acústico. Ya en *Matlab* se fijó un solo canal de grabación, una frecuencia de muestreo fue de 44100Hz y muestras de 24-bits de resolución. Estos parámetros se usaron con la función *audiorecorder* para crear una grabadora y así poder capturar cada señal de prueba. Al terminar, las señales perturbadas se guardaron en formato WAV con la función *audiowrite*.

8.2. Selección de las señales

Dentro de las señales escogidas se hizo una gran distinción entre señales determinísticas y *clips* musicales. Dentro del primer grupo se encuentran las señales básicas (sinusoides, ondas cuadradas y ondas diente de sierra), que solo necesitan de la frecuencia fundamental para ser descritas. Un aumento de complejidad a los sinusoides fue controlar ya sea amplitud, frecuencia o ambas por un *LFO* y barrer su frecuencia en un tiempo dado. El otro grupo fueron los primeros segundos de alguna canción, de artistas varios. El listado de las señales empleadas se muestra en los Cuadros 1 y 2.

El porqué de haber escogido este set de señales determinísticas recae en su naturaleza periódica. Filtrar las señales básicas equivale a alimentar al filtro varias veces con la misma información. Se esperó que el filtro lograra aprender rápido los coeficientes para este tipo de señales. Un paso más arriba fue modular amplitud y frecuencia en las cuales se agrega complejidad, pero se conserva la misma naturaleza periódica de las señales. Restante fue el barrido sinusoidal, cuyo objetivo fue el de ver qué tal maneja el filtro un cambio lineal de frecuencias.

En relación a los clips musicales, la gran variedad de señales fue para presentar ante los filtros señales con contenido espectral más complejo, que evoluciona de forma más dinámica en el dominio temporal. Se quizo ver el filtrado ante secciones de audio en las que solo ejecuta un instrumento. El otro grupo ya involucraba pieza musicales complejas, en las que escucha a varios instrumentos en conjunto. Se procuró también variar el género musical, entre los cuales se escogió *rock*, *jazz* y clásica.

Señal	Parámetro(s)
Ondas sinusoidales	f = [250 Hz, 1 kHz, 10 kHz]
Ondas cuadradas	f = [250 Hz, 1 kHz, 10 kHz]
Ondas diente de sierra	f = [250 Hz, 1 kHz, 10 kHz]
Onda sinusoidal AM	$f_{central}$ = 2.5 kHz, LFO_f = 0.5 Hz
Onda sinusoidal FM	$f_{central}$ = 2.5 kHz, LFO_f = 10 Hz
Onda sinusoidal AM y FM	$f_{central}$ = 2.5 kHz, LFO_{AM} = 0.5 Hz, LFO_{FM} = 10 Hz
Barrido sinusoidal	f_0 = 20 Hz, f_1 = 10 kHz, t = 10s

Cuadro 1: Diferentes señales determinísticas empleadas y los respectivos parámetro(s) empleados.

Nombre	Artista	t(s)	instrumentos
Bohemian Rhapsody	Queen	12	vocales
Cadence	The Long Faces	12	batería
Peru	Julian Lage	30	guitarra
Week No. 8	Fabrizio Paterlini	30	piano
Lonely Cat (Demo)	The Kooks	45	guitarra + vocales
Atlantic Limited	Julian Lage	45	guitarra + contrabajo + batería
Karma Police	Radiohead	60	guitarra + piano + vocales
The Blue Danube	-	60	orquesta

Cuadro 2: Diferentes clips musicales empleados. El tiempo t hace referencia a los primeros segundos utilizados de la canción en donde se escuchan los instrumentos listados.

8.3. Implementación de los filtros adaptativos

Ya con las señales deseada y perturbada, se terminó de preparar la data y prosiguió a definir ciertos parámetros. Luego se definió el orden N de cada uno de los filtros. Se terminó la preparación al definir el resto de parámetros. Para el filtro LMS estándar se estableció la tasa de aprendizaje β. Para el filtro RLS se establecieron el factor de persistencia λ y el inverso del estimado de la señal de entrada δ. Los valores implementados para cada set de casos se presenta en el Cuadro 3.

Una vez hecho lo anterior, ya fue posible filtrar la señal perturbada $x[n]$ para obtener la estimada $y[n]$. El proceso se muestra en la Figura 17. La implementación de ambos tipos de filtro se realizó en Matlab, aplicando los algoritmos 2 y 3. Cabe resaltar que el filtro RLS no depende explícitamente de la señal de error $e[n]$, sino que usa los estimados de la matriz de correlación S y el vector de correlación cruzada p.

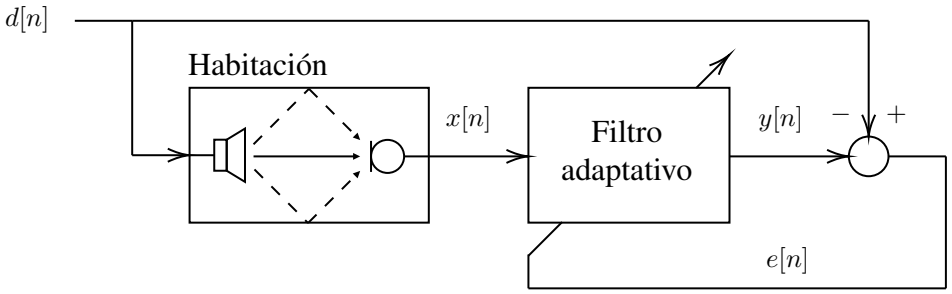


Figura 17: Diagrama que muestra el proceso de filtrado.

Filtro	determinísticas	Clips musicales
LMS estándar	$\beta = 0.1$, N = 50	$\beta = 0.01$, N = 1000
LMS normalizado	N = 50	N = 1000
RLS convencional	$\lambda = 0.5$, $\delta = 2$, N = 20	$\lambda = 0.5$, $\delta = 2$, N = 20

Cuadro 3: Parámetros y orden de cada filtro al ser empleado con las distintas señales de prueba.

Algoritmo 2: Filtro adaptativo por LMS

Resultado: Señal estimada $y[n]$
si *LMS estándar* **entonces**
  definir tasa de aprendizaje β;
fin
obtener el tamaño M de las señales $d[n]$ y $x[n]$;
preasignar memoria para la señal estimada $y[n]$;
inicializar linea de $delay's$ unitarios en cero;
inicializar pesos w en cero;
para $i \leq M$ **hacer**
  colocar la i-ésima muestra al principio de la linea de $delay's$;
  calcular el estimado de la señal $y[n]$;
  calcular el error $e[n]$;
  actualizar pesos w;
  generar corrimiento unitario en linea de $delay's$;
  guardar la i-ésima muestra de la señal $y[n]$;
fin

Algoritmo 3: Filtro adaptativo por RLS

Resultado: Señal estimada $y[n]$
definir el factor de persistencia λ (entre 0 y 1);
definir δ;
obtener el tamaño M de las señales $d[n]$ y $x[n]$;
preasignar memoria para la señal estimada $y[n]$;
inicializar linea de $delay's$ unitarios en cero;
inicializar vector de correlación cruzada p;
inicializar la matriz de correlación S;
para $i \leq M$ **hacer**
  colocar la i-ésima muestra al principio de la linea de $delay's$;
  obtener el estimado de la matriz S;
  obtener el estimado del vector p;
  actualizar vector de pesos $w = S * p$;
  calcular el estimado de la señal $y[n]$;
  generar corrimiento unitario en linea de $delay's$;
  guardar la i-ésima muestra de la señal $y[n]$;
  actualizar estimados S y p anteriores;
fin

8.4. Implementación de las redes neuronales

Se combinaron diferentes arquitecturas neurales y funciones de activación, para luego probar cada combinación con cada señal de prueba. Las arquitecturas escogidas fueron la TDNN y *Focused Gamma*. De las funciones de activación a probar con cada arquitectura fueron: la sigmoide, tanh y sinusoidal. El entrenamiento de cada red se hizo por el método estándar *backpropagation*. Comó útlimo detalle, las redes también fueron implementadas en

MATLAB.

Independientemente del caso que fuera, hubo ciertos parámetros que se mantuvieron constantes en cada arquitectura y tipo de entrenamiento. Los parámetros generales se observan en el Cuadro 4, los cuales se encontraron de forma heurística. En dicho cuadro los parámetros ϵ y μ inicial son únicamente necesarios para la arquitectura *Gamma*. Los demás parámetros son compartidos por ambas arquitecturas. Ya en cuanto al entrenamiento por cada tipo de señal, los hiperparámetros se mantuvieron constantes.

El entrenamiento para cada tipo de señales, determinísticas y clips musicales, tuvo una ligera variación en cuanto a otros detalles. Para las determinísticas se optó por una red neuronal por cada señal. Por otra parte, para los clips musicales se decidió primero entrenar la red neronal con toda la lista de reproducción y luego filtrar en *feedforward* cada canción por la red. Se hizo de esta forma ya que el interés principal de este trabajo consiste en lograr que una única red logre eliminar el ruido y perturbaciones de las señales más complejas, que son los clips musicales.

Parámetro	Valor
Tamaño de *batch*	4410
Número de *epochs*	5
Tasa de aprendizaje β	0.1
Tasa de aprendizaje ϵ	(5e3)*β
Valor de μ inicial	0.75
Nodos en la entrada	250
Nodos intermedios	250

Cuadro 4: Parámetros constantes durante el entrenamiento de las diferentes redes neuronales.

La función de activación sigmoide se tuvo que modificar para poder acoplarse al rango en el que se encontraban las señales. Originalmente dicha función de activación está definida entre 0 y 1 (Figura 7). Ya que las pistas de audio oscilan entre -1 y 1, no sería posible obtener con el sigmode original señales de salida de la misma naturaleza. Por lo tanto, fue necesario centrar la función en 0 y multiplicarla por un factor de 2. La modificación permitió obtener una forma similar a la función tanh, con lo cual ya fue posible el filtrado de las diferentes señales de prueba $x[n]$.

La función RBF no fue posible implementarla por el mismo impedimento que presentó el sigmoide. Se probó usar la RBF gaussiana, aunque se notó que existe en el mismo rango que el sigmoide original. Un cambio pertinente sería no usar una función gaussiana, sino una radial que este definida tanto para valores positivos como negativos. Una posible alternativa sería la función *sinc*, que sí cumple con dicha condición.

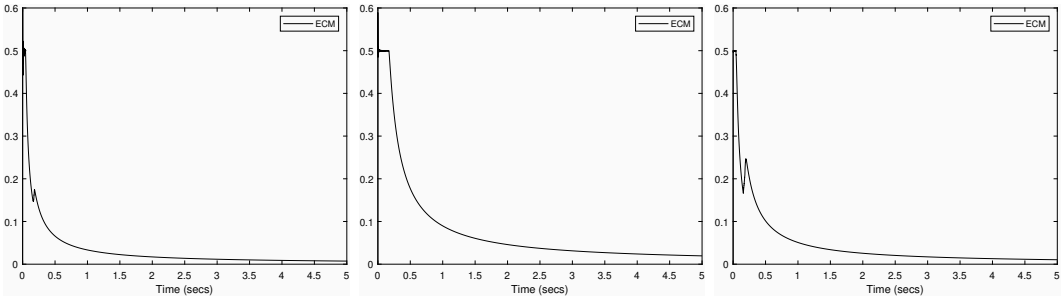
CAPÍTULO 9

Filtros adaptativos

9.1. LMS estándar

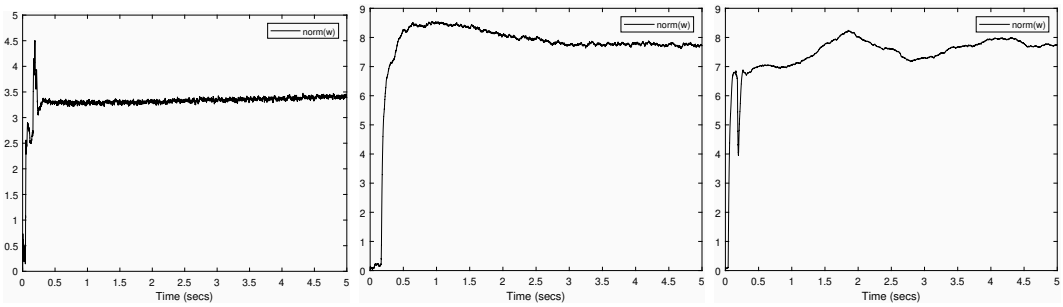
9.1.1. Pruebas con señales determinísticas

Ondas sinusoidales



(a) 250 Hz (b) 1 kHz (c) 10 kHz

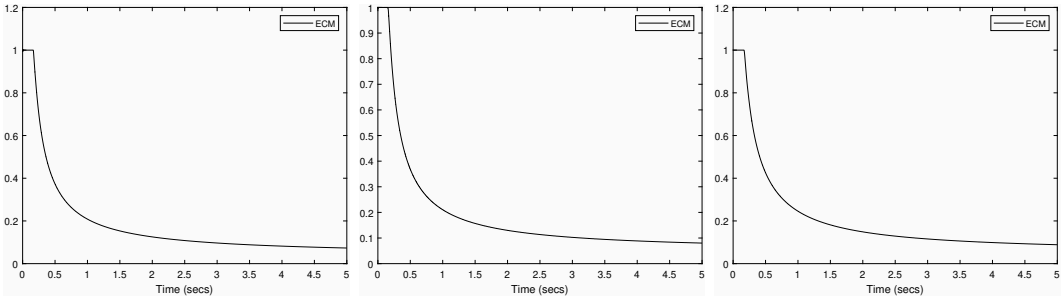
Figura 18: ECM del filtrado de sinusoides por LMS estándar.



(a) 250 Hz (b) 1 kHz (c) 10 kHz

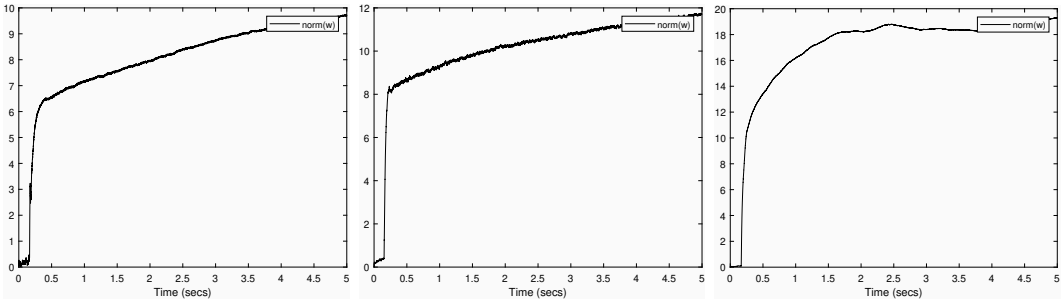
Figura 19: Magnitud del vector **w** del filtrado de sinusoides por LMS estándar.

Ondas cuadradas



(a) 250 Hz (b) 1 kHz (c) 10 kHz

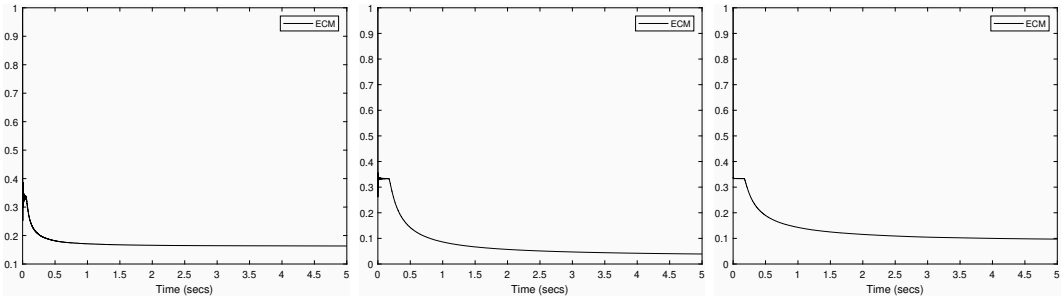
Figura 20: ECM del filtrado de ondas cuadradas por LMS estándar.



(a) 250 Hz (b) 1 kHz (c) 10 kHz

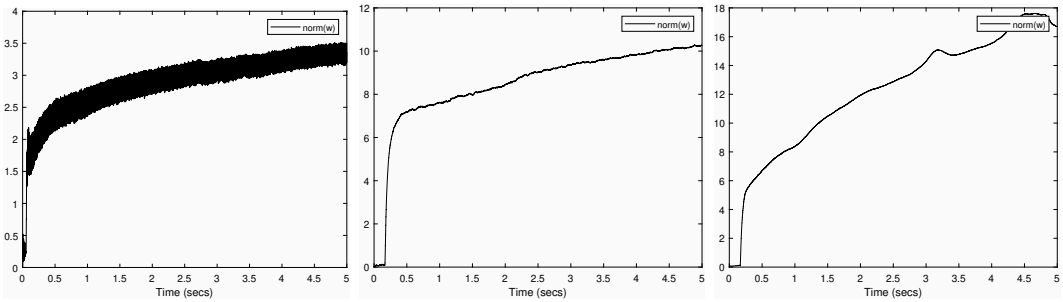
Figura 21: Magnitud del vector **w** del filtrado de ondas cuadradas por LMS estándar.

Ondas diente de sierra



(a) 250 Hz (b) 1 kHz (c) 10 kHz

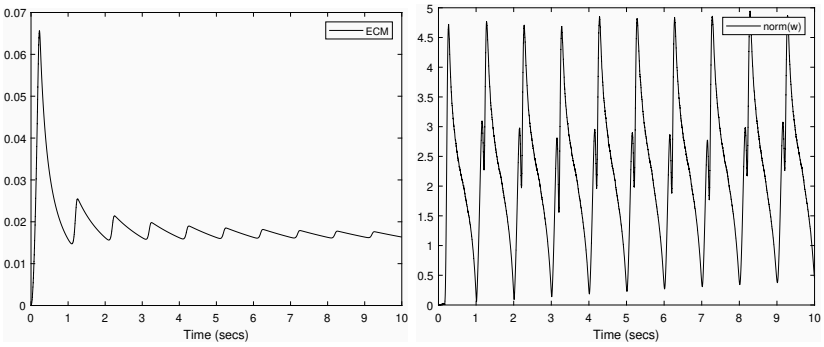
Figura 22: ECM del filtrado de ondas diente de sierra por LMS estándar.



(a) 250 Hz (b) 1 kHz (c) 10 kHz

Figura 23: Magnitud del vector **w** del filtrado de ondas diente de sierra por LMS estándar.

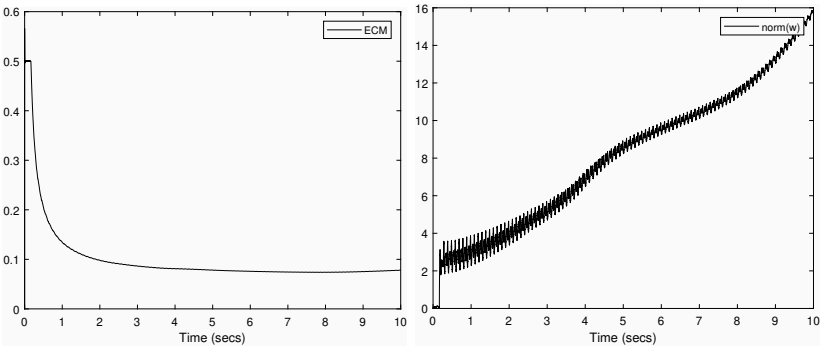
Onda AM



(a) ECM. (b) Magnitud del vector **w**.

Figura 24: Resultados del filtrado de un sinusoide de amplitud modulada por LMS estándar.

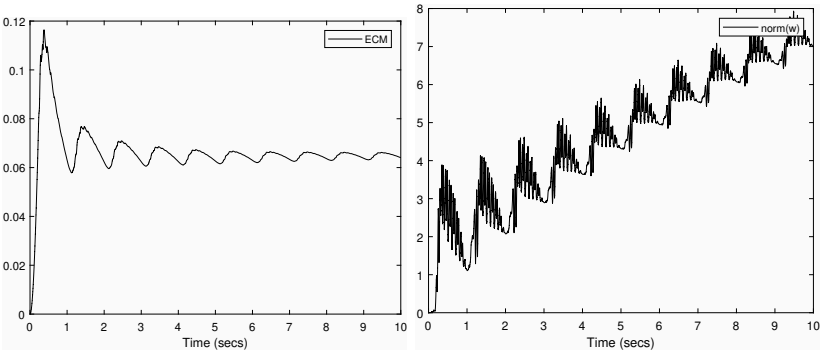
Onda FM



(a) ECM. (b) Magnitud del vector **w**.

Figura 25: Resultados del filtrado de un sinusoide de frecuencia modulada por LMS estándar.

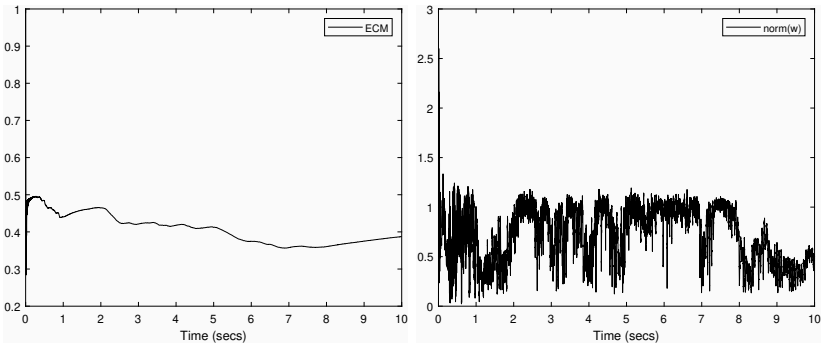
Onda AM y FM



(a) ECM. (b) Magnitud del vector **w**.

Figura 26: Resultados del filtrado de un sinusoide de amplitud y frecuencia modulada por LMS estándar.

Barrido sinusoidal



(a) ECM. (b) Magnitud del vector **w**.

Figura 27: Resultados del filtrado de un barrido sinusoidal (de 20 Hz a 10 kHz) por LMS estándar.

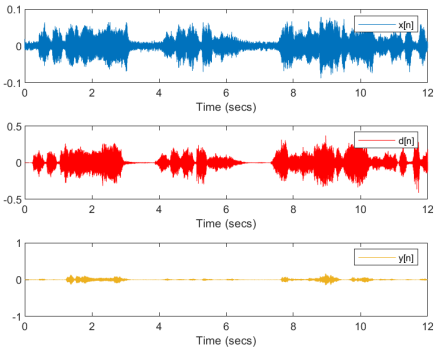
ECM finales y máximas magnitudes de w

Señal	ECM final	Máx $\|\|\mathbf{w}\|\|$
Onda sinusoidal 250 Hz	0.0072624	4.5041
Onda sinusoidal 1 kHz	0.019509	8.5417
Onda sinusoidal 10 kHz	0.010425	8.2473
Onda cuadrada 250 Hz	0.073163	9.7328
Onda cuadrada 1 kHz	0.080238	11.7591
Onda cuadrada 10 kHz	0.088599	19.3036
Onda diente de sierra 250 Hz	0.1633	3.5136
Onda diente de sierra 1 kHz	0.039048	10.288
Onda diente de sierra 10 kHz	0.097161	17.6184
Onda AM	0.016358	4.945
Onda FM	0.077998	15.8568
Onda AM y FM	0.064038	7.9326
Barrido sinusoidal	0.38735	2.6007
Promedio	0.0865	9.6034

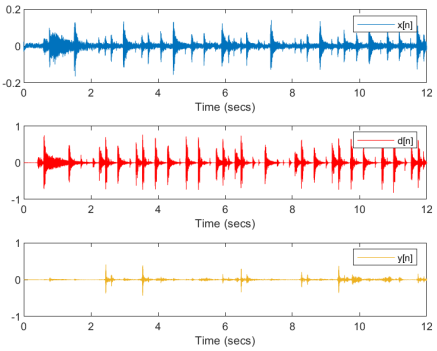
Cuadro 5: ECM finales y máximas magnitudes de **w** alcanzados por el LMS estándar para señales determinísticas.

9.1.2. Pruebas con *clips* musicales

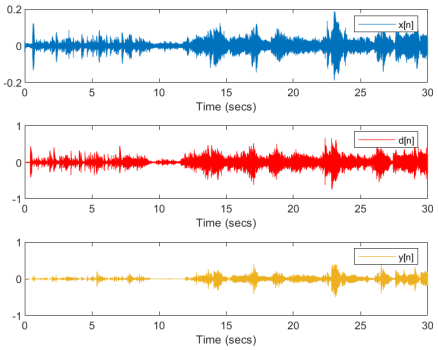
Un único instrumento



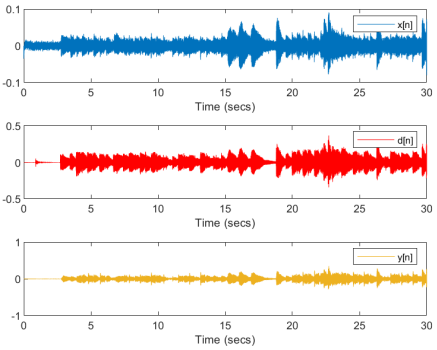
(a) *Bohemian Rhapsody*



(b) *Cadence*

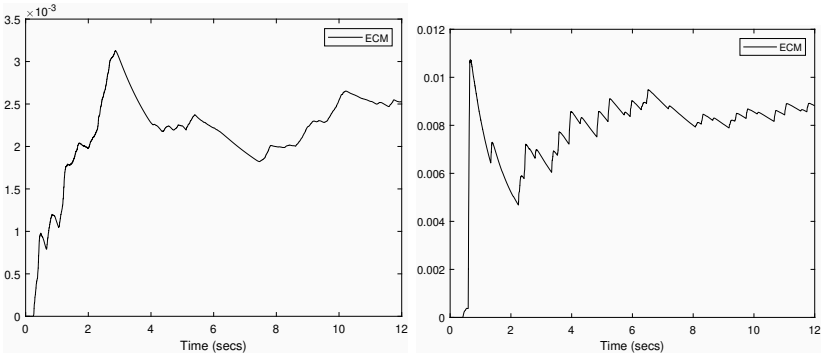


(c) *Peru*

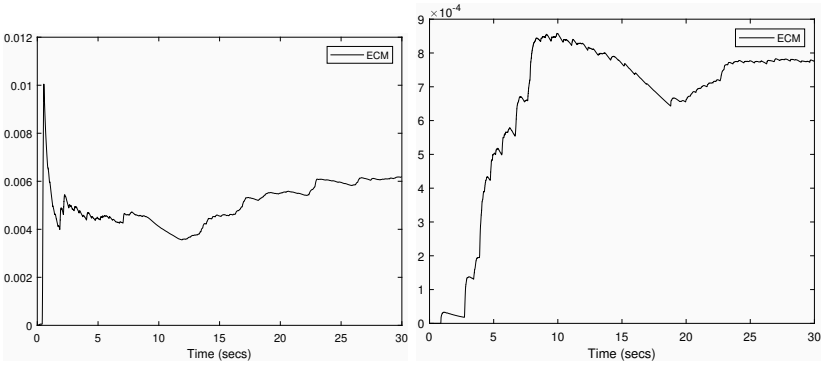


(d) *Week No. 8*

Figura 28: Señales del filtrado de clips de un solo instrumento por LMS estándar.

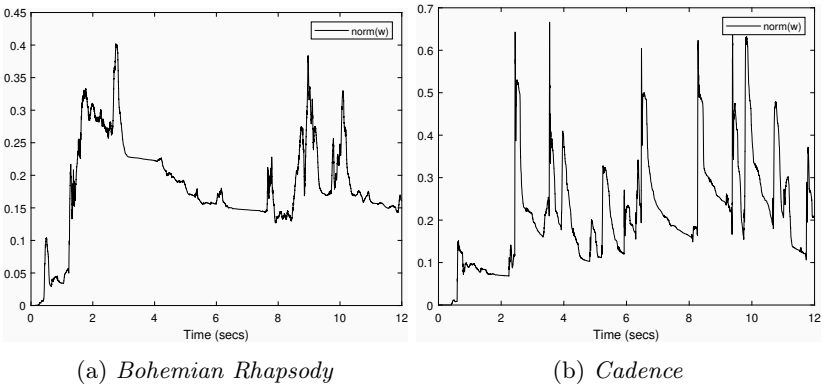


(a) *Bohemian Rhapsody* (b) *Cadence*

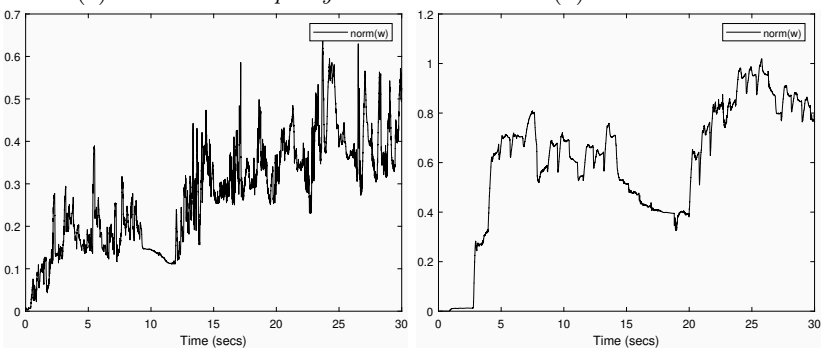


(c) *Peru* (d) *Week No. 8*

Figura 29: ECM del filtrado de clips de un solo instrumento por LMS estándar.

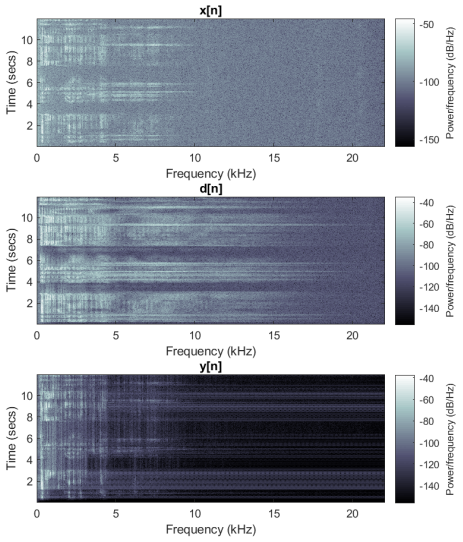


(a) *Bohemian Rhapsody* (b) *Cadence*

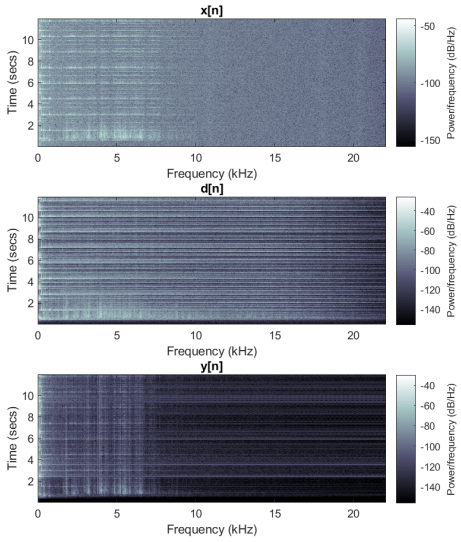


(c) *Peru* (d) *Week No. 8*

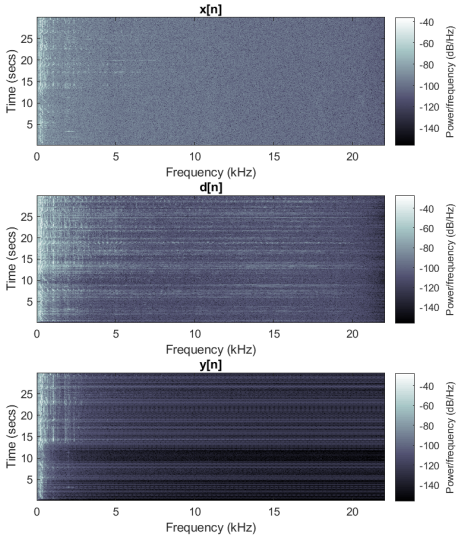
Figura 30: Magnitud del vector **w** del filtrado de clips de un solo instrumento por LMS estándar



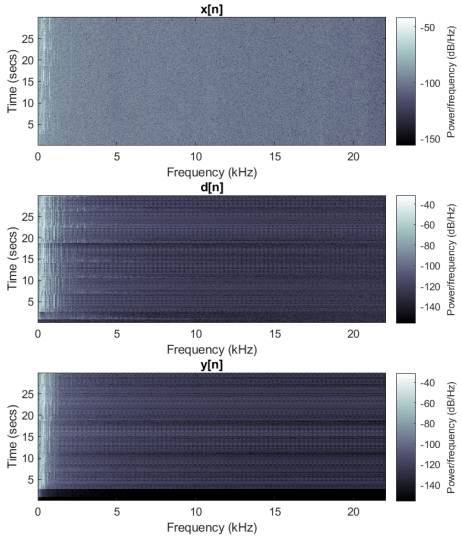
(a) *Bohemian Rhapsody*



(b) *Cadence*



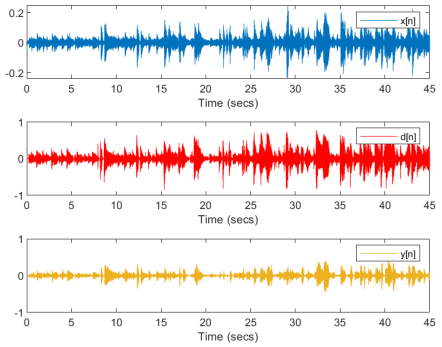
(c) *Peru*



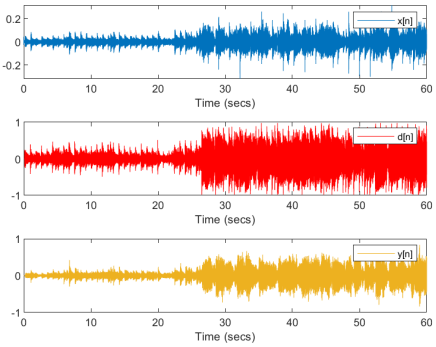
(d) *Week No. 8*

Figura 31: Espectrogramas del filtrado de clips de un solo instrumento por LMS estándar.

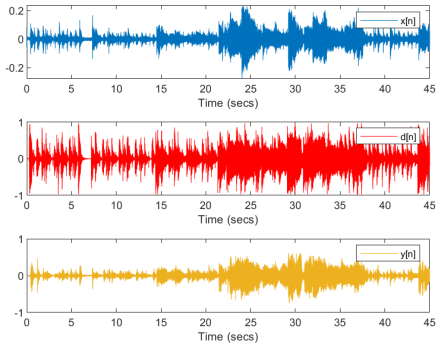
Varios instrumentos



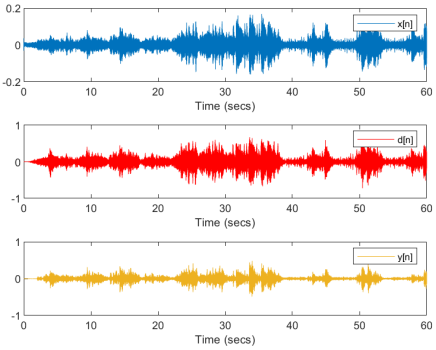
(a) *Lonely Cat*



(b) *Karma Police*

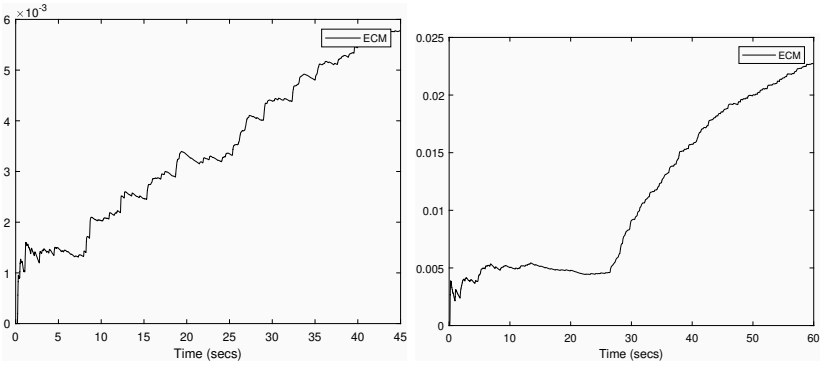


(c) *Atlantic Limited*

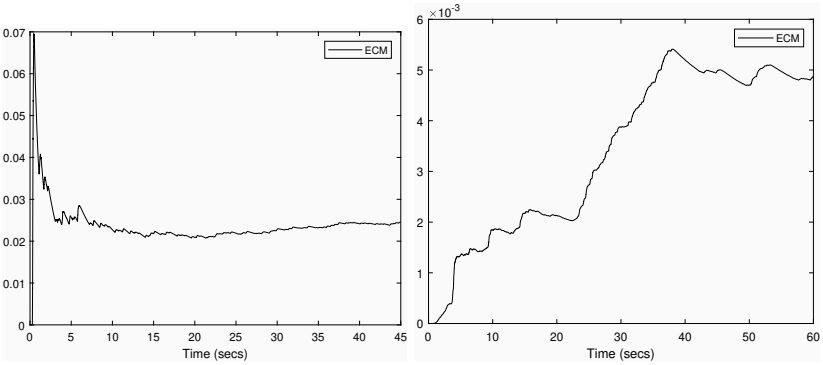


(d) *The Blue Danube*

Figura 32: Señales del filtrado de clips de varios instrumentos por LMS estándar.

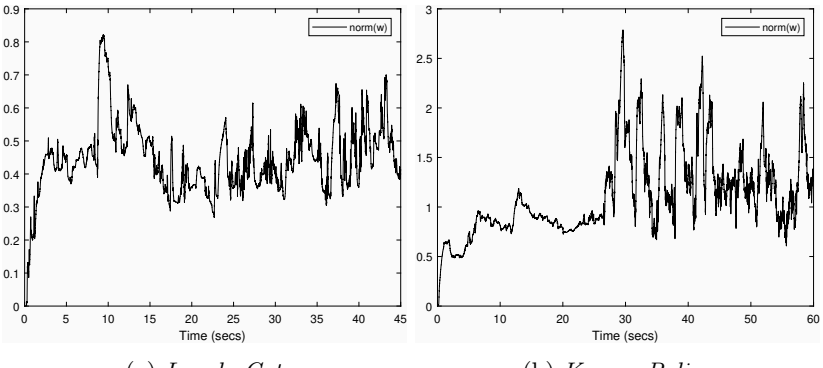


(a) *Lonely Cat* (b) *Karma Police*

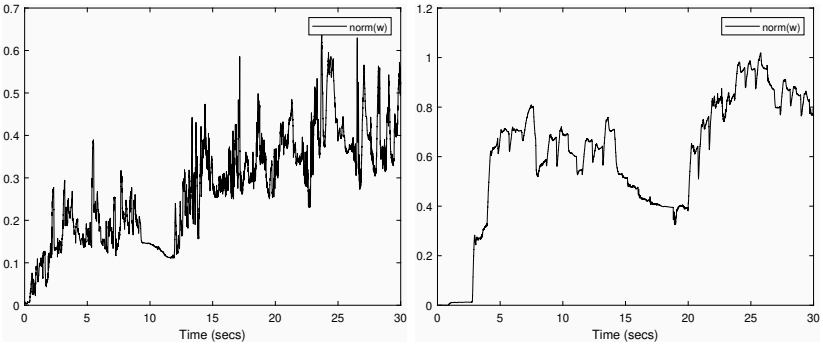


(c) *Atlantic Limited* (d) *The Blue Danube*

Figura 33: ECM del filtrado de clips de varios instrumentos por LMS estándar.

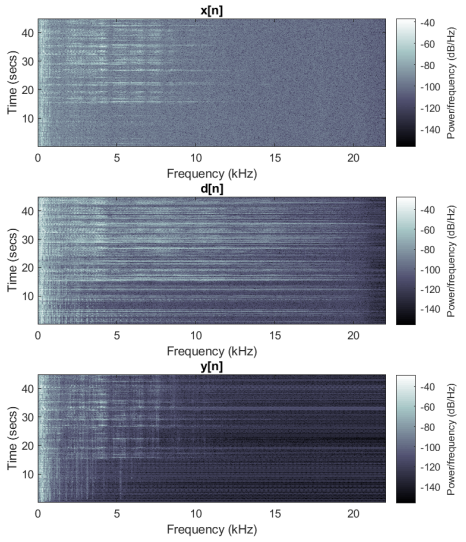


(a) *Lonely Cat* (b) *Karma Police*

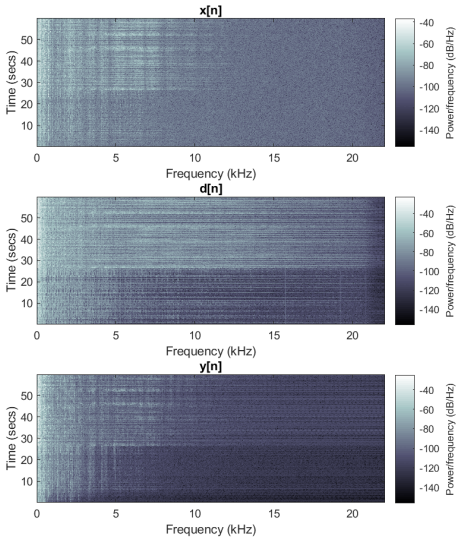


(c) *Atlantic Limited* (d) *The Blue Danube*

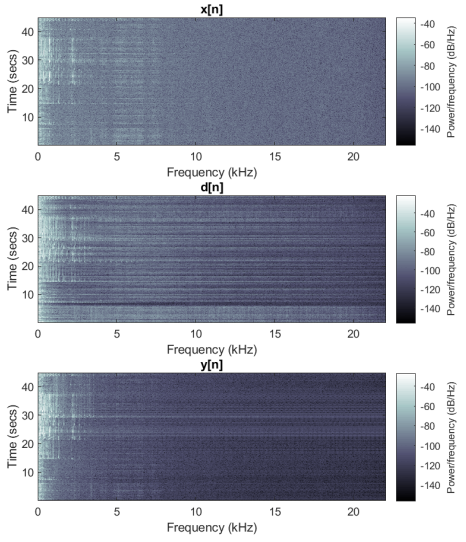
Figura 34: Magnitud del vector **w** del filtrado de clips de varios instrumentos por LMS estándar.



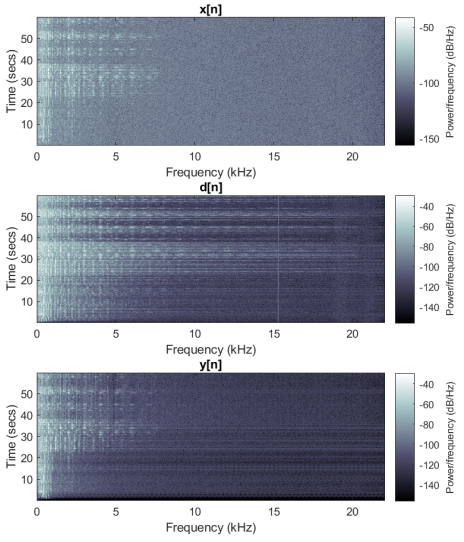
(a) *Lonely Cat*



(b) *Karma Police*



(c) *Atlantic Limited*



(d) *The Blue Danube*

Figura 35: Espectrogramas del filtrado de clips de varios instrumentos por LMS estándar.

ECM finales y máximas magnitudes de w

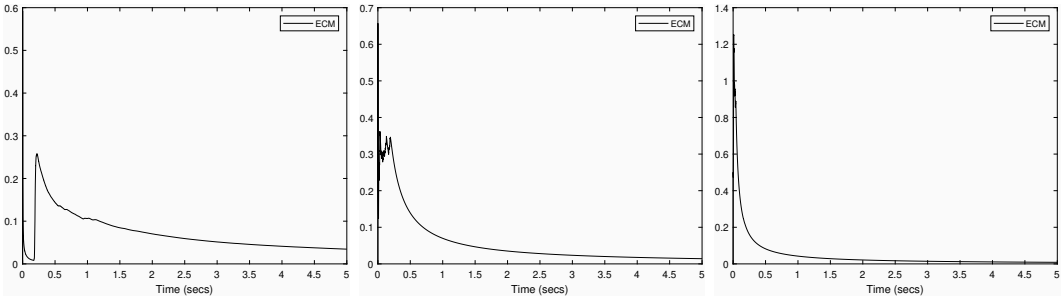
Señal	ECM final	Máx $\|\|\mathbf{w}\|\|$
Bohemian Rhapsody	0.0025287	0.40208
Cadence	0.0088115	0.66609
Peru	0.0061841	0.64251
Week No. 8	0.00077742	1.0192
Lonely Cat	0.005776	0.82207
Karma Police	0.022725	2.7876
Atlantic Limited	0.024499	2.133
The Blue Danube	0.0048655	0.93814
Promedio	0.0095	1.1763

Cuadro 6: ECM finales y máximas magnitudes de **w** alcanzados por el LMS estándar para los clips musicales.

9.2. LMS normalizado

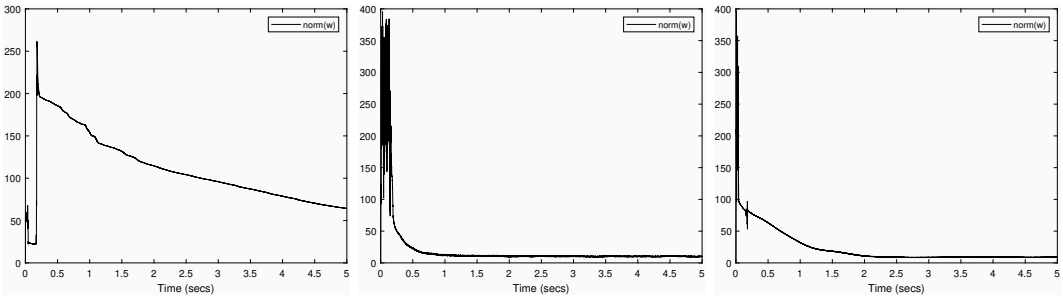
9.2.1. Pruebas con señales determinísticas

Ondas sinusoidales



(a) 250 Hz (b) 1 kHz (c) 10 kHz

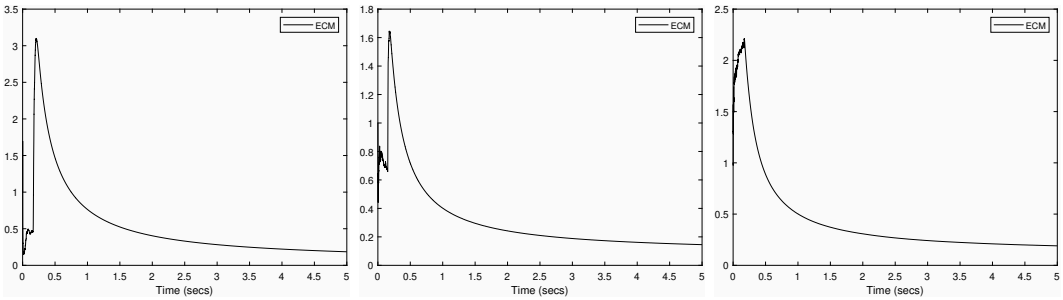
Figura 36: ECM del filtrado de sinusoides por LMS normalizado.



(a) 250 Hz (b) 1 kHz (c) 10 kHz

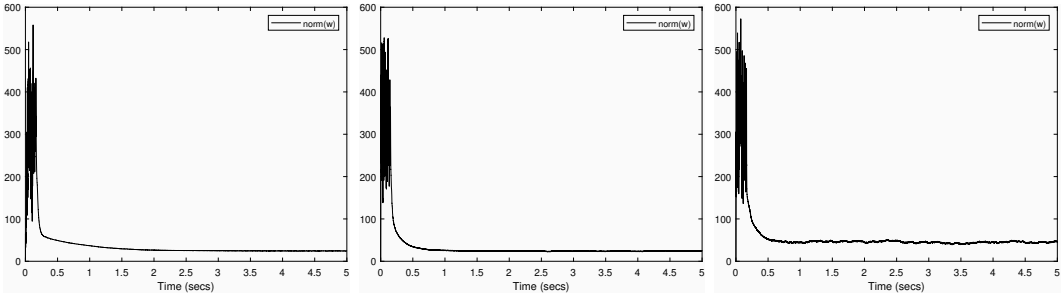
Figura 37: Magnitud del vector **w** del filtrado de sinusoides por LMS normalizado.

Ondas cuadradas



(a) 250 Hz (b) 1 kHz (c) 10 kHz

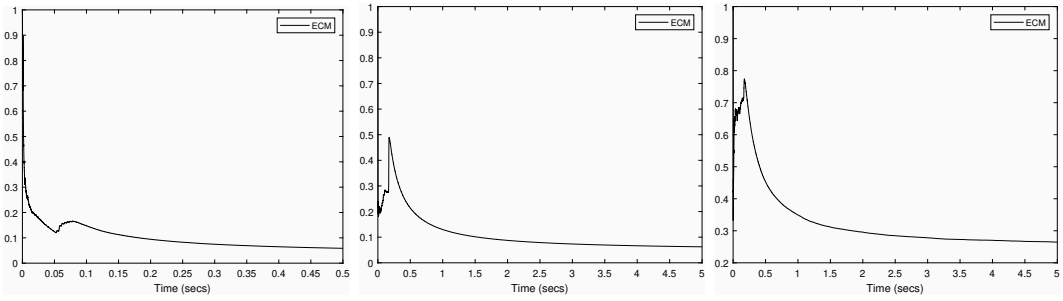
Figura 38: ECM del filtrado de ondas cuadradas por LMS normalizado.



(a) 250 Hz (b) 1 kHz (c) 10 kHz

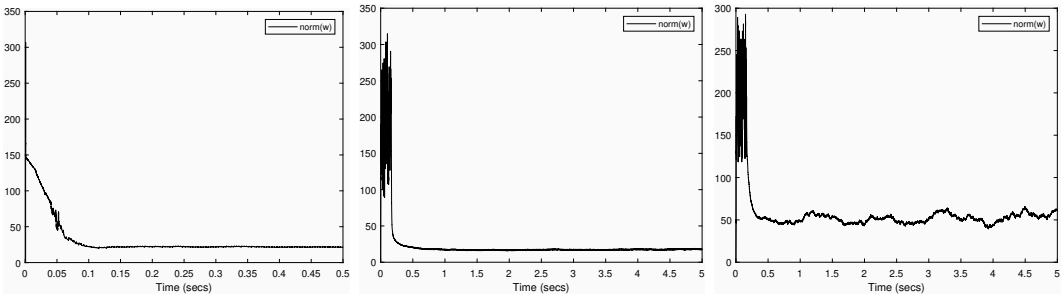
Figura 39: Magnitud del vector **w** del filtrado de ondas cuadradas por LMS normalizado.

Ondas diente de sierra



(a) 250 Hz (b) 1 kHz (c) 10 kHz

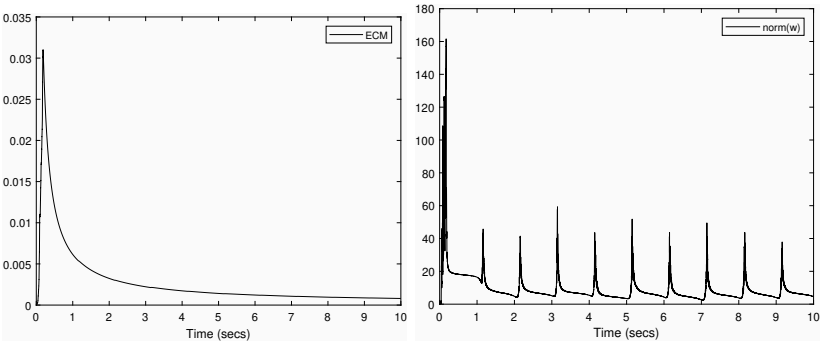
Figura 40: ECM del filtrado de ondas diente de sierra por LMS normalizado.



(a) 250 Hz (b) 1 kHz (c) 10 kHz

Figura 41: Magnitud del vector **w** del filtrado de ondas diente de sierra por LMS normalizado.

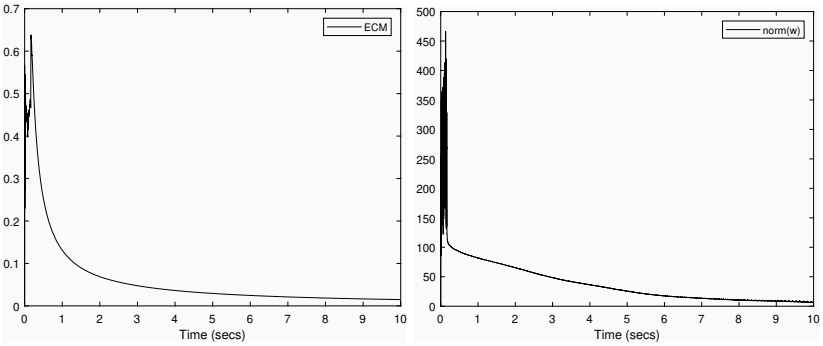
Onda AM



(a) ECM. (b) Magnitud del vector **w**.

Figura 42: Resultados del filtrado de un sinusoide de amplitud modulada por LMS normalizado.

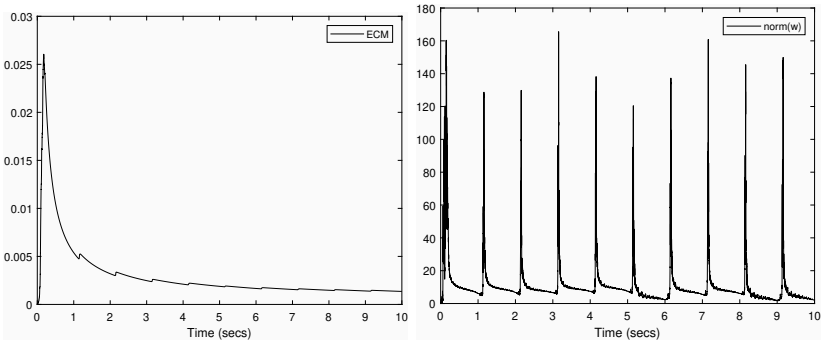
Onda FM



(a) ECM. (b) Magnitud del vector **w**.

Figura 43: Resultados del filtrado de un sinusoide de frecuencia modulada por LMS normalizado.

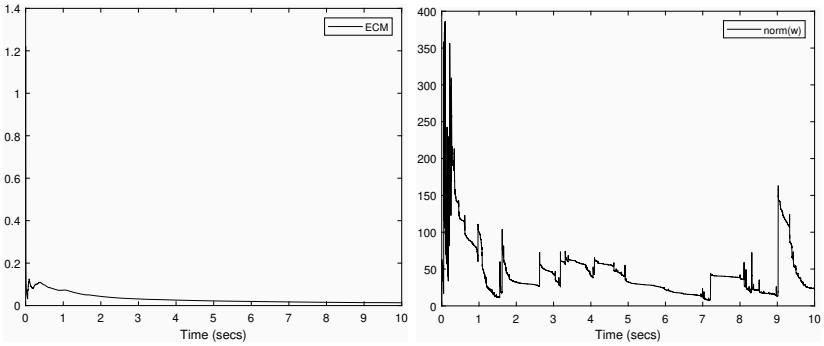
Onda AM y FM



(a) ECM. (b) Magnitud del vector **w**.

Figura 44: Resultados del filtrado de un sinusoide de amplitud y frecuencia modulada por LMS normalizado.

Barrido sinusoidal



(a) ECM. (b) Magnitud del vector **w**.

Figura 45: Resultados del filtrado de un barrido sinusoidal (de 20 Hz a 10 kHz) por LMS normalizado.

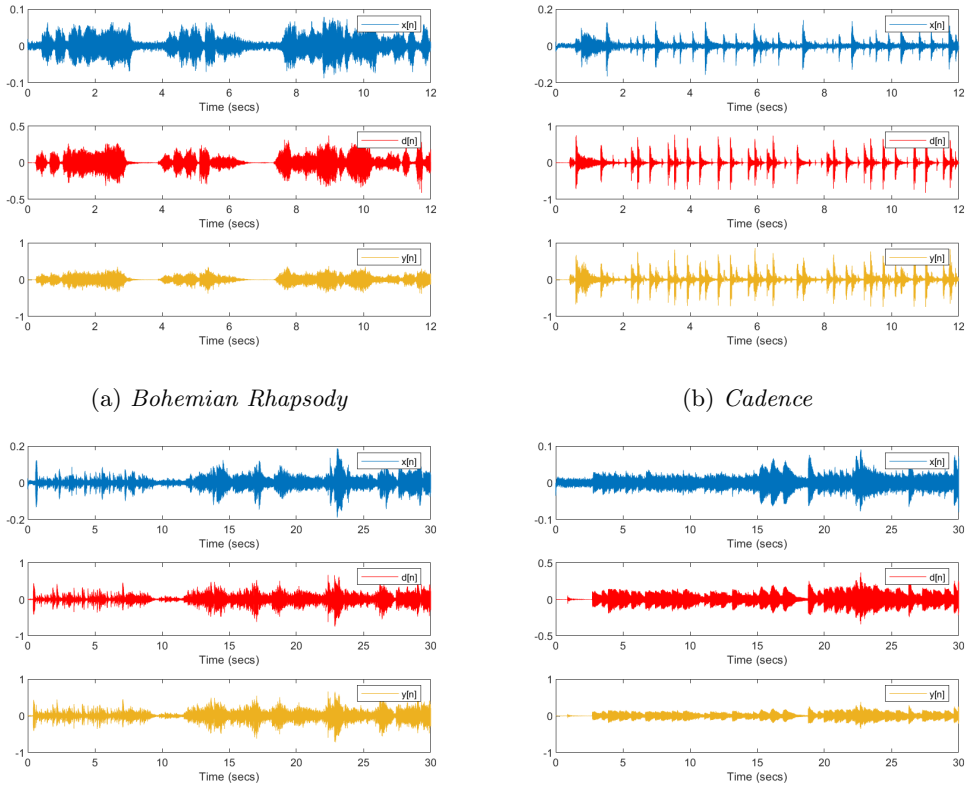
ECM finales y máximas magnitudes de w

Señal	ECM final	Máxima magnitud **w**
Onda sinusoidal 250 Hz	0.034502	261.5551
Onda sinusoidal 1 kHz	0.0142	395.4416
Onda sinusoidal 10 kHz	0.0086	396.6694
Onda cuadrada 250 Hz	0.1852	558.1159
Onda cuadrada 1 kHz	0.14531	527.9163
Onda cuadrada 10 kHz	0.19034	572.5863
Onda diente de sierra 250 Hz	0.034199	307.0588
Onda diente de sierra 1 kHz	0.062454	314.822
Onda diente de sierra 10 kHz	0.26455	292.9779
Onda AM	0.00080252	161.5862
Onda FM	0.014863	466.2694
Onda AM y FM	0.0013539	165.5719
Barrido sinusoidal	0.013315	386.1382
Promedio	0.0746	369.7469

Cuadro 7: ECM finales y máximas magnitudes de **w** alcanzados por el LMS normalizado para señales determinísticas.

9.2.2. Pruebas con *clips* musicales

Un único instrumento



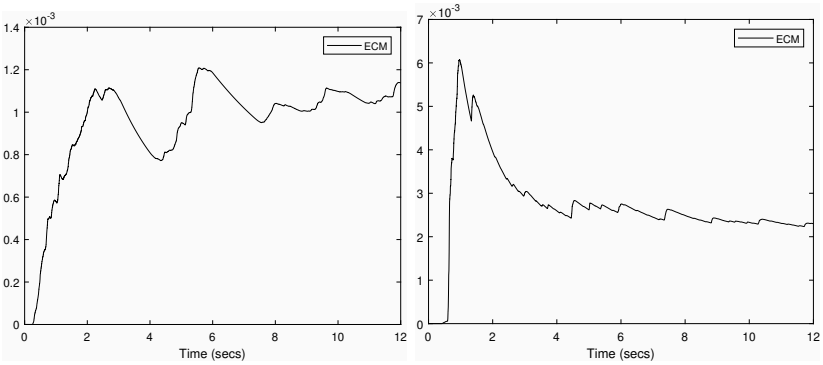
(a) *Bohemian Rhapsody*

(b) *Cadence*

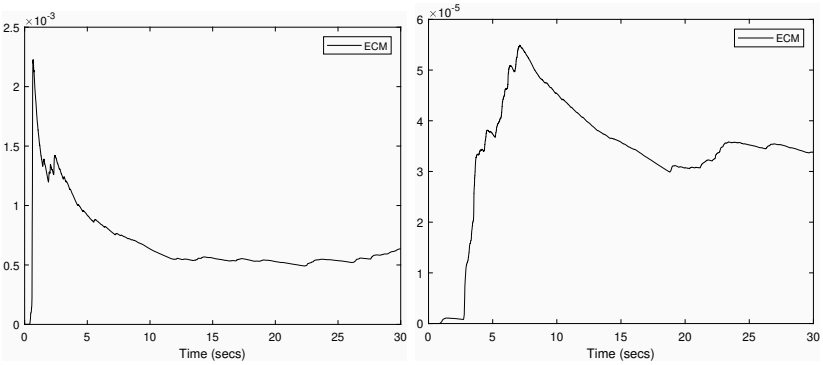
(c) *Peru*

(d) *Week No. 8*

Figura 46: Señales del filtrado de clips de un solo instrumento por LMS normalizado.

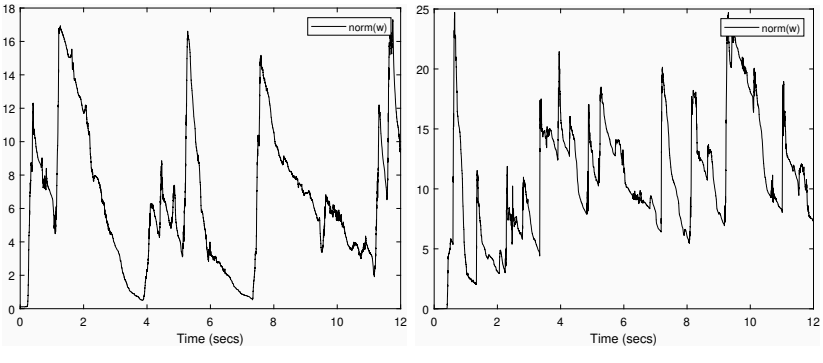


(a) *Bohemian Rhapsody* (b) *Cadence*

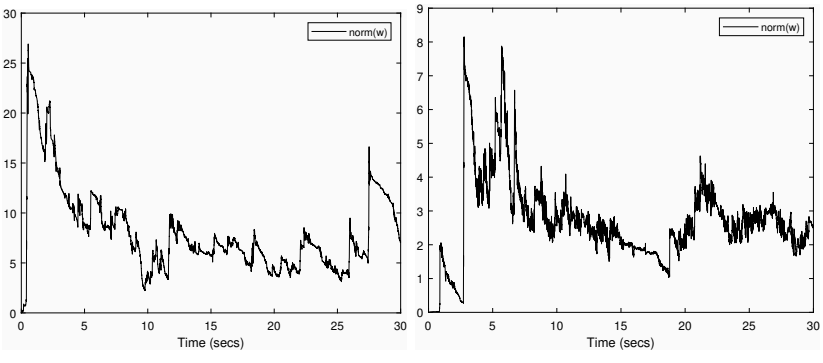


(c) *Peru* (d) *Week No. 8*

Figura 47: ECM del filtrado de clips de un solo instrumento por LMS normalizado.

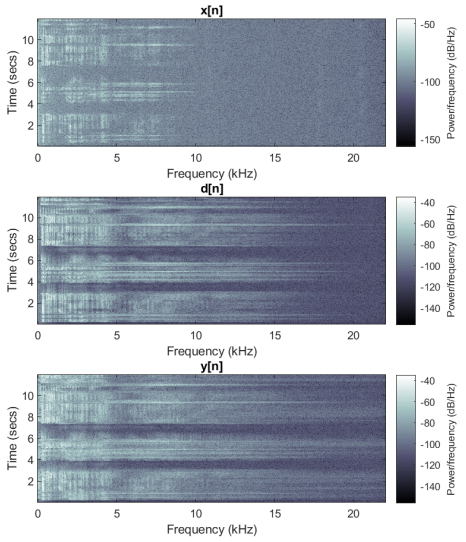


(a) *Bohemian Rhapsody* (b) *Cadence*

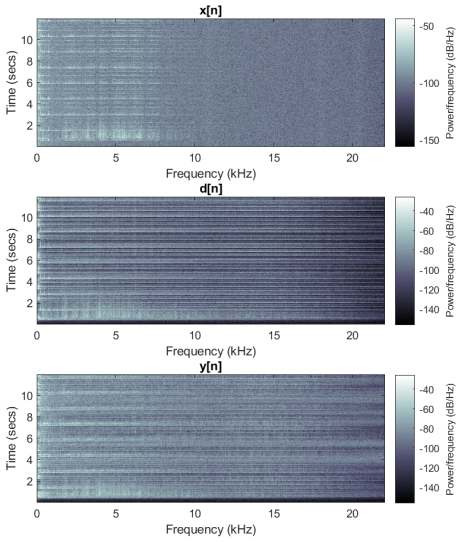


(c) *Peru* (d) *Week No. 8*

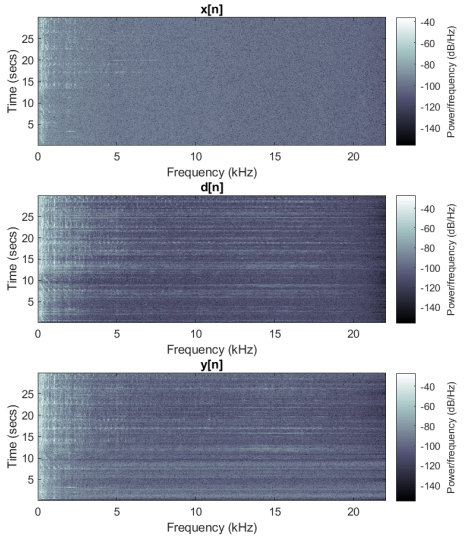
Figura 48: Magnitud del vector **w** del filtrado de clips de un solo instrumento por LMS normalizado



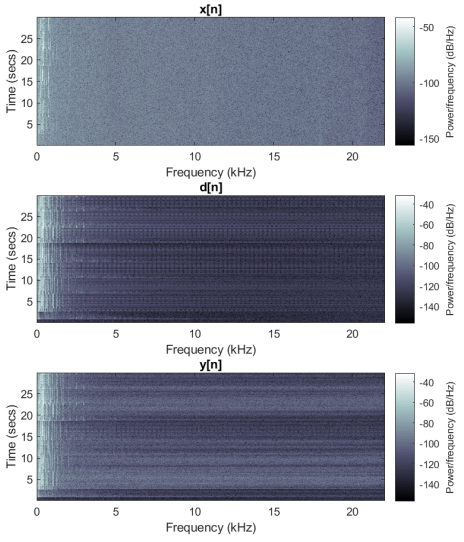
(a) *Bohemian Rhapsody*



(b) *Cadence*



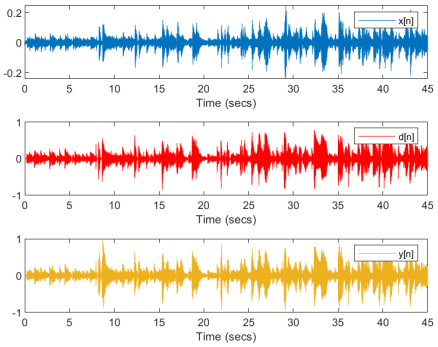
(c) *Peru*



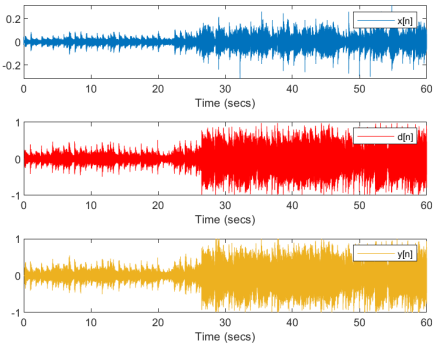
(d) *Week No. 8*

Figura 49: Espectrogramas del filtrado de clips de un solo instrumento por LMS normalizado.

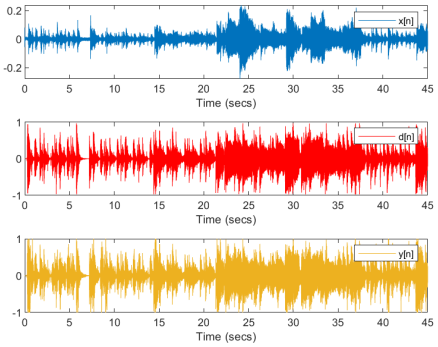
Varios instrumentos



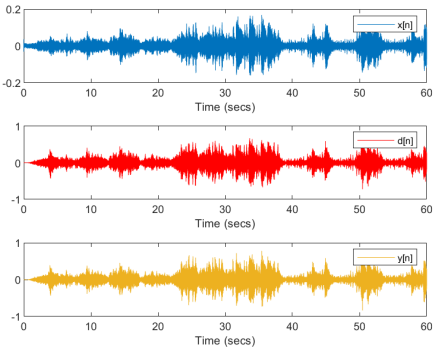
(a) *Lonely Cat*



(b) *Karma Police*

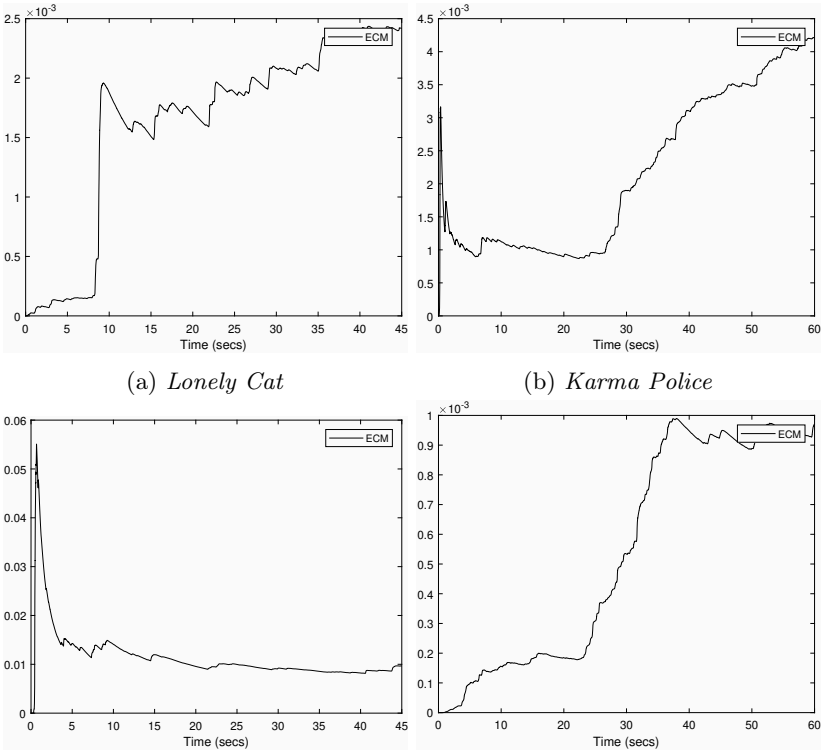


(c) *Atlantic Limited*



(d) *The Blue Danube*

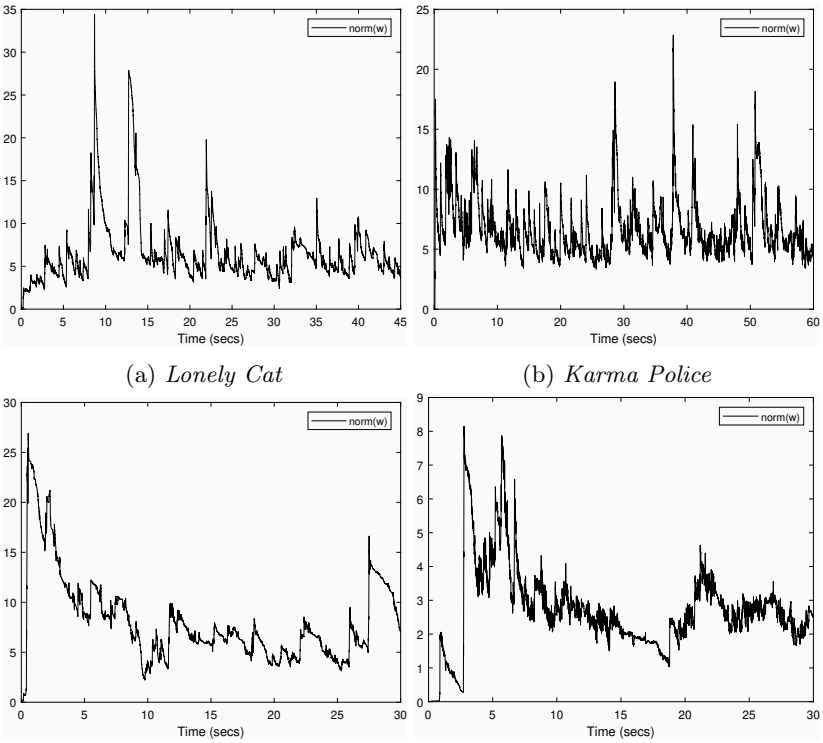
Figura 50: Señales del filtrado de clips de varios instrumentos por LMS normalizado.



(a) *Lonely Cat* (b) *Karma Police*

(c) *Atlantic Limited* (d) *The Blue Danube*

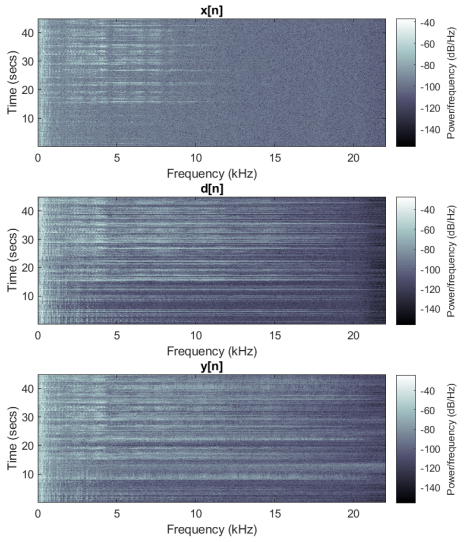
Figura 51: ECM del filtrado de clips de varios instrumentos por LMS normalizado.



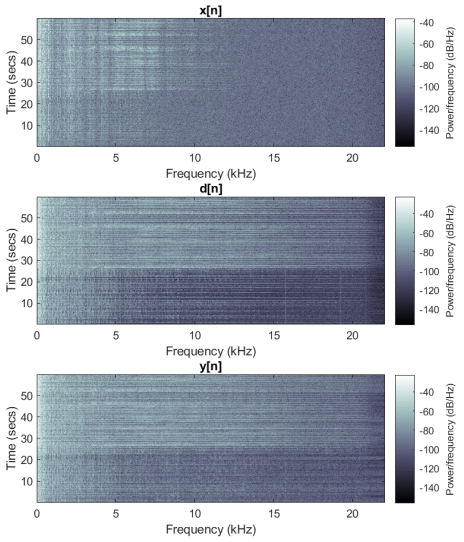
(a) *Lonely Cat* (b) *Karma Police*

(c) *Atlantic Limited* (d) *The Blue Danube*

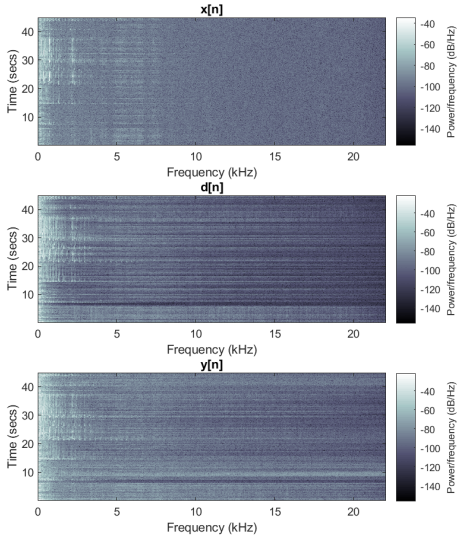
Figura 52: Magnitud del vector **w** del filtrado de clips de varios instrumentos por LMS normalizado.



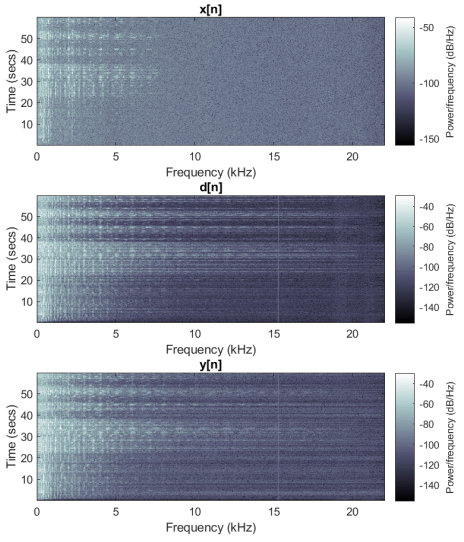
(a) *Lonely Cat*



(b) *Karma Police*



(c) *Atlantic Limited*



(d) *The Blue Danube*

Figura 53: Espectrogramas del filtrado de clips de varios instrumentos por LMS normalizado.

ECM finales y máximas magnitudes de w

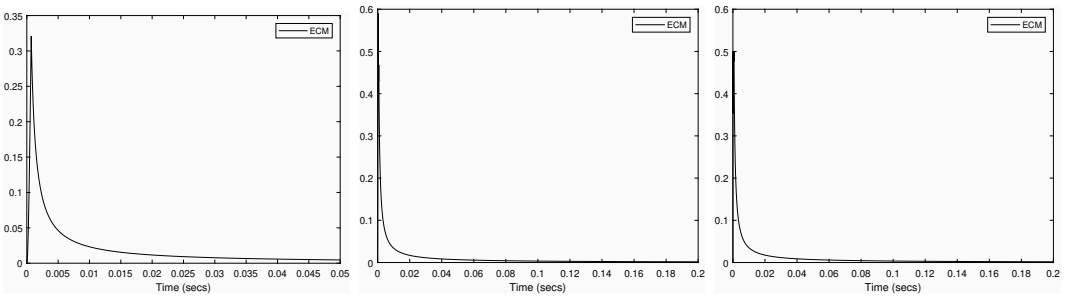
Señal	ECM final	Máx $\|\|\mathbf{w}\|\|$
Bohemian Rhapsody	0.001138	17.3041
Cadence	0.0022991	24.7189
Peru	0.00063522	26.9124
Week No. 8	3.3792e-05	8.1481
Lonely Cat	0.0024177	34.3625
Karma Police	0.0042097	22.8534
Atlantic Limited	0.0096533	102.6302
The Blue Danube	0.00097149	11.2464
Promedio	0.0027	31.0220

Cuadro 8: ECM finales y máximas magnitudes de **w** alcanzados por el LMS normalizado para los clips musicales.

9.3. RLS convencional

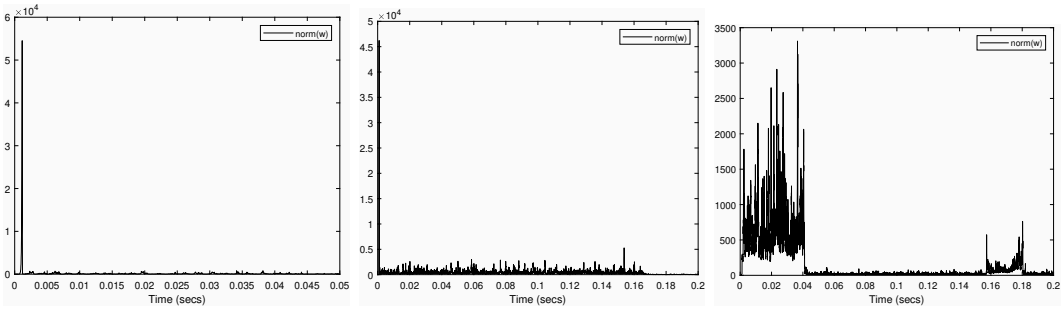
9.3.1. Pruebas con señales determinísticas

Ondas sinusoidales



(a) 250 Hz (b) 1 kHz (c) 10 kHz

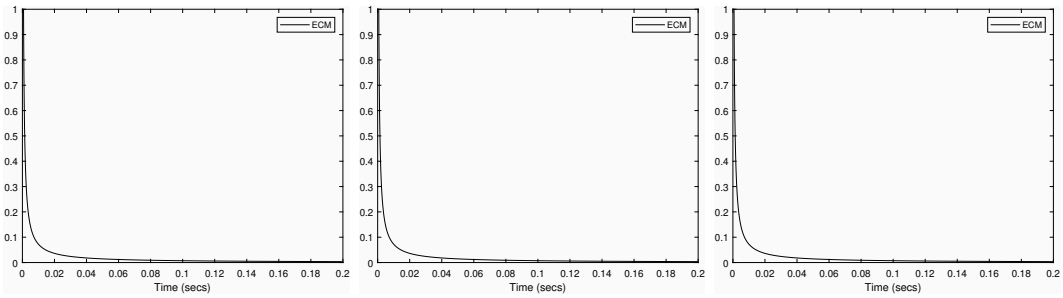
Figura 54: ECM del filtrado de sinusoides por RLS convencional.



(a) 250 Hz (b) 1 kHz (c) 10 kHz

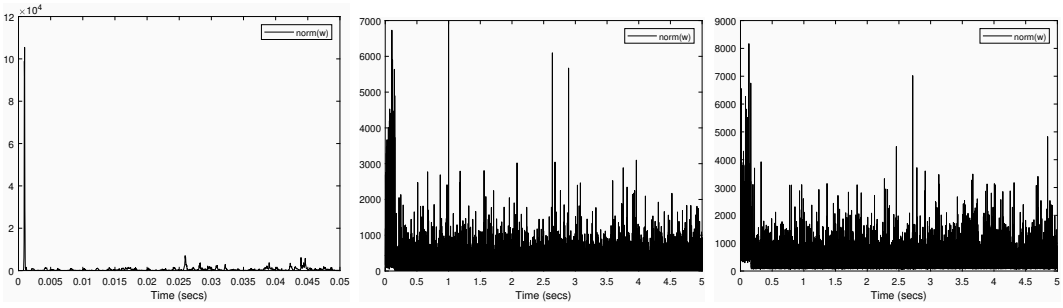
Figura 55: Magnitud del vector **w** del filtrado de sinusoides por RLS convencional.

Ondas cuadradas



(a) 250 Hz (b) 1 kHz (c) 10 kHz

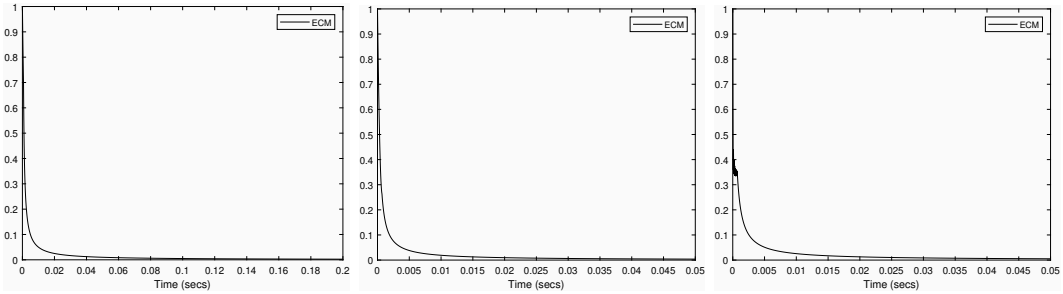
Figura 56: ECM del filtrado de ondas cuadradas por RLS convencional.



(a) 250 Hz (b) 1 kHz (c) 10 kHz

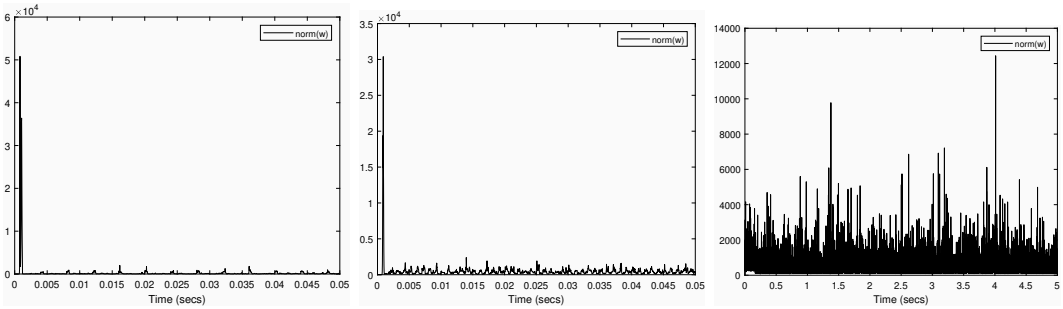
Figura 57: Magnitud del vector **w** del filtrado de ondas cuadradas por RLS convencional.

Ondas diente de sierra



(a) 250 Hz (b) 1 kHz (c) 10 kHz

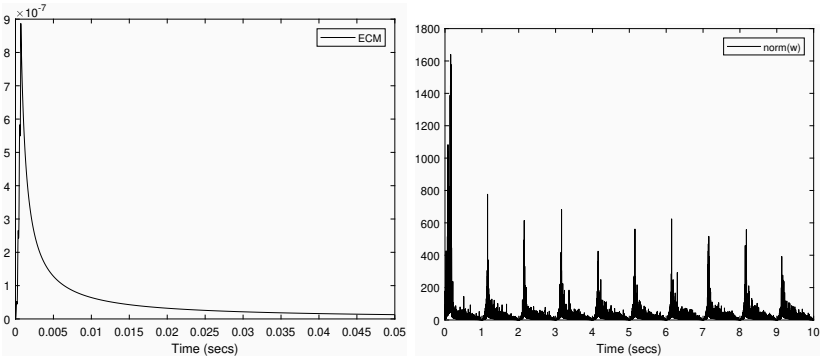
Figura 58: ECM del filtrado de ondas diente de sierra por RLS convencional.



(a) 250 Hz (b) 1 kHz (c) 10 kHz

Figura 59: Magnitud del vector **w** del filtrado de ondas diente de sierra por RLS convencional.

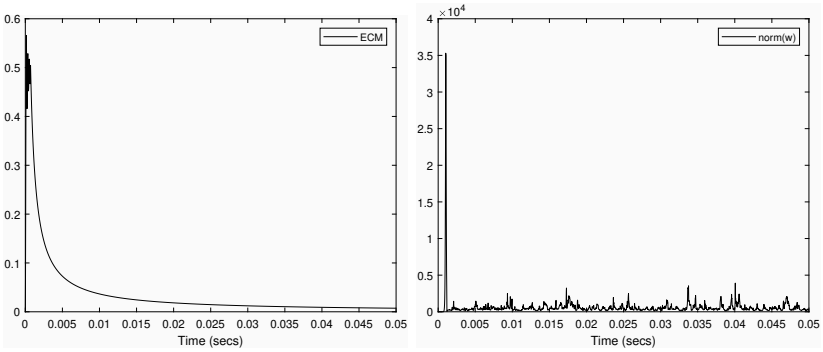
Onda AM



(a) ECM. (b) Magnitud del vector **w**.

Figura 60: Resultados del filtrado de un sinusoide de amplitud modulada por RLS convencional.

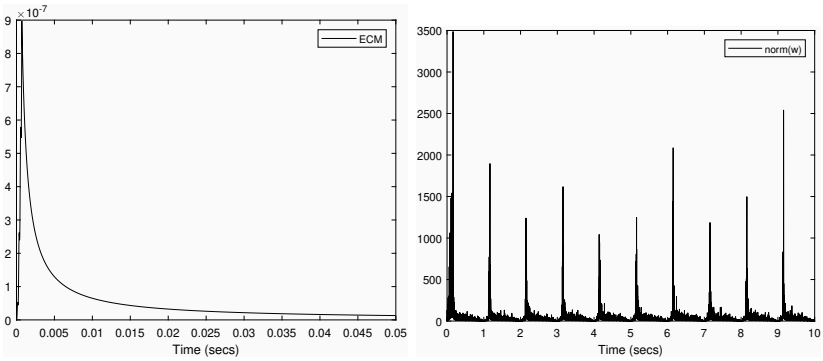
Onda FM



(a) ECM. (b) Magnitud del vector **w**.

Figura 61: Resultados del filtrado de un sinusoide de frecuencia modulada por RLS convencional.

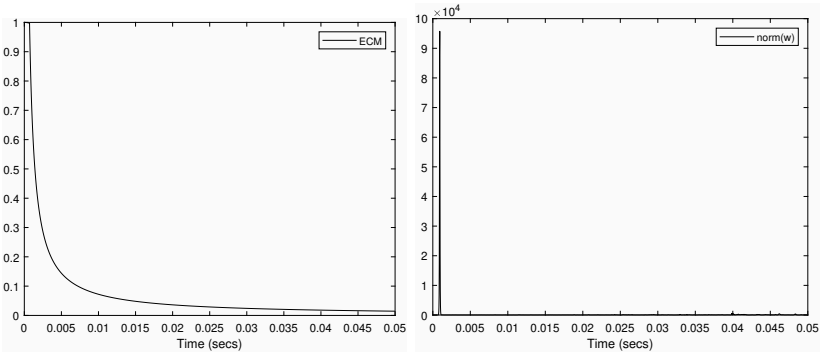
Onda AM y FM



(a) ECM. (b) Magnitud del vector **w**.

Figura 62: Resultados del filtrado de un sinusoide de amplitud y frecuencia modulada por RLS convencional.

Barrido sinusoidal



(a) ECM. (b) Magnitud del vector **w**.

Figura 63: Resultados del filtrado de un barrido sinusoidal (de 20 Hz a 10 kHz) por RLS convencional.

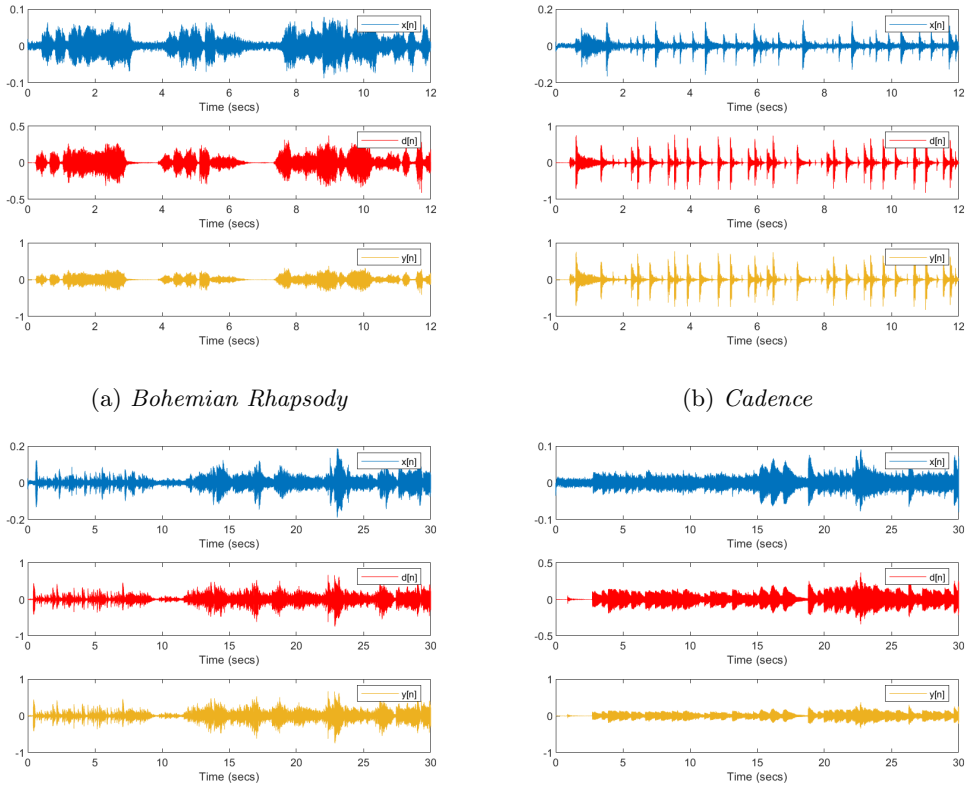
ECM finales y máximas magnitudes de w

Señal	ECM final	Máxima magnitud **w**
Onda sinusoidal 250 Hz	4.6534e-05	54521.5602
Onda sinusoidal 1 kHz	6.7994e-05	46217.6002
Onda sinusoidal 10 kHz	7.0312e-05	3310.5286
Onda cuadrada 250 Hz	0.00014513	105448.8674
Onda cuadrada 1 kHz	0.00014512	6994.4512
Onda cuadrada 10 kHz	0.00014512	8169.6459
Onda diente de sierra 250 Hz	0.00010019	50832.5572
Onda diente de sierra 1 kHz	3.8253e-05	30387.0989
Onda diente de sierra 10 kHz	5.1357e-05	12440.499
Onda AM	6.7192e-11	1640.5796
Onda FM	3.6613e-05	35302.1942
Onda AM y FM	7.3658e-11	3486.194
Barrido sinusoidal	7.2364e-05	95737.2772
Promedio	7.0692e-05	3.4961e+04

Cuadro 9: ECM finales y máximas magnitudes de **w** alcanzados por el RLS convencional para señales determinísticas.

9.3.2. Pruebas con *clips* musicales

Un único instrumento



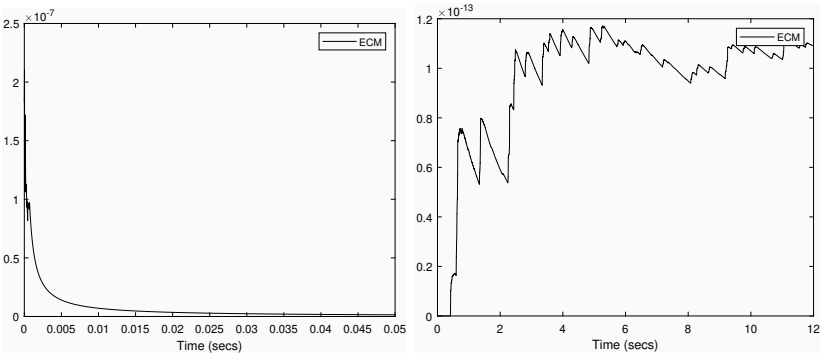
(a) *Bohemian Rhapsody*

(b) *Cadence*

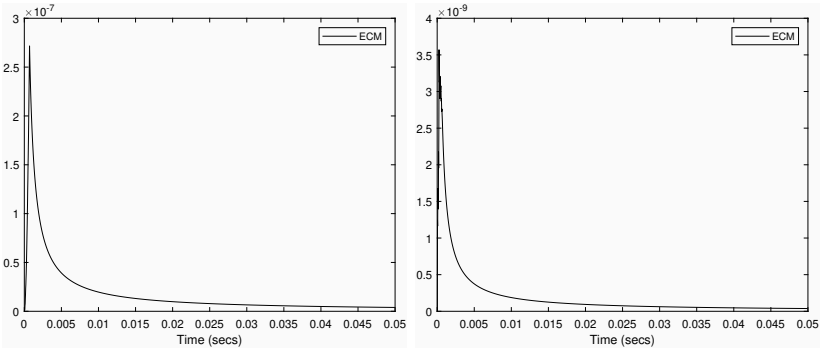
(c) *Peru*

(d) *Week No. 8*

Figura 64: Señales del filtrado de clips de un solo instrumento por RLS convencional.

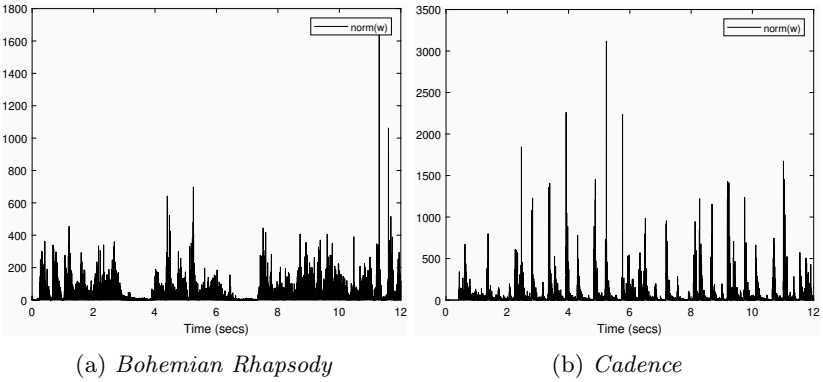


(a) *Bohemian Rhapsody* (b) *Cadence*

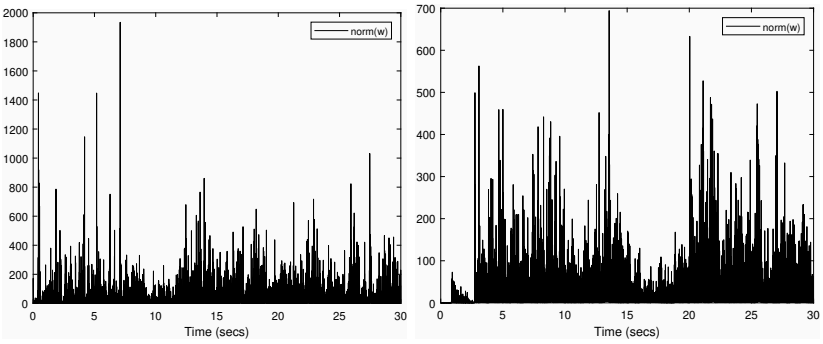


(c) *Peru* (d) *Week No. 8*

Figura 65: ECM del filtrado de clips de un solo instrumento por RLS convencional.

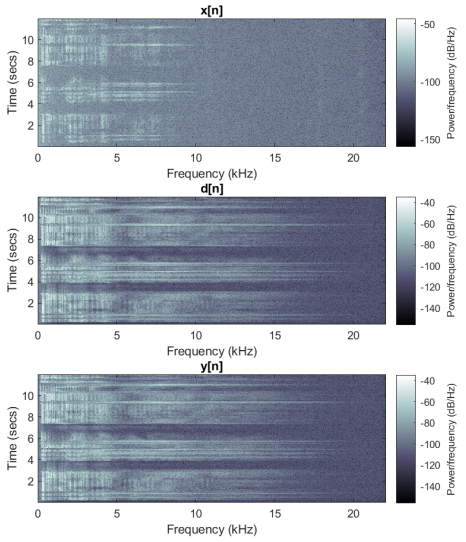


(a) *Bohemian Rhapsody* (b) *Cadence*

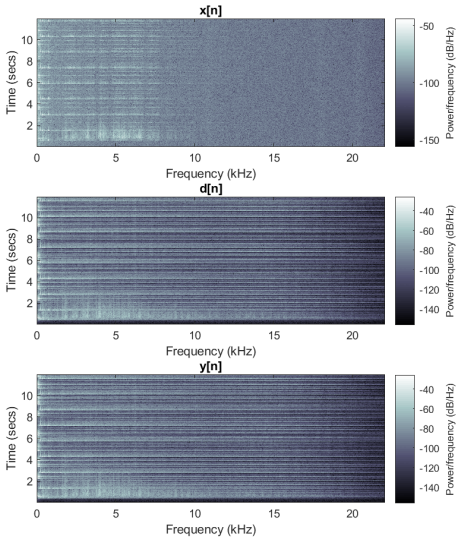


(c) *Peru* (d) *Week No. 8*

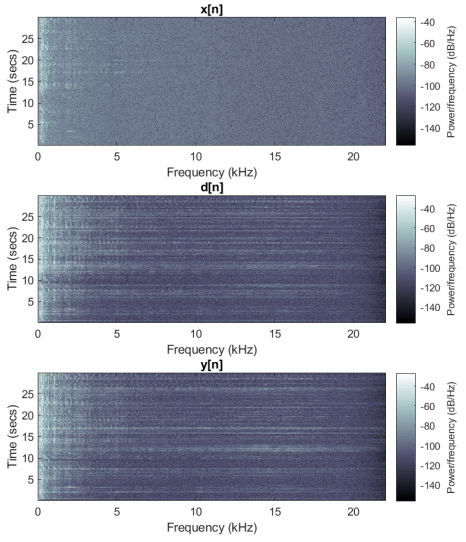
Figura 66: Magnitud del vector **w** del filtrado de clips de un solo instrumento por RLS convencional



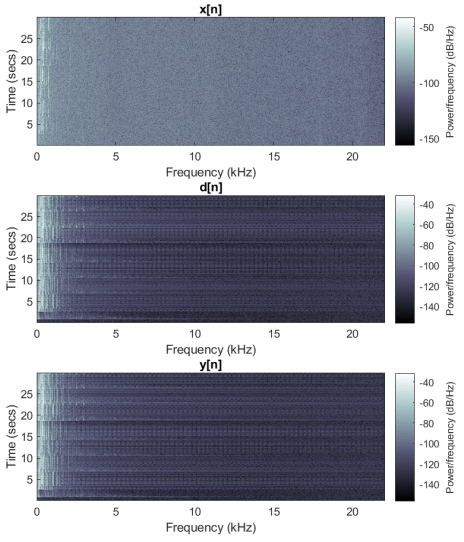
(a) *Bohemian Rhapsody*



(b) *Cadence*



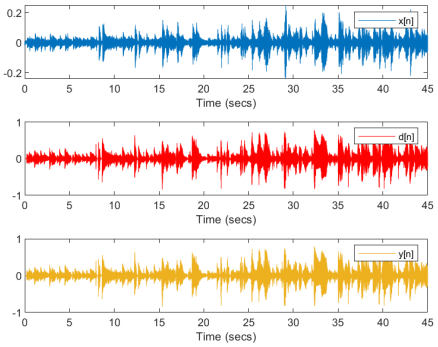
(c) *Peru*



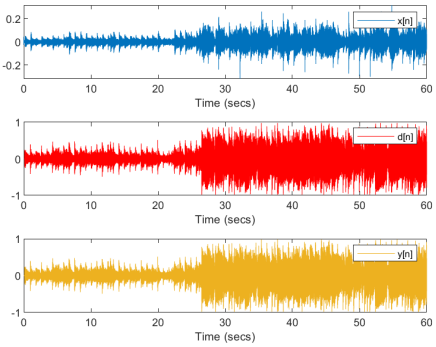
(d) *Week No. 8*

Figura 67: Espectrogramas del filtrado de clips de un solo instrumento por RLS convencional.

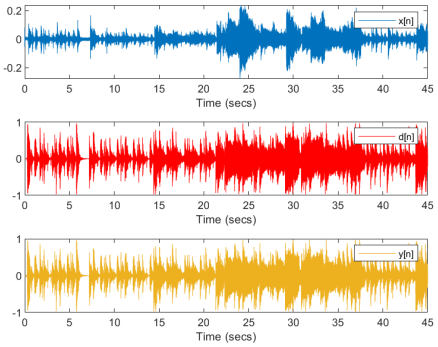
Varios instrumentos



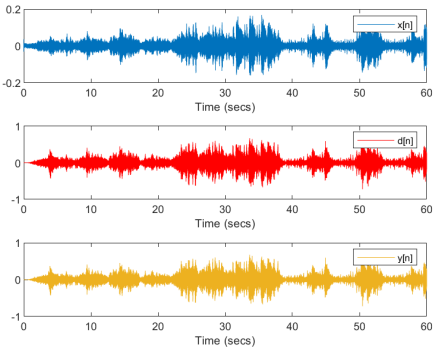
(a) *Lonely Cat*



(b) *Karma Police*

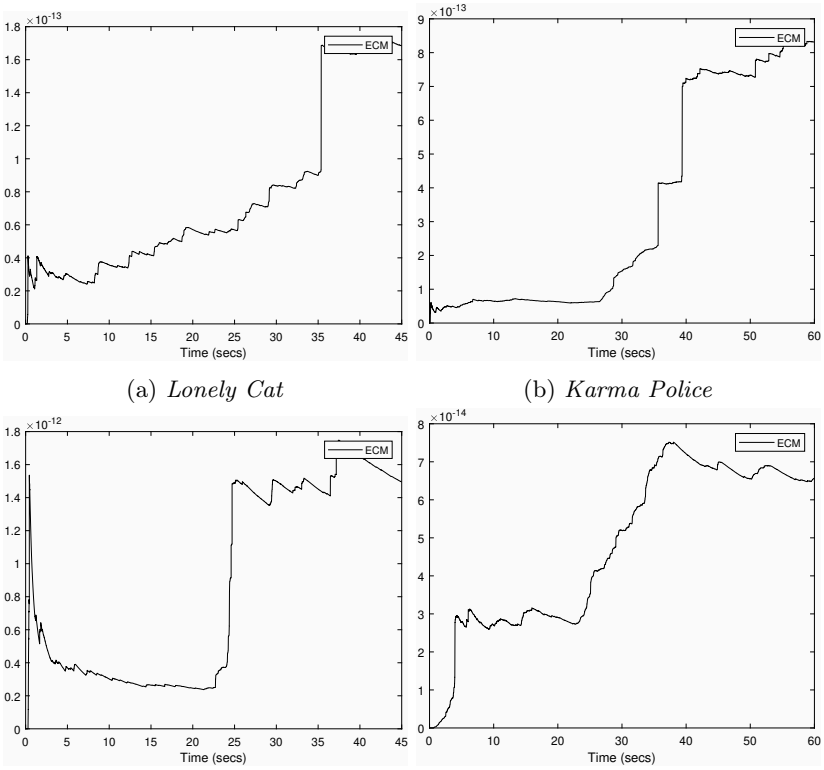


(c) *Atlantic Limited*



(d) *The Blue Danube*

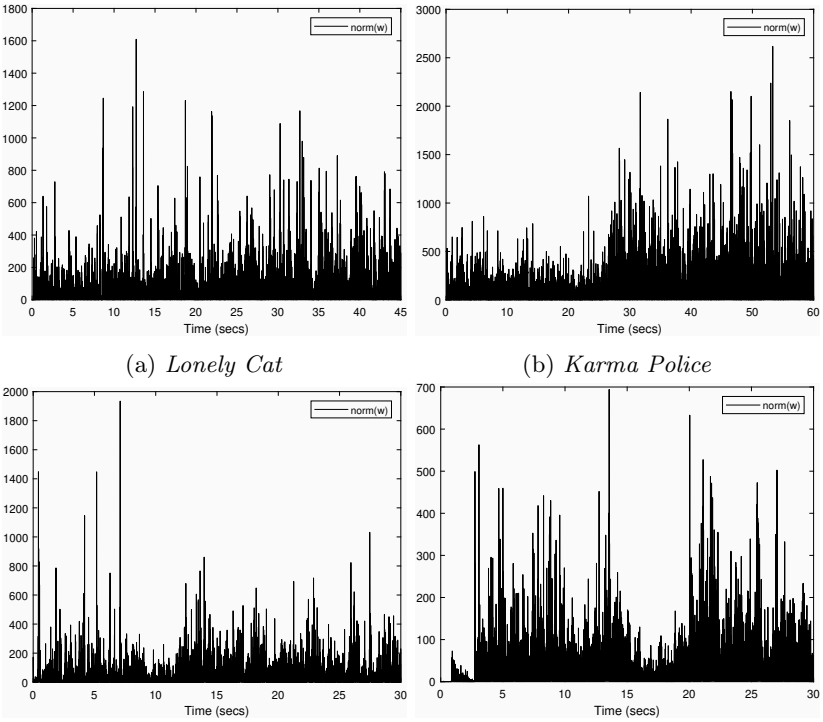
Figura 68: Señales del filtrado de clips de varios instrumentos por RLS convencional.



(a) *Lonely Cat* (b) *Karma Police*

(c) *Atlantic Limited* (d) *The Blue Danube*

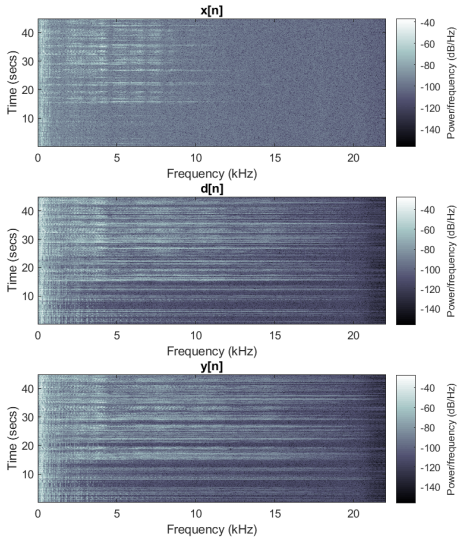
Figura 69: ECM del filtrado de clips de varios instrumentos por RLS convencional.



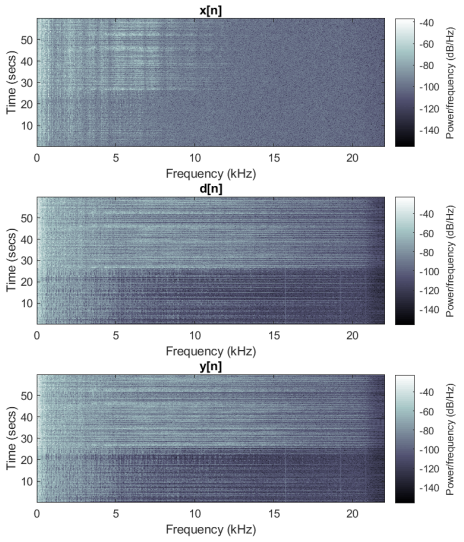
(a) *Lonely Cat* (b) *Karma Police*

(c) *Atlantic Limited* (d) *The Blue Danube*

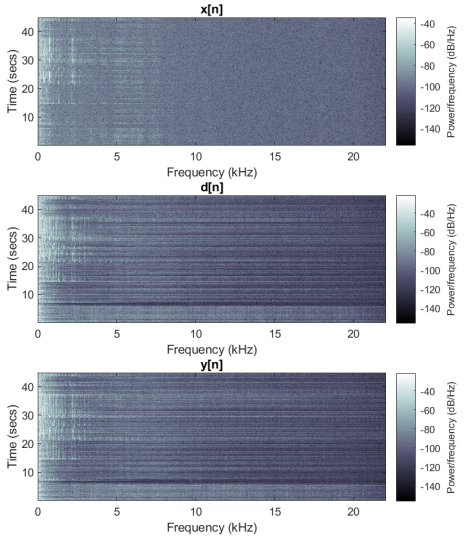
Figura 70: Magnitud del vector **w** del filtrado de clips de varios instrumentos por RLS convencional.



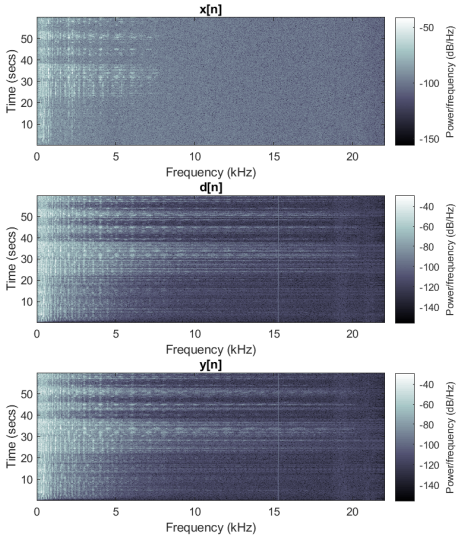
(a) *Lonely Cat*



(b) *Karma Police*



(c) *Atlantic Limited*



(d) *The Blue Danube*

Figura 71: Espectrogramas del filtrado de clips de varios instrumentos por RLS convencional.

ECM finales y Máx ||w||

Señal	ECM final	Máx ||**w**||
Bohemian Rhapsody	5.9192e-12	1648.5892
Cadence	1.0892e-13	3117.9246
Peru	6.7246e-12	1934.788
Week No. 8	8.0166e-14	694.1968
Lonely Cat	1.6833e-13	1609.5396
Karma Police	8.308e-13	2617.7971
Atlantic Limited	1.4942e-12	3988.7618
The Blue Danube	6.5714e-14	1576.4592
Promedio	1.9240e-12	2.1485e+03

Cuadro 10: ECM finales y máximas magnitudes de **w** alcanzados por el RLS convencional para los clips musicales.

9.4. Discusión de resultados

9.4.1. Evolución del ECM en el filtrado de señales determinísticas

Un denominador común que tuvieron los tres tipos de filtros fue el comportamiento de ECM al tratar con las señales determinísticas. Para las ondas básicas (sinusoidales, cuadradas y diente de sierra) los tres filtros presentaron el mismo decaimiento exponencial para esta medida de error. Para el resto de señales, moduladas y el barrido frecuencial, únicamente el RLS conservó la forma para todos los casos. El LMS normalizado tuvo la excepción con el barrido sinusoidal (Figura 45) y el LMS estándar solo lo volvió a presentar en la onda de frecuencia modulada (Figura 25). Se resalta además el cambio en escala para la gráficas del ECM del filtro RLS, la cual se muestra en tiempos de 0.2 y 0.05 segundos; este cambio deja ver la velocidad con la que el RLS logró emparejar la señal $y[n]$ con $d[n]$. El porqué de la evolución del ECM para cada filtro se analiza más a detalle en la sección "Robustez del RLS y el LMS".

9.4.2. Evolución en la magnitud de w en el filtrado de señales determinísticas

La magnitud del vector **w** evolucionó de forma diferente en cada filtro al tratar con las ondas básicas. Para el LMS estándar, en la mayoría de los casos, el mayor esfuerzo lo realizó al principio y luego iba haciendo ajustes menores; la excepción fue para los sinusoides de 250 Hz y de 1 kHz (Figura 19), en donde las magnitud **w** pareció converger a un valor constante. Para el LMS normalizado fue diferente ya que fue mucho más agresivo al principio, pero después logró reducir considerablemente dicha magnitud y mantenerla relativamente constante. El RLS a veces presentó un esfuerzo impulsional al principio y otras veces mantenía un esfuerzo agresivo todo el tiempo.

La magnitud del vector **w** también presentó una serie de cambios característicos con las señales moduladas y el barrido frecuencial. Para el LMS estándar, la evolución fue periódica para el caso AM (Figura 24) y en los otros de modulación parecía no converger a un valor (Figuras 25 y 26). Para el LMS normalizado se mantuvo la agresividad inicial, aunque hubo excepciones (Figuras 60 y 62) en donde se presentaron una serie de aumentos periódicos del vector de pesos. Para el RLS, se presentó la misma serie de impulsos para los mismos casos; para los otros se mostró un único impulso al principio.

9.4.3. Filtrado de los *clips* musicales

El LMS estándar se comportó de una forma característica al momento de filtrar los diferentes *clips* musicales. Lo que más resalta al observar las señales $y[n]$ obtenidas, es que la amplitud de las mismas no es comparable a la de las señales deseadas. Esto concuerda con el hecho de que esta variación de LMS no consigue el ajuste apropiado si hay una gran discrepancia en amplitud entre las señales $x[n]$ y $d[n]$, si no que mantiene una tasa de aprendizaje constante β. Adicionalmente se observó cómo el filtro elimina bandas enteras de frecuencia al generar un gran esfuerzo de atenuación (Figura 31). Por último se nota el promedio de las máximas magnitudes $||\mathbf{w}||$ (Cuadro 6), el cual indica que fue el menos agresivo de los tres filtros.

El LMS normalizado obtuvo un filtrado diferente al de la versión estándar. Al observar las señales generadas $y[n]$ contra las deseadas $d[n]$, estas sí son comparables en amplitud. Esta variedad, a diferencia de la estándar, hace un mejor ajuste al escalar la amplitud de la señal perturbada $x[n]$. Para este filtro resalta la forma del ECM para unas piezas de un solo instrumento (Figura 47), en donde se exhibe una forma de decaimiento exponencial. En cuanto a qué tanto esfuerzo aplicó sobre los coeficientes, este estuvo en el rango medio de los tres filtros (Cuadro 8).

El RLS fue el que se desempeñó mejor de las tres variedades al momento de filtrar los *clips* musicales. Este desempeño se nota en el promedio del error, el cual fue el más pequeño de los tres. Sin embargo, el costo de este alto rendimiento fueron los cambios bruscos y repentinos en la magnitud del vector de pesos **w**. Esta dinámica se manifestó a lo largo de toda la duración de los *clips*. En efecto, el promedio de los máximos valores de $||\mathbf{w}||$ fue el mayor de todos (Cuadro 10).

Los resultados escuchados en audio de los *clips musicales* fueron únicos para cada tipo de filtrado. El LMS estándar presentó la baja ganancia mostrada en las gráficas de las señales en dominio del tiempo. Para el LMS normalizado fue audible el emparejamiento en amplitud de las señales $y[n]$ y $d[n]$, pero además amplificó ruidos indeseados que no corresponden a la señal original. Para el último filtro, el RLS, se obtuvo una buena fidelidad en audio, la cual era de esperarse por el bajo ECM obtenido.

9.4.4. Robustez del RLS y el LMS

El hecho de que el RLS haya haya obtenido un ECM menor que el LMS, y más rápido (en algunos casos), se debió a la información que este algoritmo aprovecha en cada iteración.

Primero, el RLS usa los estimados de la matriz de correlación $\mathbf{R}(n)$ de la señal de entrada y vector de correlación cruzada $\mathbf{p}(n)$, lo cual se contrasta con el LMS, que únicamente emplea el error como medida de ajuste. Adicionalmente, el filtrado por RLS se vale de un factor de memoria λ, mientras que el LMS solo actúa en función del error presente. Y cómo último detalle, cabe mencionar que si se tuvieran las matrices $\mathbf{R}(n)$ y $\mathbf{p}(n)$ reales, el algoritmo RLS convergiría en una sola iteración. La forma en que opera el RLS lo hace más robusto ante señales no estacionarias (señales que cambian su contenido espectral con el tiempo), a diferencia del LMS.

9.4.5. Orden del filtro y complejidad computacional

En el uso de cada uno de los filtros se sopesó tanto el orden y la complejidad computacional que implica cada uno. A pesar de que se emplea el lema de inversión de matriz para reducir la complejidad del RLS a un mismo orden N, este sigue siendo más caro computacionalmente de implementar que el LMS. Tomando esta consideración, con la señales determinísticas, estaría justificado emplear LMS. Sin embargo, ya en la implementación de los clips el orden del RLS fue mucho menor al del LMS escogido.

9.4.6. Significado de la variación en w

Se hace notar la diferencia en la magnitud del vector $\mathbf{w}$, la cual se estableció en diferentes valores para las diferentes señales aplicadas a un mismo filtro. Es suficiente saber qué la norma de $\mathbf{w}$ varía para saber que los pesos individuales tampoco son los mismos. Se nota que para algunas pruebas con señales determinísticas pareciera haber convergencia a valores muy cercanos en magnitud (para las ondas básicas, con el filtro LMS normalizado, por ejemplo). A pesar de esto, dichas magnitudes siguen siendo diferentes a las encontradas en el filtrado de *clips* musicales.

Los diferentes valores de pesos $\mathbf{w}$ obtenidos hablan sobre el tipo de sistema con el que se está tratando. Como no se convergió en los pesos $\mathbf{w}$ para todos las señales de prueba en ninguno de los filtros, no se identificó el sistema general del cuarto. Por lo tanto el sistema no es LTI (lineal e invariante en el tiempo). Sin embargo, al emplear un filtro FIR y al haber obtenido que en los coeficientes $\mathbf{w}$ van cambiando conforme el tiempo, se habla del sistema como LTV (lineal, variante en el tiempo). Un sistema LTV, entonces, se puede interpretar como la linealización sobre una trayectoria de un sistema no lineal.

En base a lo encontrado se saben los cambios que se deben realizar al modelo. En vez de un filtro FIR que es LTI, se debe emplear un modelo que capture las no linealidades. Una red neuronal podría ser capaz de cumplir con susodicha tarea. Obteniendo el modelo del inverso del cuarto entonces sería posible generalizar y prescindir de una señal deseada $d[n]$ para eliminar la reverberación y demás ruido disruptivo.

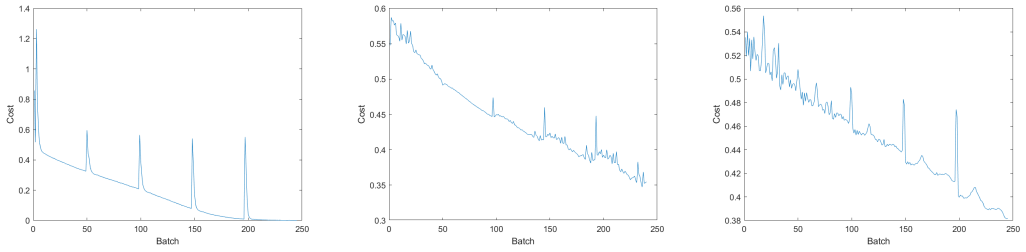
CAPÍTULO 10

Redes neuronales

10.1. TDNN

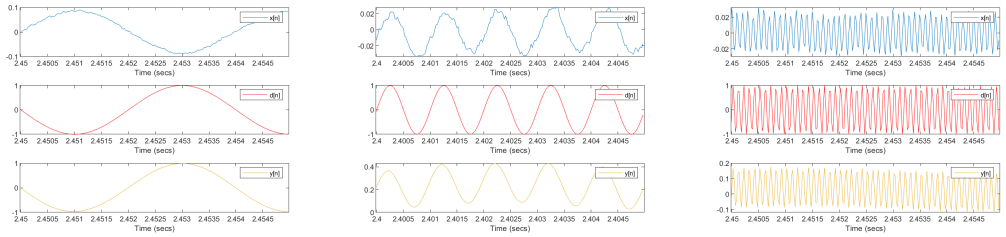
10.1.1. Red neuronal por pista

Activación sigmoide



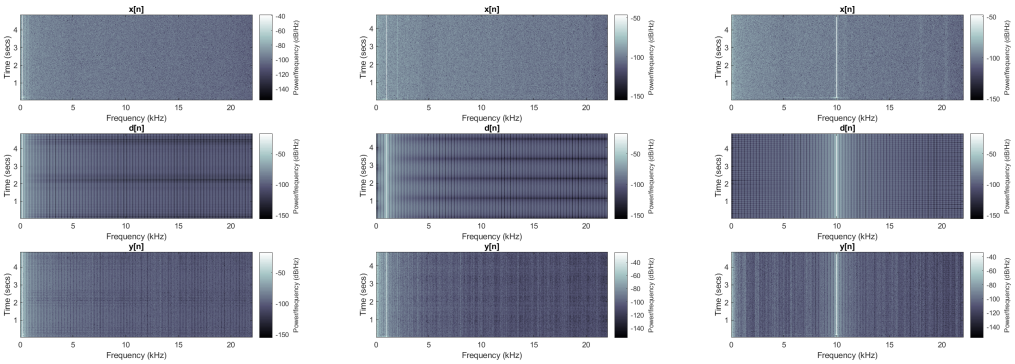
(a) 250 Hz (b) 1 kHz (c) 10 kHz

Figura 72: Costo por batch del entrenamiento de redes TDNN para sinusoides.



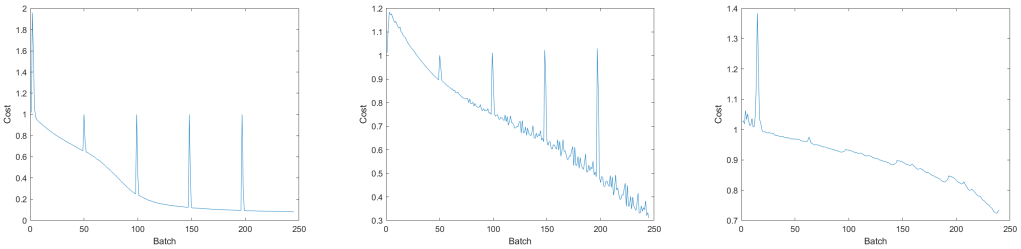
(a) 250 Hz (b) 1 kHz (c) 10 kHz

Figura 73: Señales generadas por las redes TDNN para sinusoides.



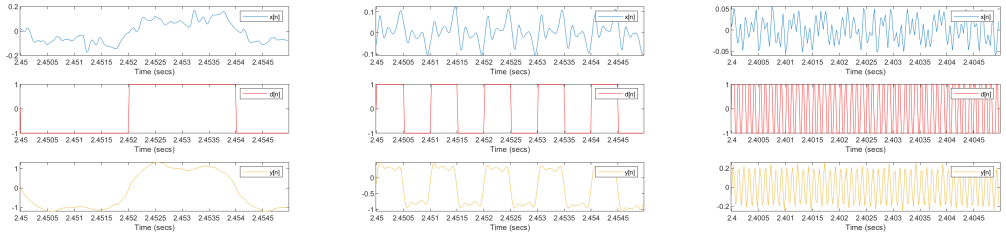
(a) 250 Hz (b) 1 kHz (c) 10 kHz

Figura 74: Espectrogramas generados por las redes TDNN para sinusoides.



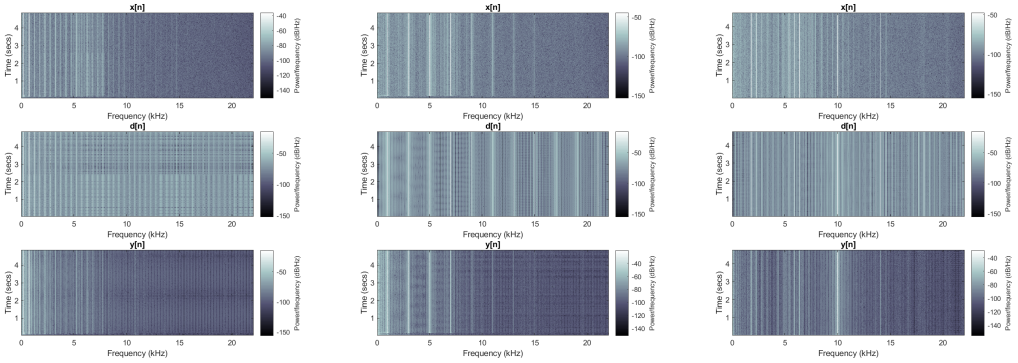
(a) 250 Hz (b) 1 kHz (c) 10 kHz

Figura 75: Costo por batch del entrenamiento de redes TDNN para ondas cuadradas.



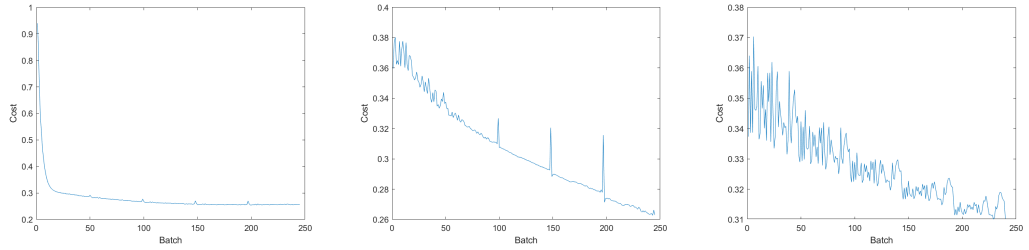
(a) 250 Hz (b) 1 kHz (c) 10 kHz

Figura 76: Señales generadas por las redes TDNN para ondas cuadradas.



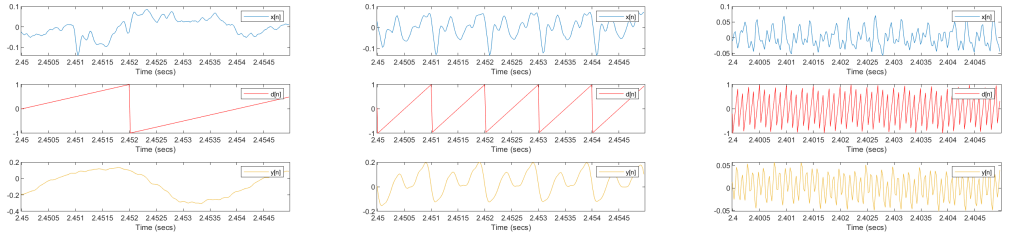
(a) 250 Hz (b) 1 kHz (c) 10 kHz

Figura 77: Espectrogramas generados por las redes TDNN para ondas cuadradas.



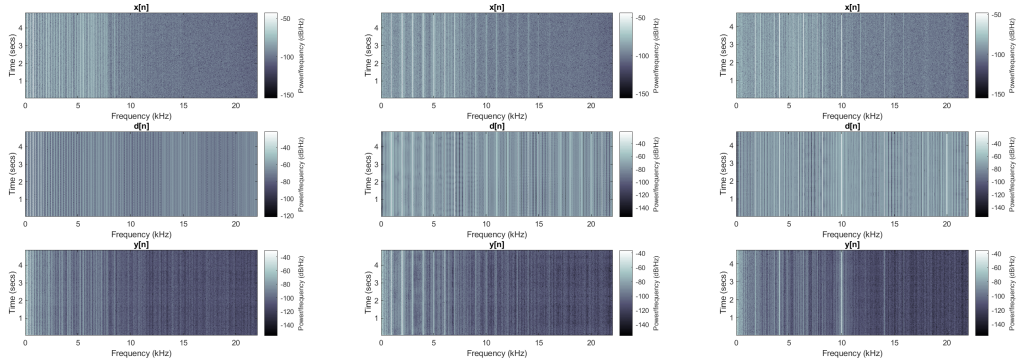
(a) 250 Hz (b) 1 kHz (c) 10 kHz

Figura 78: Costo por batch del entrenamiento de redes TDNN para ondas diente de sierra.



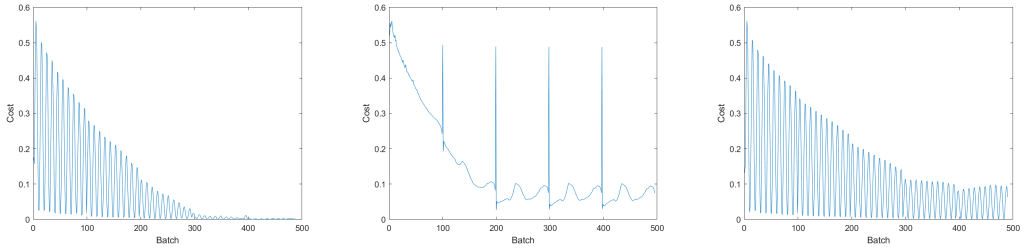
(a) 250 Hz (b) 1 kHz (c) 10 kHz

Figura 79: Señales generadas por las redes TDNN para ondas diente de sierra.



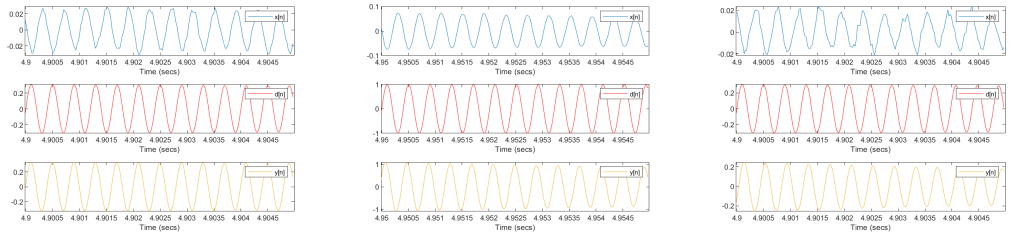
(a) 250 Hz (b) 1 kHz (c) 10 kHz

Figura 80: Espectrogramas generados por las redes TDNN para ondas diente de sierra.



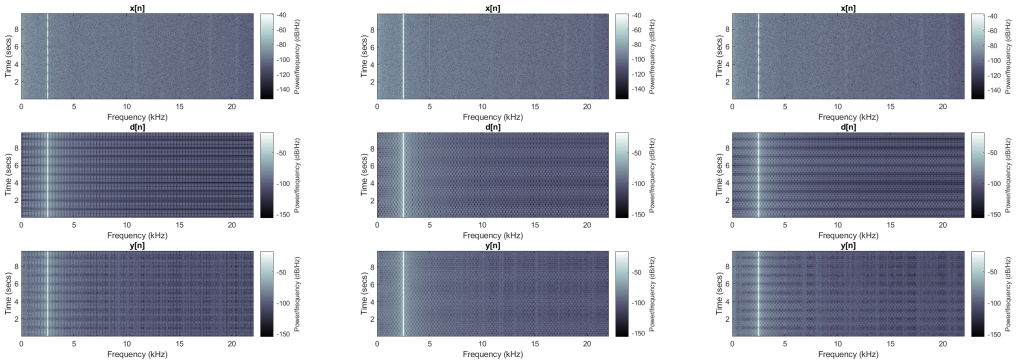
(a) AM (b) FM (c) AM y FM

Figura 81: Costo por batch del entrenamiento de redes TDNN para las señales AM, FM y AM-FM.



(a) AM (b) FM (c) AM y FM

Figura 82: Salidas generadas por las redes TDNN para las señales AM, FM y AM-FM.



(a) AM (b) FM (c) AM y FM

Figura 83: Espectrogramas generados por las redes TDNN para las señales AM, FM y AM-FM.

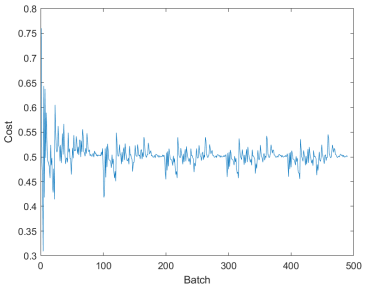


Figura 84: Costo por batch del entrenamiento de redes TDNN para la señal chirp.

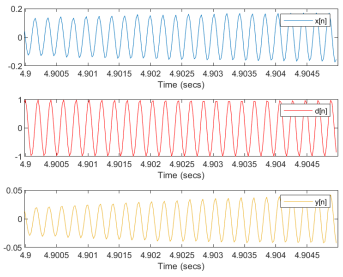


Figura 85: Salidas generadas por las redes TDNN para la señal chirp.

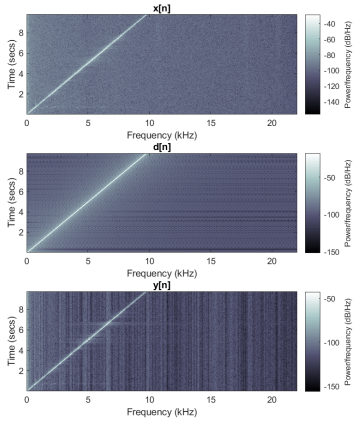
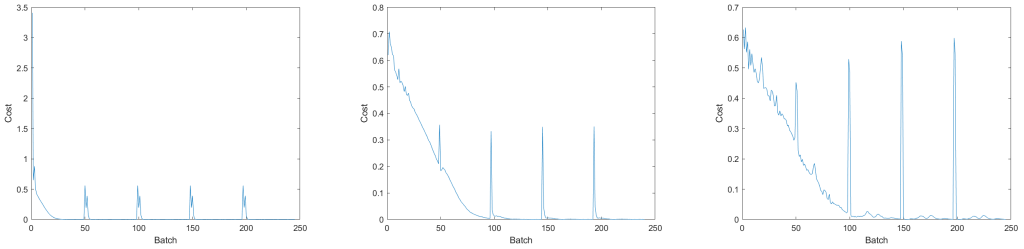


Figura 86: Espectrogramas generados por las redes TDNN para la señal chirp.

Señal	Error promedio
Onda sinusoidal 250 Hz	0.050242
Onda sinusoidal 1 kHz	0.53469
Onda sinusoidal 10 kHz	0.55539
Onda cuadrada 250 Hz	0.21318
Onda cuadrada 1 kHz	0.50362
Onda cuadrada 10 kHz	0.85229
Onda diente de sierra 250 Hz	0.40827
Onda diente de sierra 1 kHz	0.45698
Onda diente de sierra 10 kHz	0.47875
Onda AM	0.03236
Onda FM	0.22537
Onda AM y FM	0.16818
Barrido sinusoidal	0.63501
Promedio	0.3934

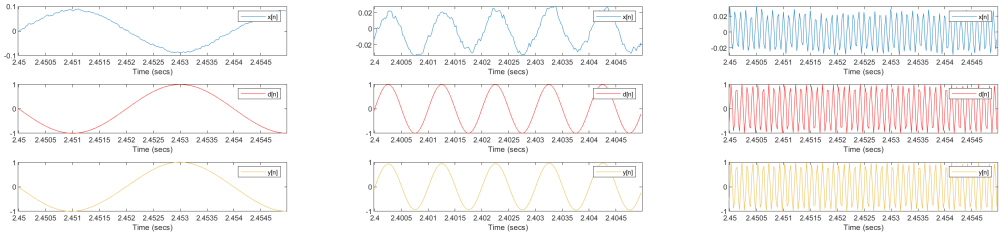
Cuadro 11: Errores promedios de la señales determinísticas por las redes TDNN con activación sigmoide.

Activación tanh



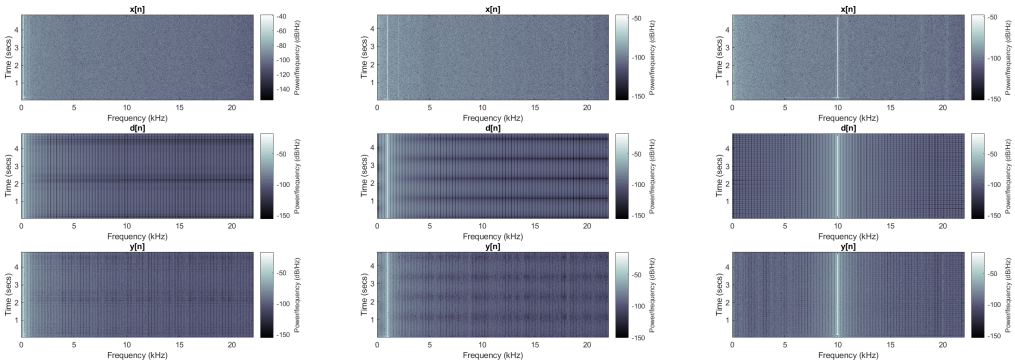
(a) 250 Hz (b) 1 kHz (c) 10 kHz

Figura 87: Costo por batch del entrenamiento de redes TDNN para sinusoides.



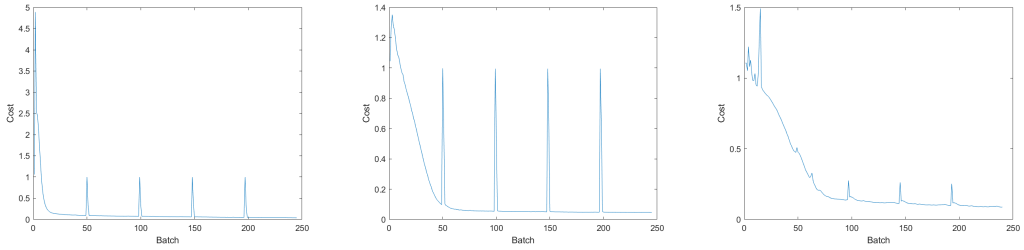
(a) 250 Hz (b) 1 kHz (c) 10 kHz

Figura 88: Señales generadas por las redes TDNN para sinusoides.



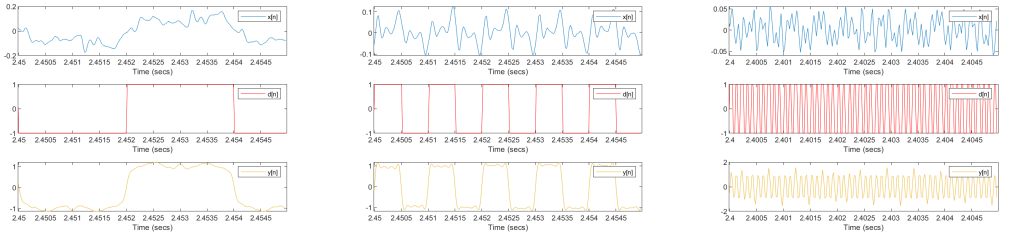
(a) 250 Hz (b) 1 kHz (c) 10 kHz

Figura 89: Espectrogramas generados por las redes TDNN para sinusoides.



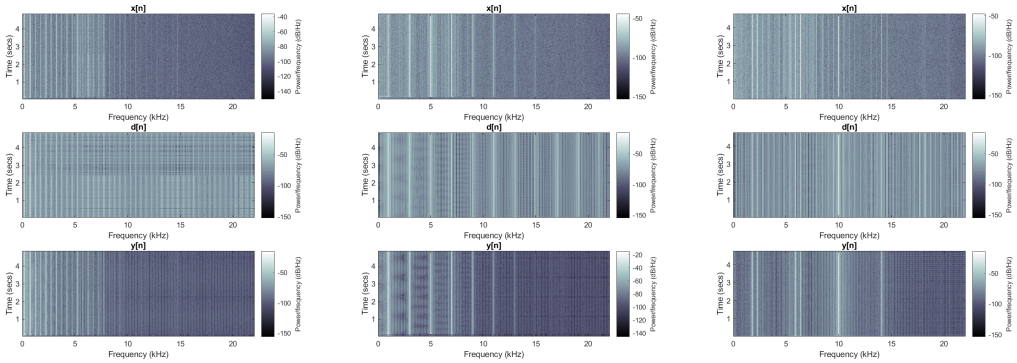
(a) 250 Hz (b) 1 kHz (c) 10 kHz

Figura 90: Costo por batch del entrenamiento de redes TDNN para ondas cuadradas.



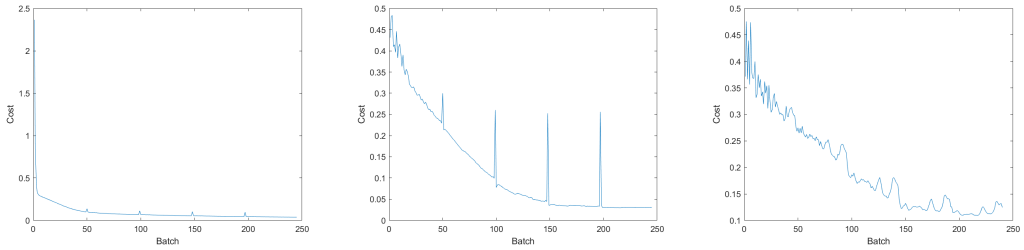
(a) 250 Hz (b) 1 kHz (c) 10 kHz

Figura 91: Señales generadas por las redes TDNN para ondas cuadradas.



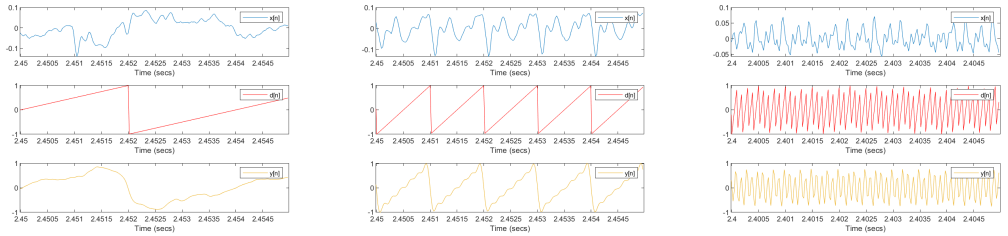
(a) 250 Hz (b) 1 kHz (c) 10 kHz

Figura 92: Espectrogramas generados por las redes TDNN para ondas cuadradas.



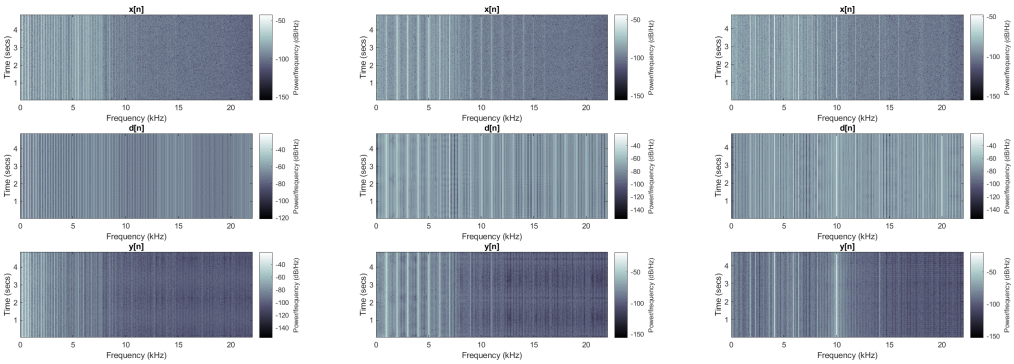
(a) 250 Hz (b) 1 kHz (c) 10 kHz

Figura 93: Costo por batch del entrenamiento de redes TDNN para ondas diente de sierra.



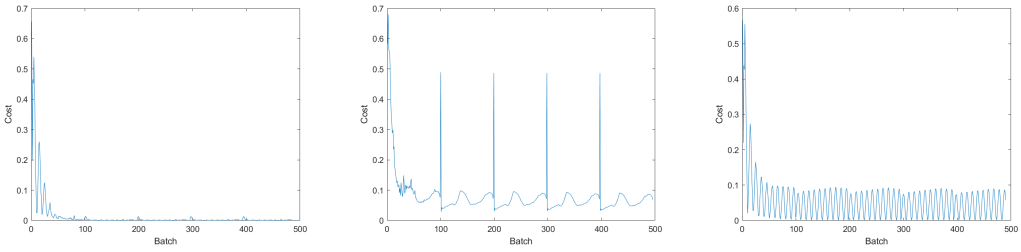
(a) 250 Hz (b) 1 kHz (c) 10 kHz

Figura 94: Señales generadas por las redes TDNN para ondas diente de sierra.



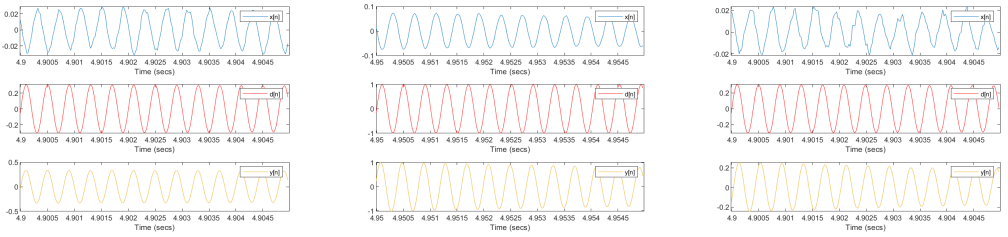
(a) 250 Hz (b) 1 kHz (c) 10 kHz

Figura 95: Espectrogramas generados por las redes TDNN para ondas diente de sierra.



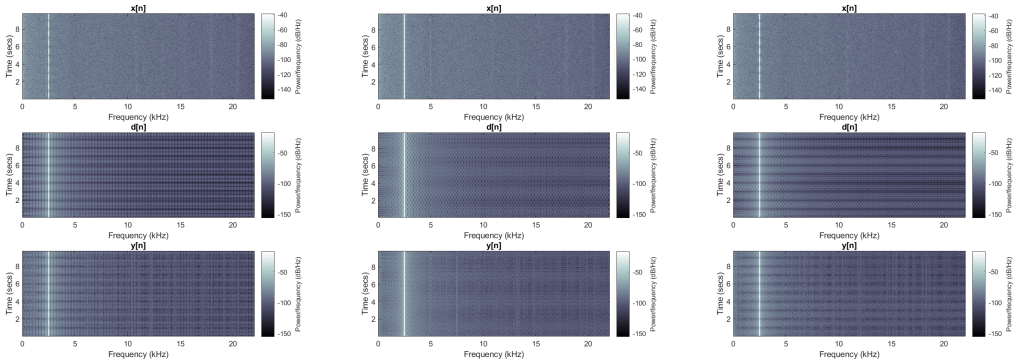
(a) AM (b) FM (c) AM y FM

Figura 96: Costo por batch del entrenamiento de redes TDNN para las señales AM, FM y AM-FM.



(a) AM (b) FM (c) AM y FM

Figura 97: Salidas generadas por las redes TDNN para las señales AM, FM y AM-FM.



(a) AM (b) FM (c) AM y FM

Figura 98: Espectrogramas generados por las redes TDNN para las señales AM, FM y AM-FM.

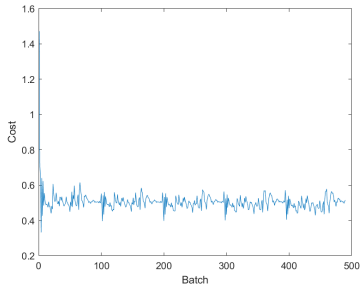


Figura 99: Costo por batch del entrenamiento de redes TDNN para la señal chirp.

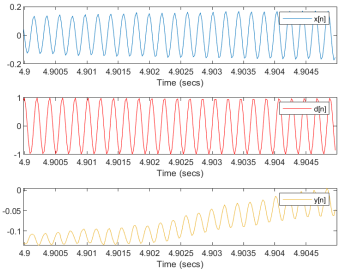


Figura 100: Salidas generadas por las redes TDNN para la señal chirp.

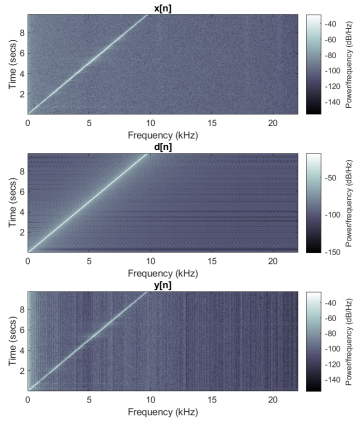
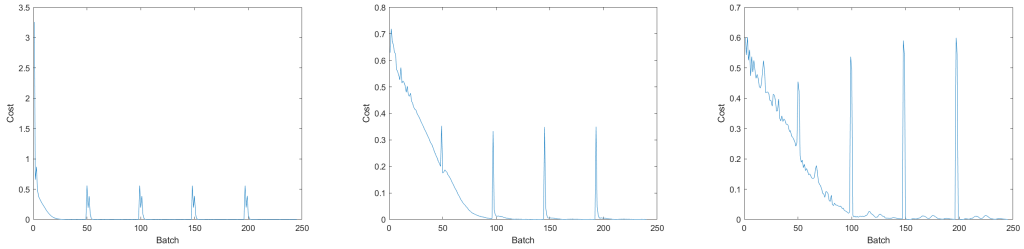


Figura 101: Espectrogramas generados por las redes TDNN para la señal chirp.

Señal	Error promedio
Onda sinusoidal 250 Hz	0.042115
Onda sinusoidal 1 kHz	0.047062
Onda sinusoidal 10 kHz	0.076345
Onda cuadrada 250 Hz	0.1515
Onda cuadrada 1 kHz	0.14486
Onda cuadrada 10 kHz	0.23831
Onda diente de sierra 250 Hz	0.1254
Onda diente de sierra 1 kHz	0.095729
Onda diente de sierra 10 kHz	0.2741
Onda AM	0.032782
Onda FM	0.21547
Onda AM y FM	0.16412
Barrido sinusoidal	0.63115
Promedio	0.1722

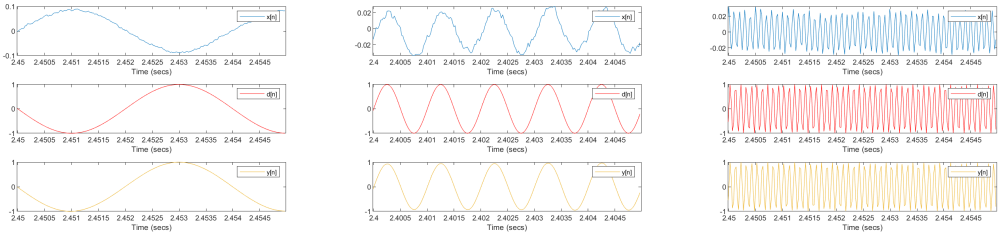
Cuadro 12: Errores promedios de la señales determinísticas por las redes TDNN con activación tanh.

Activación sinusoidal



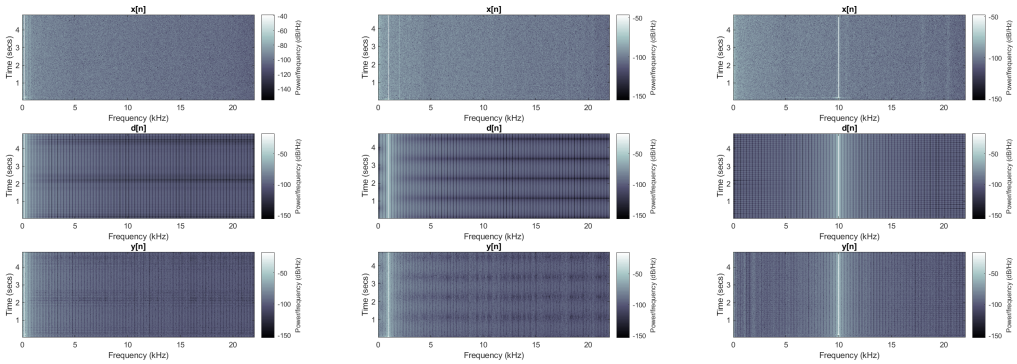
(a) 250 Hz (b) 1 kHz (c) 10 kHz

Figura 102: Costo por batch del entrenamiento de redes TDNN para sinusoides.



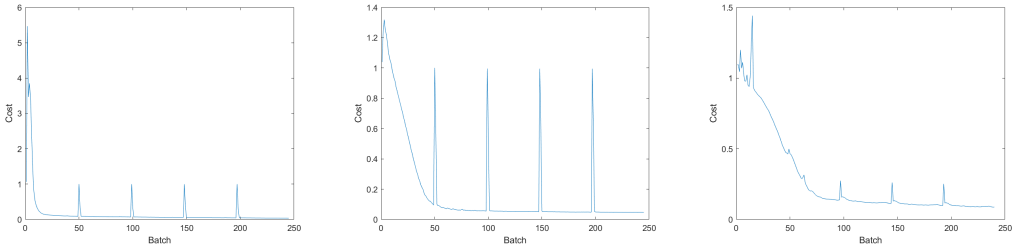
(a) 250 Hz (b) 1 kHz (c) 10 kHz

Figura 103: Señales generadas por las redes TDNN para sinusoides.



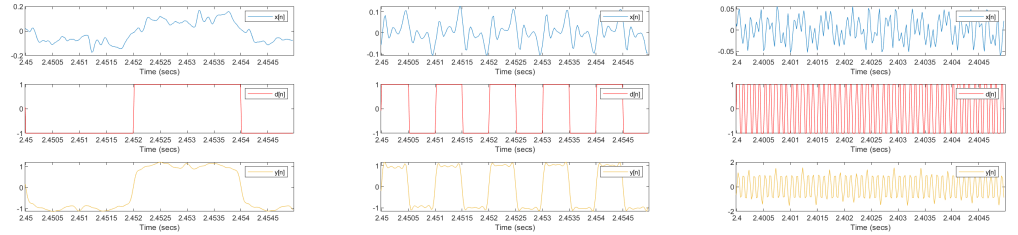
(a) 250 Hz (b) 1 kHz (c) 10 kHz

Figura 104: Espectrogramas generados por las redes TDNN para sinusoides.



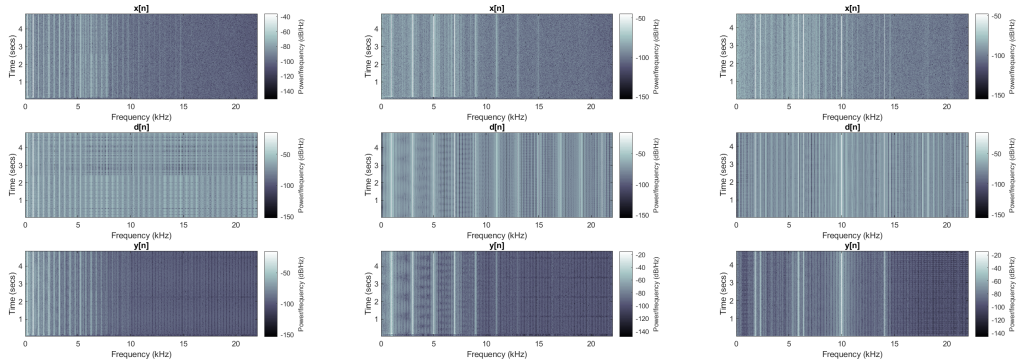
(a) 250 Hz (b) 1 kHz (c) 10 kHz

Figura 105: Costo por batch del entrenamiento de redes TDNN para ondas cuadradas.



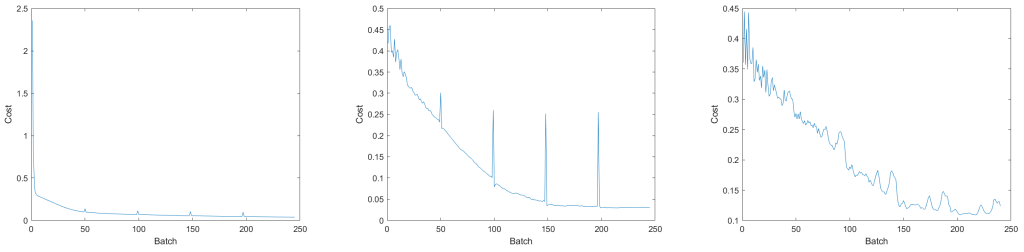
(a) 250 Hz (b) 1 kHz (c) 10 kHz

Figura 106: Señales generadas por las redes TDNN para ondas cuadradas.



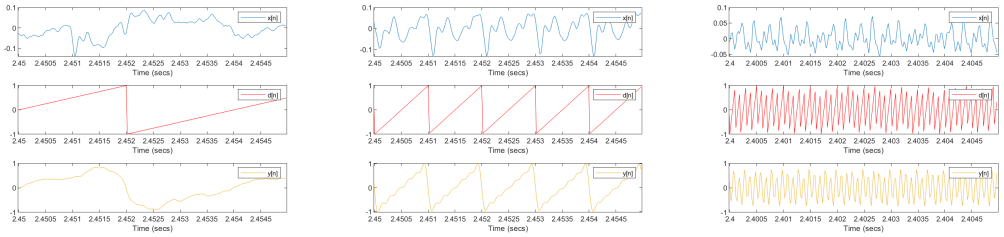
(a) 250 Hz (b) 1 kHz (c) 10 kHz

Figura 107: Espectrogramas generados por las redes TDNN para ondas cuadradas.



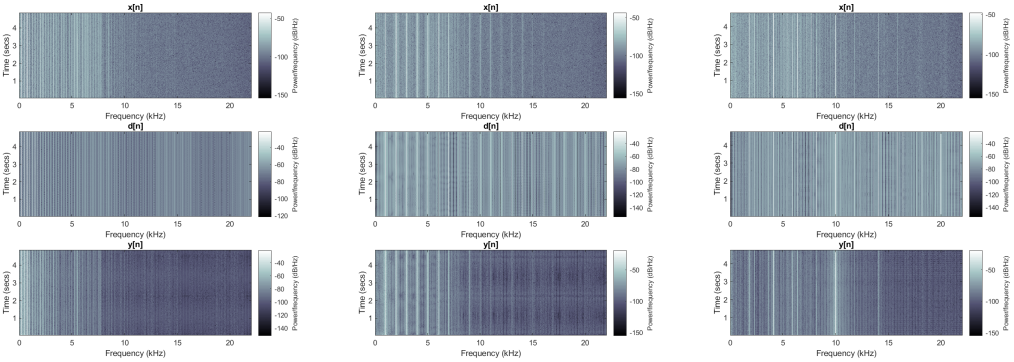
(a) 250 Hz (b) 1 kHz (c) 10 kHz

Figura 108: Costo por batch del entrenamiento de redes TDNN para ondas diente de sierra.



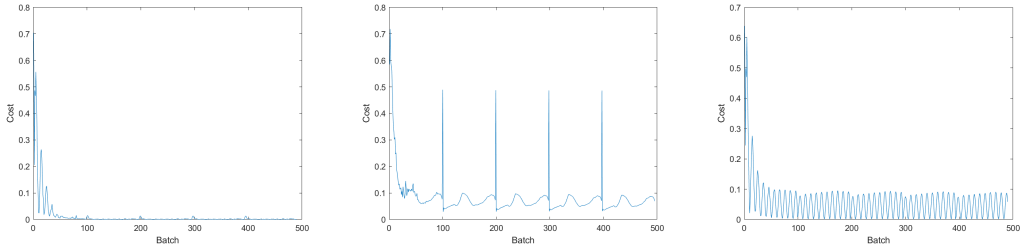
(a) 250 Hz (b) 1 kHz (c) 10 kHz

Figura 109: Señales generadas por las redes TDNN para ondas diente de sierra.



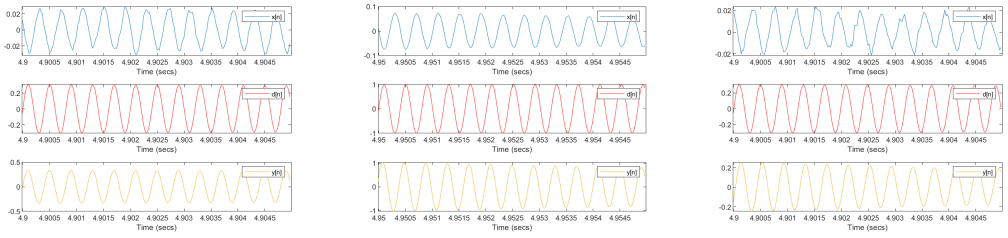
(a) 250 Hz (b) 1 kHz (c) 10 kHz

Figura 110: Espectrogramas generados por las redes TDNN para ondas diente de sierra.



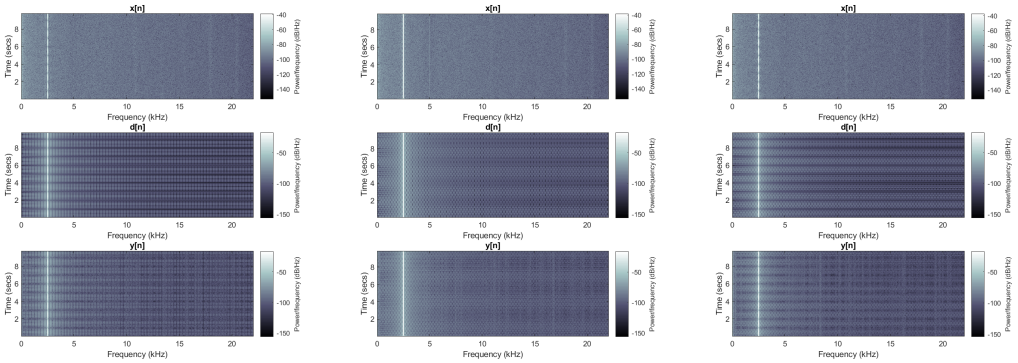
(a) AM (b) FM (c) AM y FM

Figura 111: Costo por batch del entrenamiento de redes TDNN para las señales AM, FM y AM-FM.



(a) AM (b) FM (c) AM y FM

Figura 112: Salidas generadas por las redes TDNN para las señales AM, FM y AM-FM.



(a) AM (b) FM (c) AM y FM

Figura 113: Espectrogramas generados por las redes TDNN para las señales AM, FM y AM-FM.

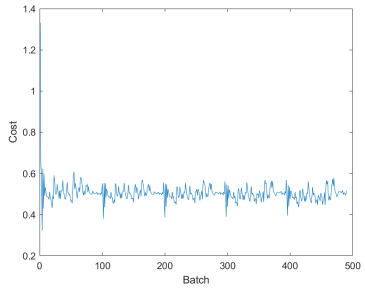


Figura 114: Costo por batch del entrenamiento de redes TDNN para la señal chirp.

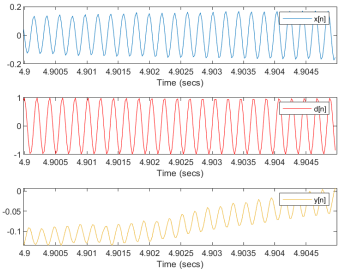


Figura 115: Salidas generadas por las redes TDNN para la señal chirp.

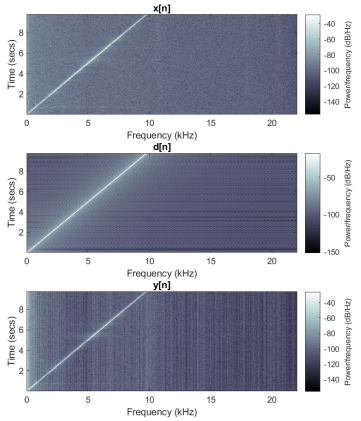


Figura 116: Espectrogramas generados por las redes TDNN para la señal chirp.

Señal	Error promedio
Onda sinusoidal 250 Hz	0.040983
Onda sinusoidal 1 kHz	0.046974
Onda sinusoidal 10 kHz	0.076403
Onda cuadrada 250 Hz	0.14911
Onda cuadrada 1 kHz	0.14582
Onda cuadrada 10 kHz	0.23843
Onda diente de sierra 250 Hz	0.12636
Onda diente de sierra 1 kHz	0.094523
Onda diente de sierra 10 kHz	0.27255
Onda AM	0.032676
Onda FM	0.2165
Onda AM y FM	0.16447
Barrido sinusoidal	0.63153
Promedio	0.1720

Cuadro 13: Errores promedios de la señales determinísticas por las redes TDNN con activación sinusoidal.

10.1.2. Red neuronal por lista de reproducción

Costo

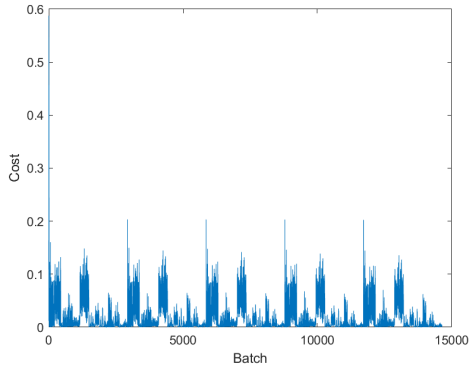


Figura 117: Evolución del costo por batch para la red TDNN con activación sigmoide.

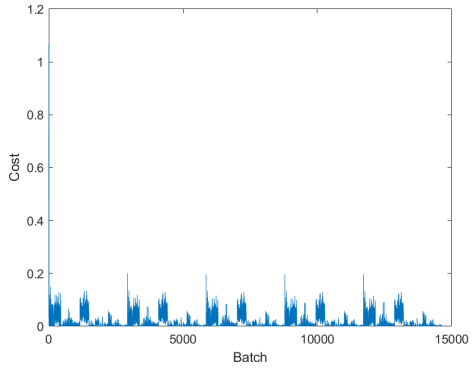


Figura 118: Evolución del costo por batch para la red TDNN con activación tanh.

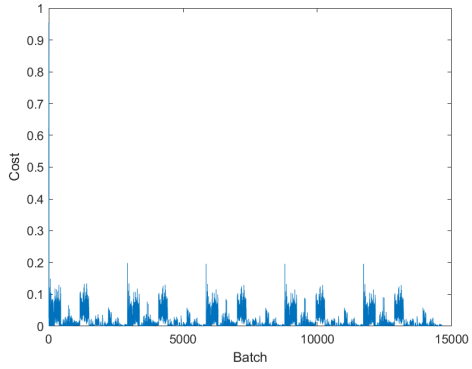
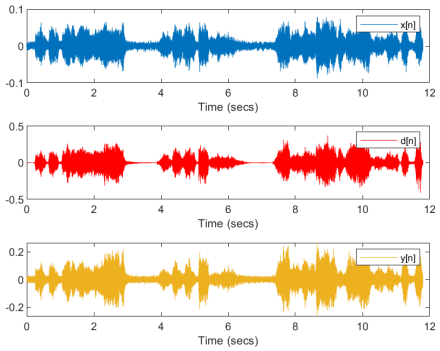
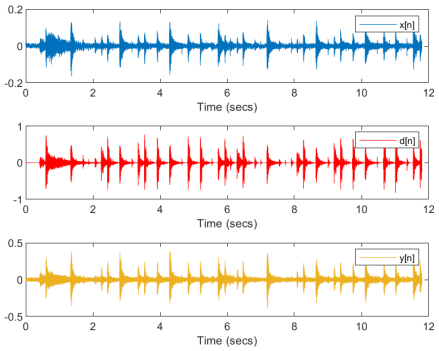


Figura 119: Evolución del costo por batch para la red TDNN con activación sinusoidal.

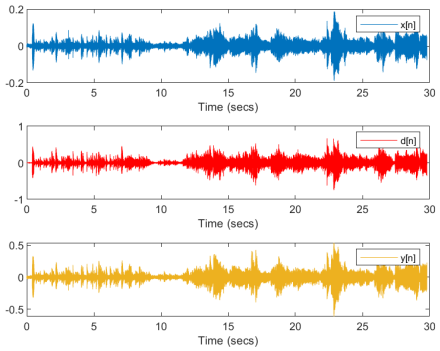
Filtrado



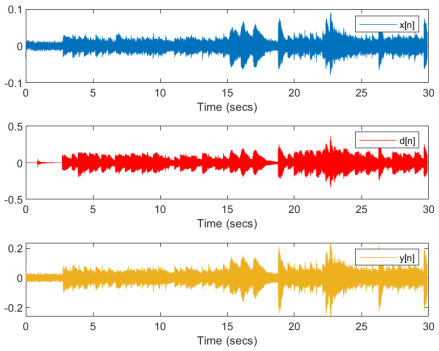
(a) *Bohemian Rhapsody*



(b) *Cadence*

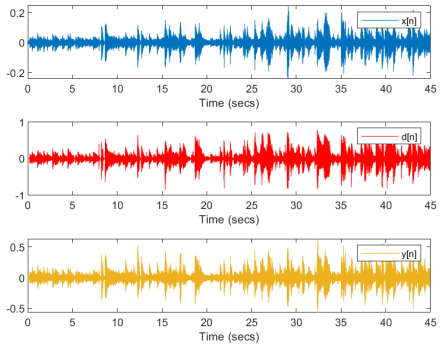


(c) *Peru*

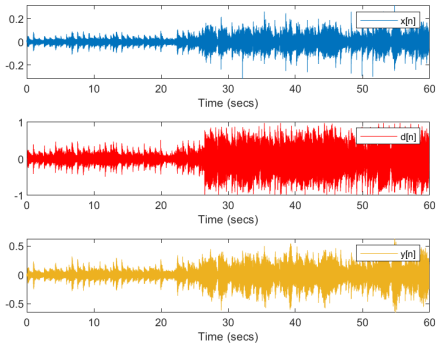


(d) *Week No. 8*

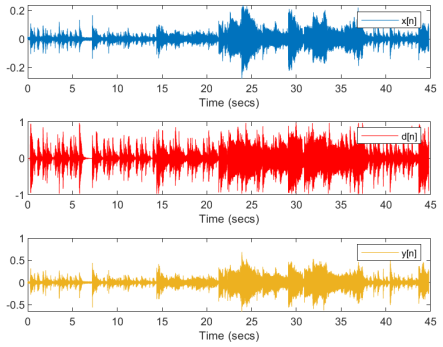
Figura 120: Señales del filtrado de clips de un solo instrumento por la TDNN con activación sinusoidal.



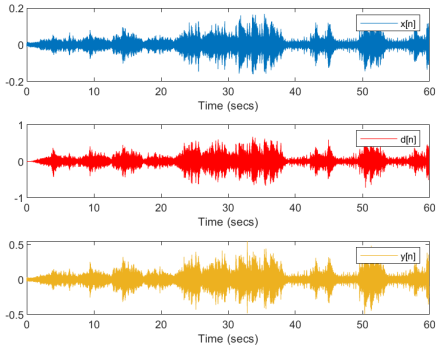
(a) *Lonely Cat*



(b) *Karma Police*

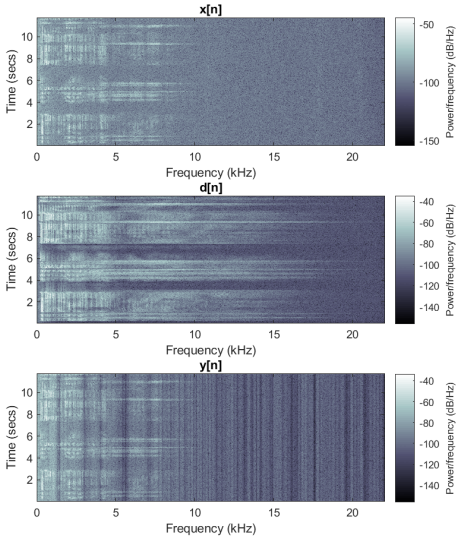


(c) *Atlantic Limited*

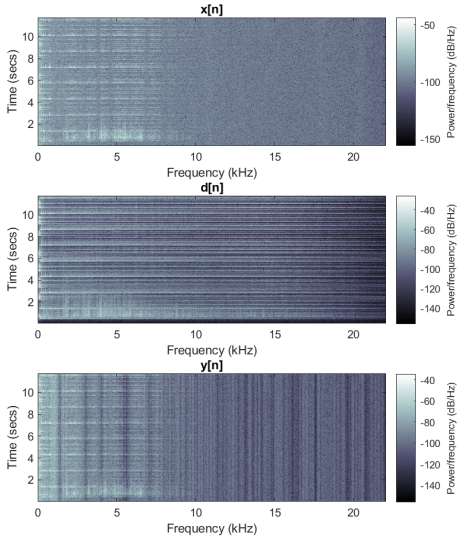


(d) *The Blue Danube*

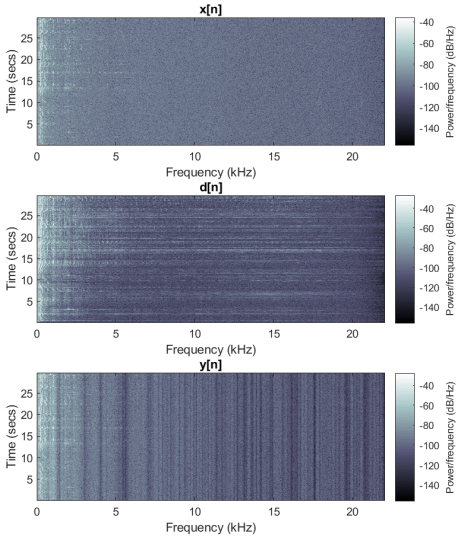
Figura 121: Señales del filtrado de clips de varios instrumentos por la TDNN con activación sinusoidal.



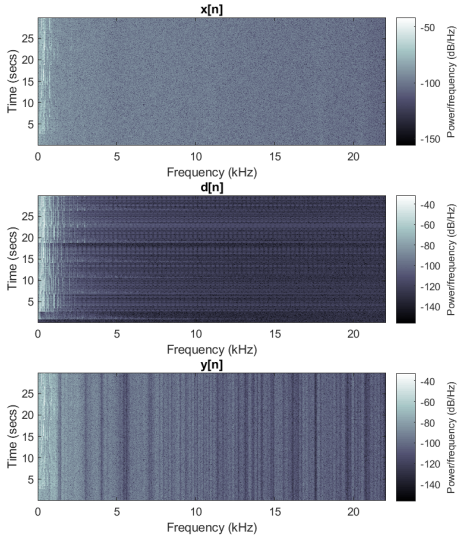
(a) *Bohemian Rhapsody*



(b) *Cadence*

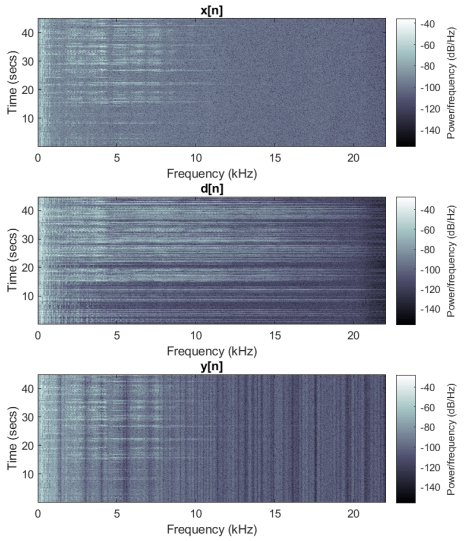


(c) *Peru*

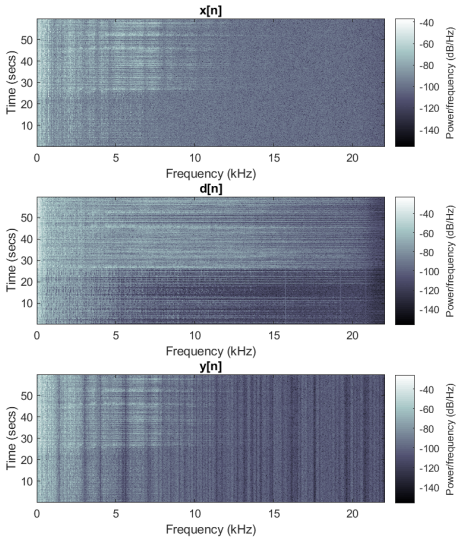


(d) *Week No. 8*

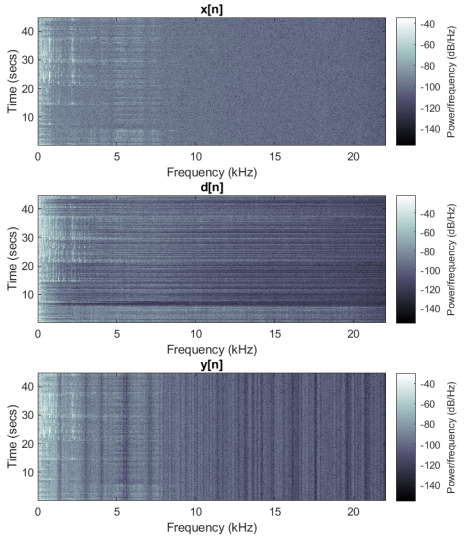
Figura 122: Espectrogramas del filtrado de clips de un solo instrumento por TDNN con activación sinusoidal.



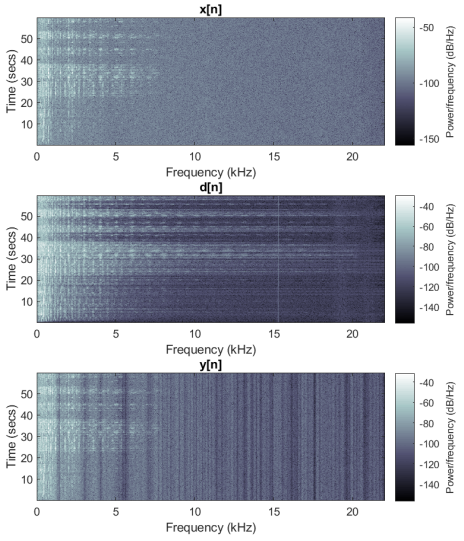
(a) *Lonely Cat*



(b) *Karma Police*



(c) *Atlantic Limited*



(d) *The Blue Danube*

Figura 123: Espectrogramas del filtrado de clips de varios instrumentos por TDNN con activación sinusoidal.

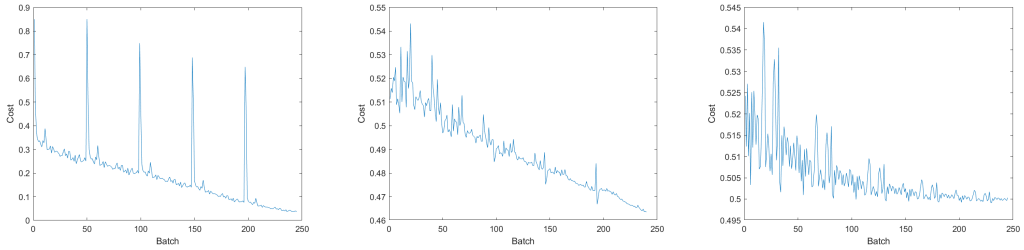
Señal	Error promedio
Bohemian Rhapsody	0.03266
Cadence	0.04599
Peru	0.044759
Week No. 8	0.031645
Lonely Cat	0.048548
Karma Police	0.1319
Atlantic Limited	0.11449
The Blue Danube	0.047487
Promedio	0.0622

Cuadro 14: Errores promedios de los clips musicales por las redes TDNN con activación sinusoidal.

10.2. *Focused Gamma*

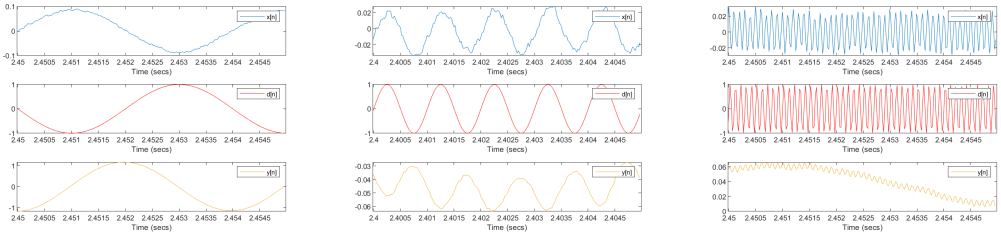
10.2.1. Red neuronal por pista

Activación sigmoide



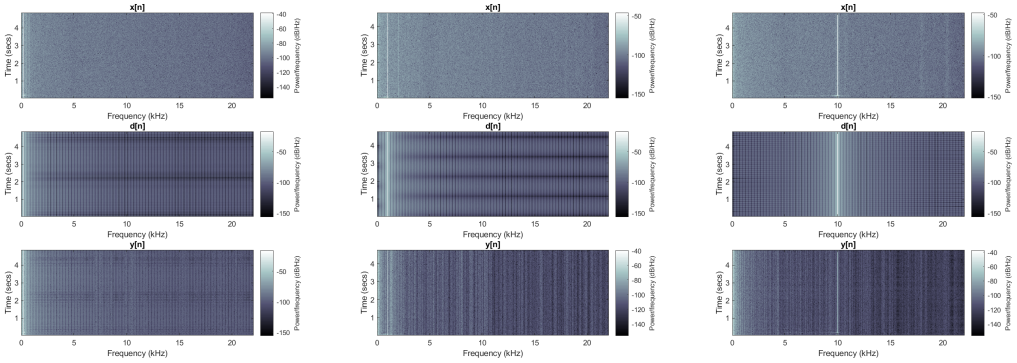
(a) 250 Hz (b) 1 kHz (c) 10 kHz

Figura 124: Costo por batch del entrenamiento de redes gamma para sinusoides.



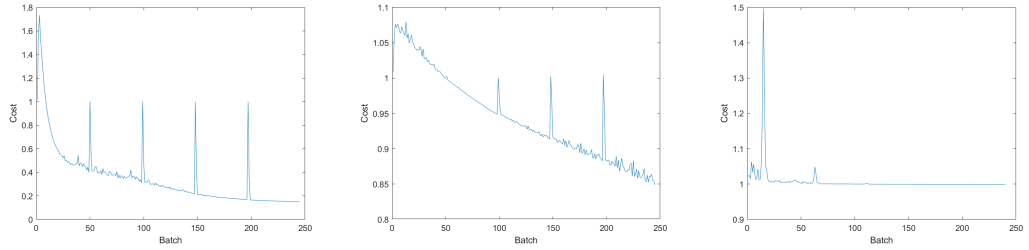
(a) 250 Hz (b) 1 kHz (c) 10 kHz

Figura 125: Señales generadas por las redes gamma para sinusoides.



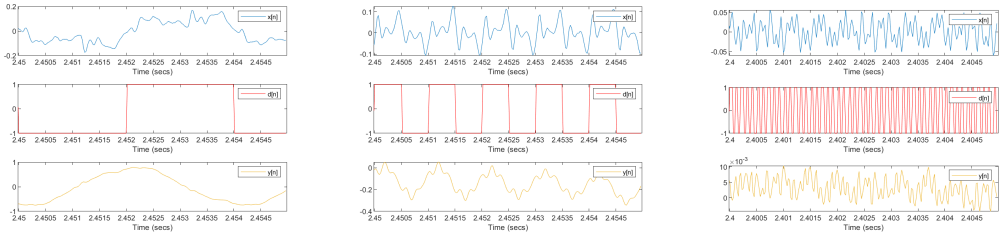
(a) 250 Hz (b) 1 kHz (c) 10 kHz

Figura 126: Espectrogramas generados por las redes gamma para sinusoides.



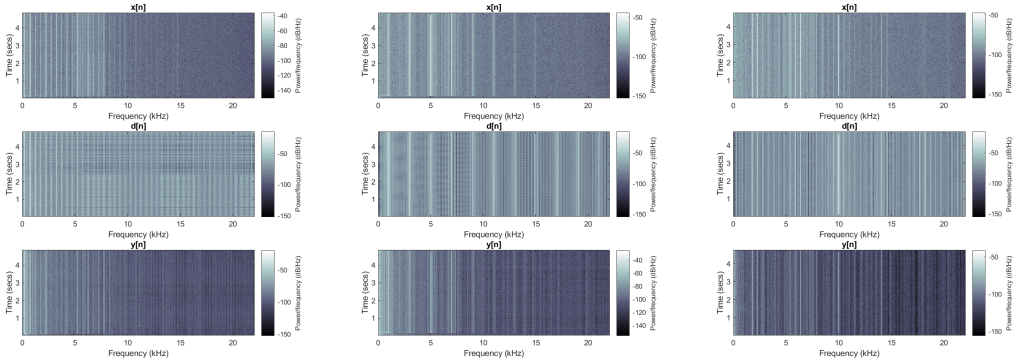
(a) 250 Hz (b) 1 kHz (c) 10 kHz

Figura 127: Costo por batch del entrenamiento de redes gamma para ondas cuadradas.



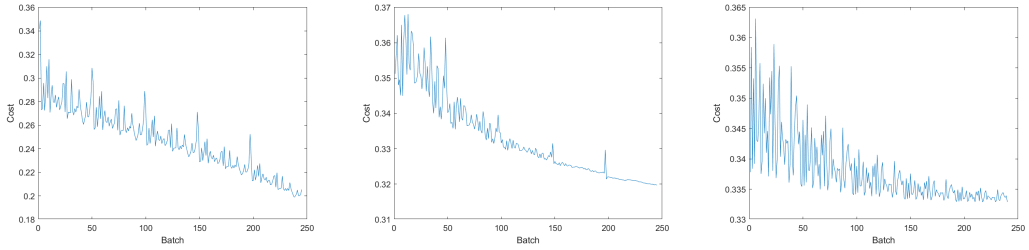
(a) 250 Hz (b) 1 kHz (c) 10 kHz

Figura 128: Señales generadas por las redes gamma para ondas cuadradas.



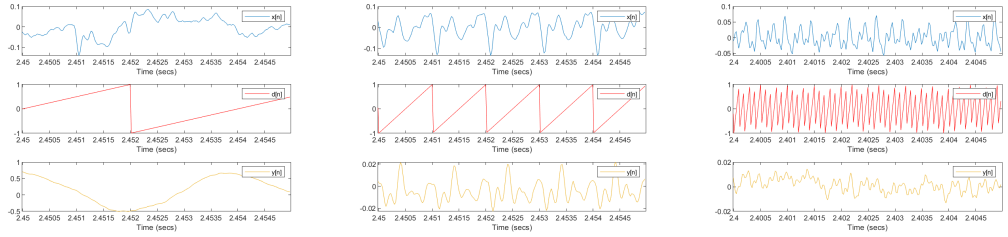
(a) 250 Hz (b) 1 kHz (c) 10 kHz

Figura 129: Espectrogramas generados por las redes gamma para ondas cuadradas.



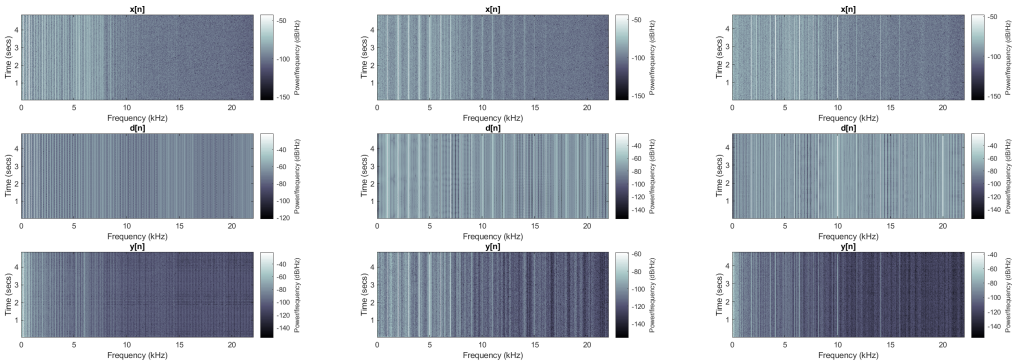
(a) 250 Hz (b) 1 kHz (c) 10 kHz

Figura 130: Costo por batch del entrenamiento de redes gamma para ondas diente de sierra.



(a) 250 Hz (b) 1 kHz (c) 10 kHz

Figura 131: Señales generadas por las redes gamma para ondas diente de sierra.



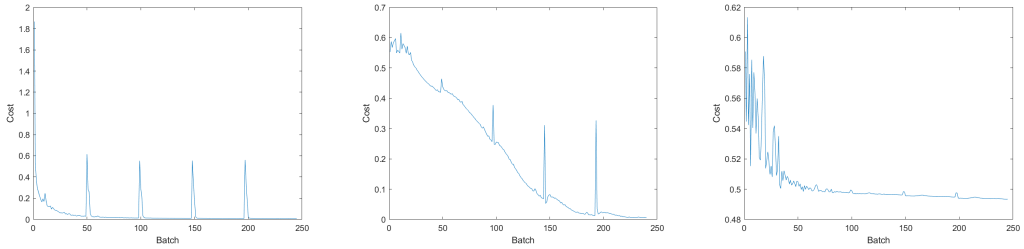
(a) 250 Hz (b) 1 kHz (c) 10 kHz

Figura 132: Espectrogramas generados por las redes gamma para ondas diente de sierra.

Señal	Error promedio
Onda sinusoidal 250 Hz	0.98246
Onda sinusoidal 1 kHz	0.64528
Onda sinusoidal 10 kHz	0.63547
Onda cuadrada 250 Hz	0.89362
Onda cuadrada 1 kHz	0.93749
Onda cuadrada 10 kHz	1.0001
Onda diente de sierra 250 Hz	0.62517
Onda diente de sierra 1 kHz	0.5011
Onda diente de sierra 10 kHz	0.4984
Onda AM	0.39987
Onda FM	0.63484
Onda AM y FM	0.41344
Barrido sinusoidal	0.63411
Promedio	0.6770

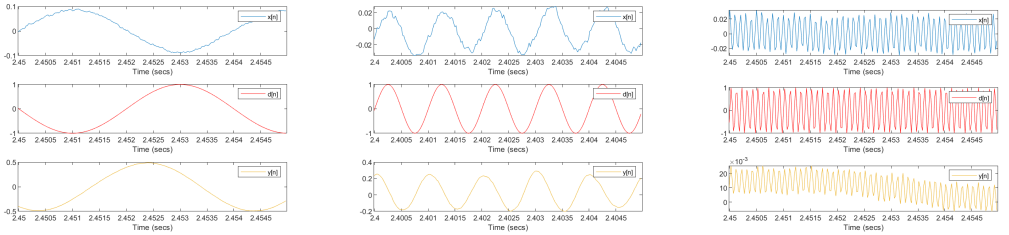
Cuadro 15: Errores promedios de la señales determinísticas por las redes gamma con activación sigmoide.

Activación tanh



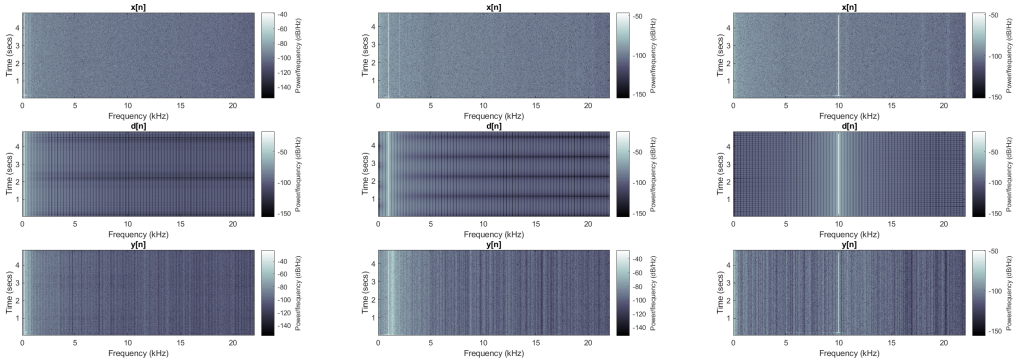
(a) 250 Hz (b) 1 kHz (c) 10 kHz

Figura 133: Costo por batch del entrenamiento de redes gamma para sinusoides.



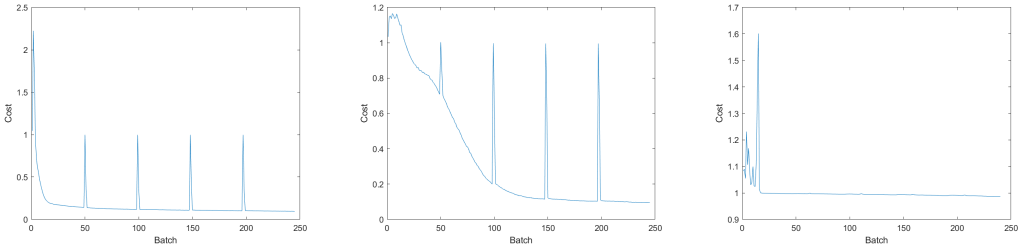
(a) 250 Hz (b) 1 kHz (c) 10 kHz

Figura 134: Señales generadas por las redes gamma para sinusoides.



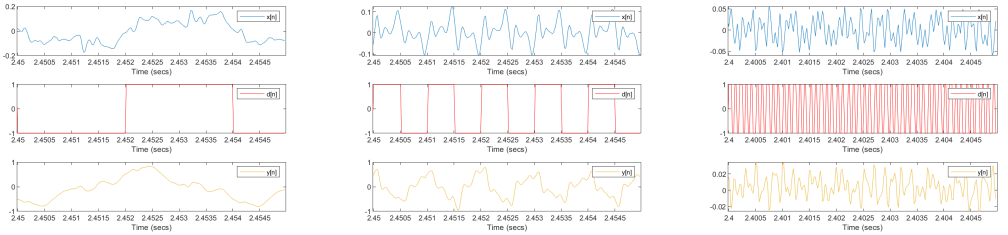
(a) 250 Hz (b) 1 kHz (c) 10 kHz

Figura 135: Espectrogramas generados por las redes gamma para sinusoides.



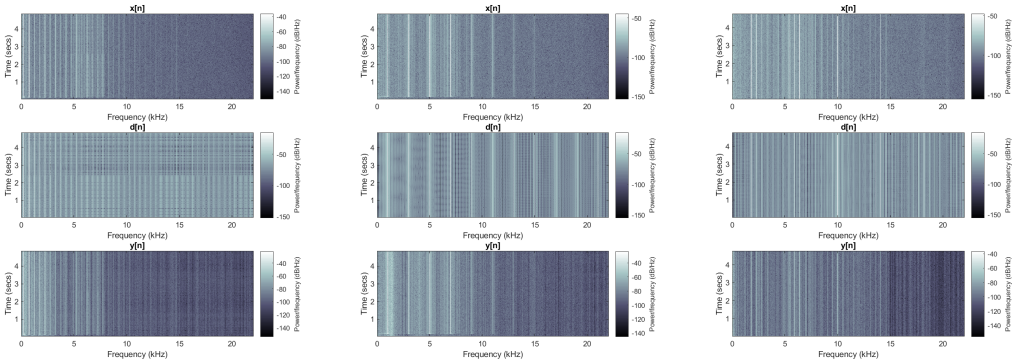
(a) 250 Hz (b) 1 kHz (c) 10 kHz

Figura 136: Costo por batch del entrenamiento de redes gamma para ondas cuadradas.



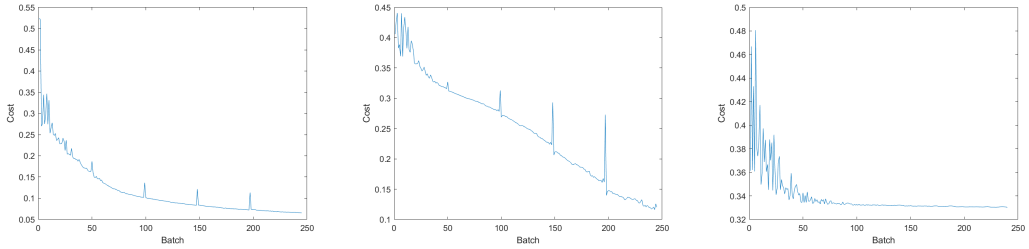
(a) 250 Hz (b) 1 kHz (c) 10 kHz

Figura 137: Señales generadas por las redes gamma para ondas cuadradas.



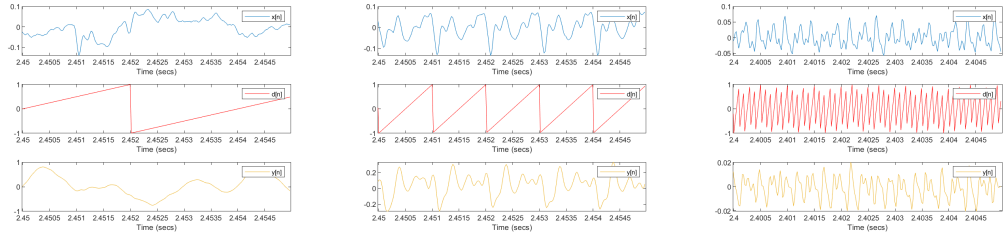
(a) 250 Hz (b) 1 kHz (c) 10 kHz

Figura 138: Espectrogramas generados por las redes gamma para ondas cuadradas.



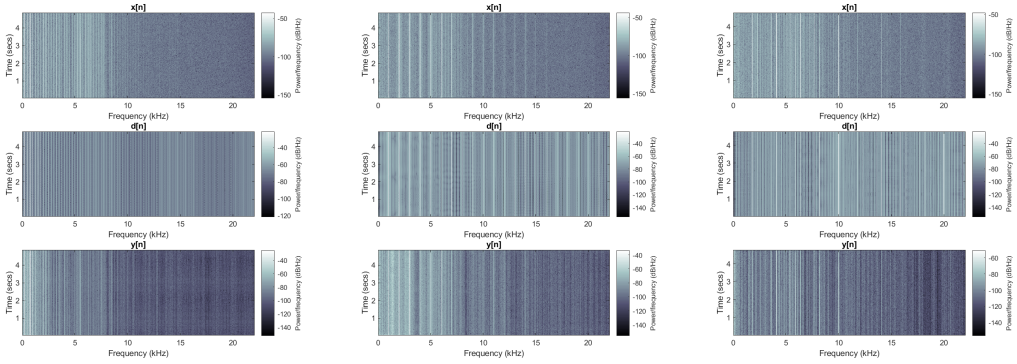
(a) 250 Hz (b) 1 kHz (c) 10 kHz

Figura 139: Costo por batch del entrenamiento de redes gamma para ondas diente de sierra.



(a) 250 Hz (b) 1 kHz (c) 10 kHz

Figura 140: Señales generadas por las redes gamma para ondas diente de sierra.



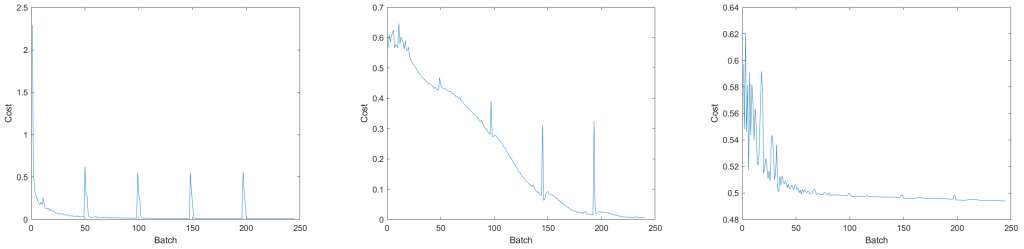
(a) 250 Hz (b) 1 kHz (c) 10 kHz

Figura 141: Espectrogramas generados por las redes gamma para ondas diente de sierra.

Señal	Error promedio
Onda sinusoidal 250 Hz	0.52737
Onda sinusoidal 1 kHz	0.62787
Onda sinusoidal 10 kHz	0.63105
Onda cuadrada 250 Hz	0.76827
Onda cuadrada 1 kHz	1.0275
Onda cuadrada 10 kHz	0.99868
Onda diente de sierra 250 Hz	0.45427
Onda diente de sierra 1 kHz	0.4715
Onda diente de sierra 10 kHz	0.49683
Onda AM	(NaN)
Onda FM	(NaN)
Onda AM y FM	0.38695
Barrido sinusoidal	(NaN)
Promedio	0.5507

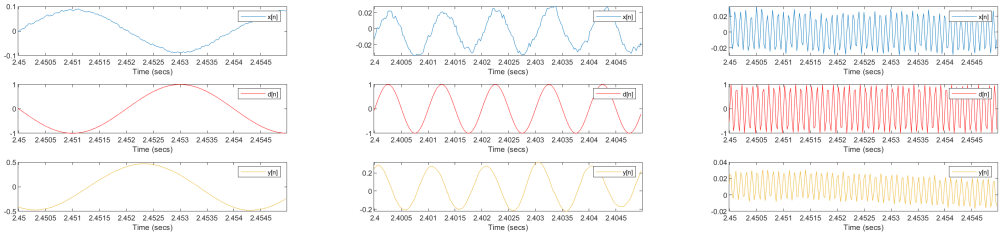
Cuadro 16: Errores promedios de la señales determinísticas por las redes gamma con activación tanh.

Activación sinusoidal



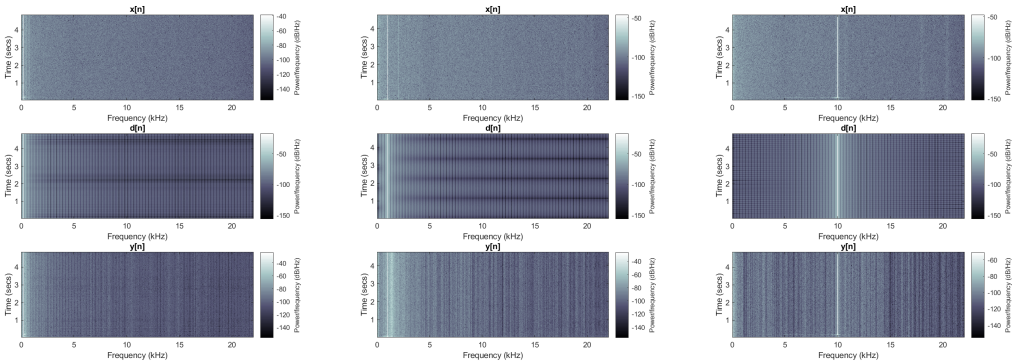
(a) 250 Hz (b) 1 kHz (c) 10 kHz

Figura 142: Costo por batch del entrenamiento de redes gamma para sinusoides.



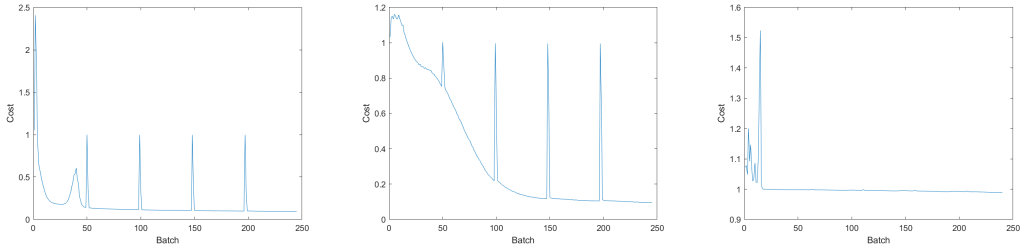
(a) 250 Hz (b) 1 kHz (c) 10 kHz

Figura 143: Señales generadas por las redes gamma para sinusoides.



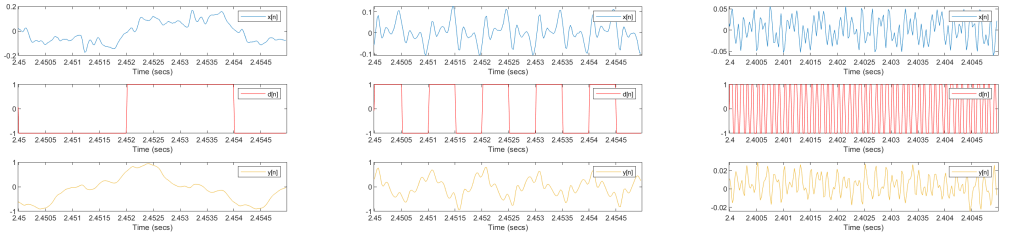
(a) 250 Hz (b) 1 kHz (c) 10 kHz

Figura 144: Espectrogramas generados por las redes gamma para sinusoides.



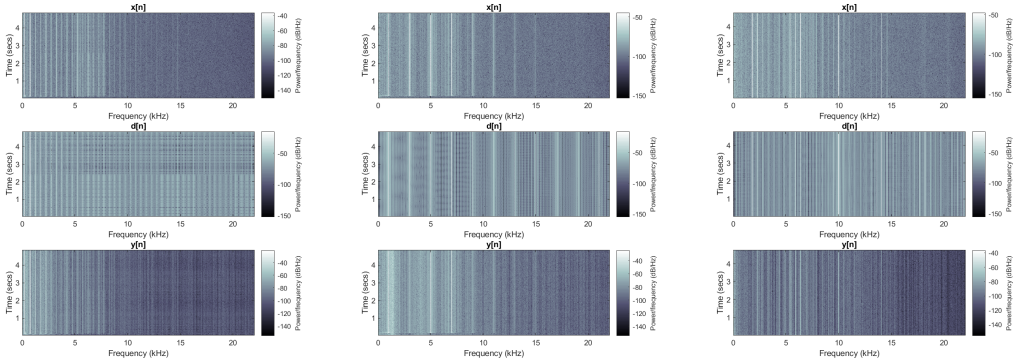
(a) 250 Hz (b) 1 kHz (c) 10 kHz

Figura 145: Costo por batch del entrenamiento de redes gamma para ondas cuadradas.



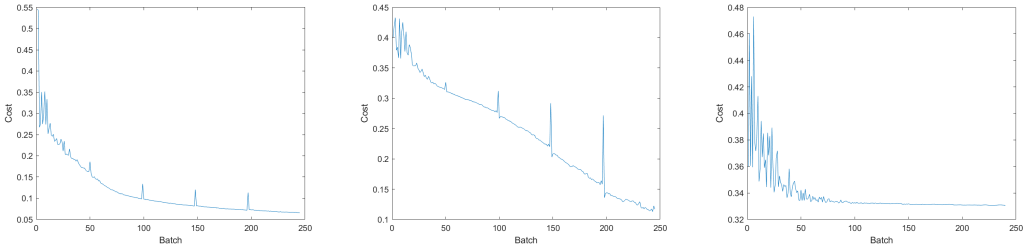
(a) 250 Hz (b) 1 kHz (c) 10 kHz

Figura 146: Señales generadas por las redes gamma para ondas cuadradas.



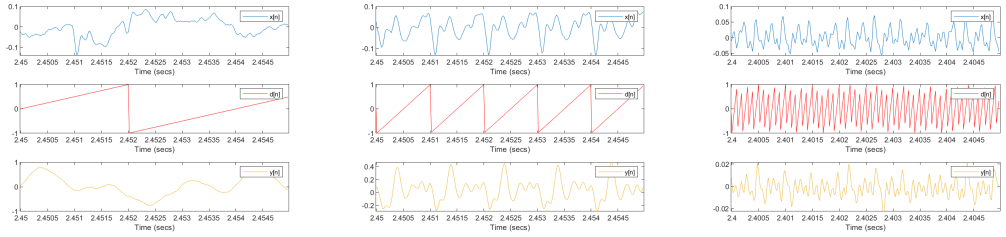
(a) 250 Hz (b) 1 kHz (c) 10 kHz

Figura 147: Espectrogramas generados por las redes gamma para ondas cuadradas.



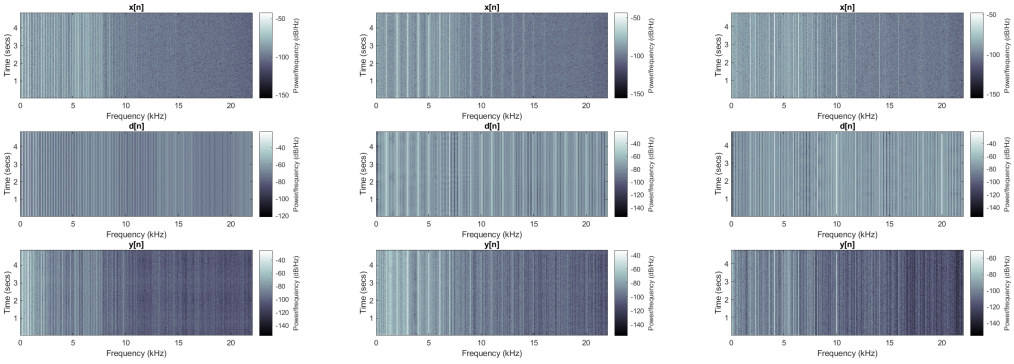
(a) 250 Hz (b) 1 kHz (c) 10 kHz

Figura 148: Costo por batch del entrenamiento de redes gamma para ondas diente de sierra.



(a) 250 Hz (b) 1 kHz (c) 10 kHz

Figura 149: Señales generadas por las redes gamma para ondas diente de sierra.



(a) 250 Hz (b) 1 kHz (c) 10 kHz

Figura 150: Espectrogramas generados por las redes gamma para ondas diente de sierra.

Señal	Error promedio
Onda sinusoidal 250 Hz	0.55403
Onda sinusoidal 1 kHz	0.59389
Onda sinusoidal 10 kHz	0.63096
Onda cuadrada 250 Hz	0.76849
Onda cuadrada 1 kHz	0.93699
Onda cuadrada 10 kHz	0.99604
Onda diente de sierra 250 Hz	0.44838
Onda diente de sierra 1 kHz	0.48929
Onda diente de sierra 10 kHz	0.49531
Onda AM	(NaN)
Onda FM	(NaN)
Onda AM y FM	0.44235
Barrido sinusoidal	(NaN)
Promedio	0.6356

Cuadro 17: Errores promedios de la señales determinísticas por las redes gamma con activación sinusoidal.

10.2.2. Red neuronal por lista de reproducción

Costo

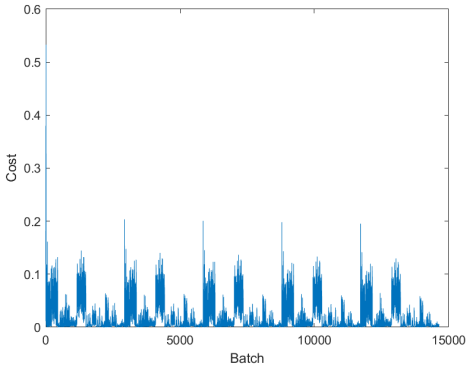


Figura 151: Evolución del costo por batch para la red gamma con activación sigmoide.

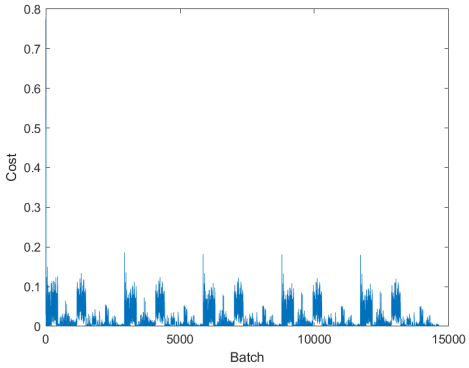


Figura 152: Evolución del costo por batch para la red gamma con activación tanh.

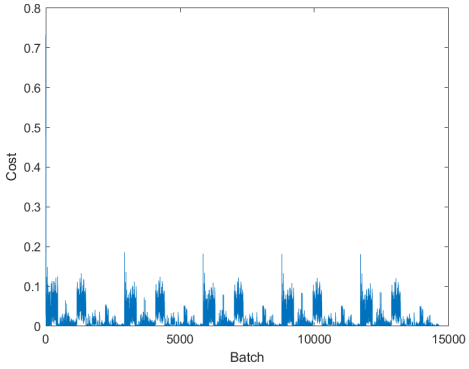
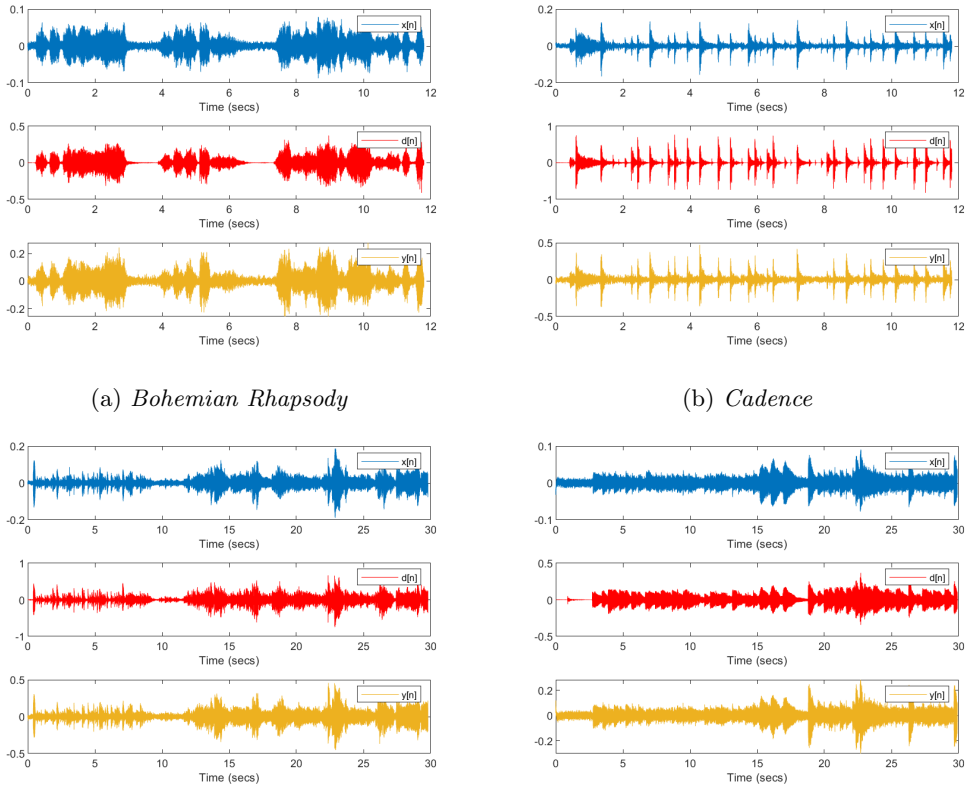


Figura 153: Evolución del costo por batch para la red gamma con activación sinusoidal.

Filtrado



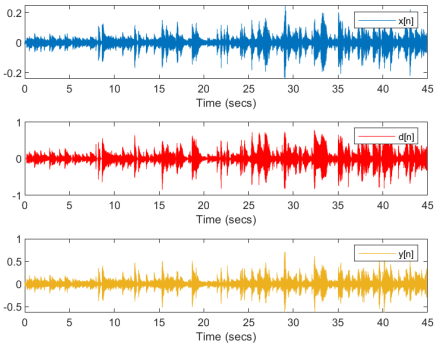
(a) *Bohemian Rhapsody*

(b) *Cadence*

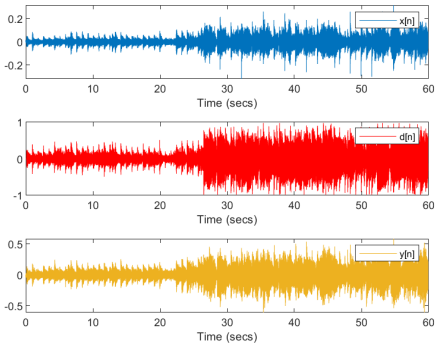
(c) *Peru*

(d) *Week No. 8*

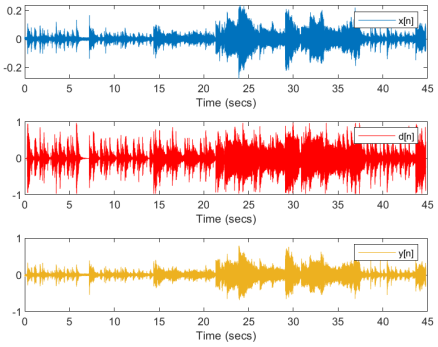
Figura 154: Señales del filtrado de clips de un solo instrumento por la gamma con activación sinusoidal.



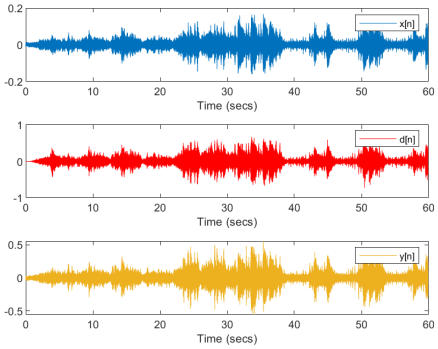
(a) *Lonely Cat*



(b) *Karma Police*

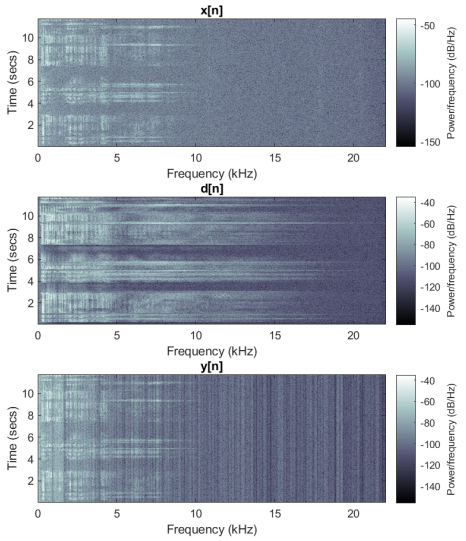


(c) *Atlantic Limited*

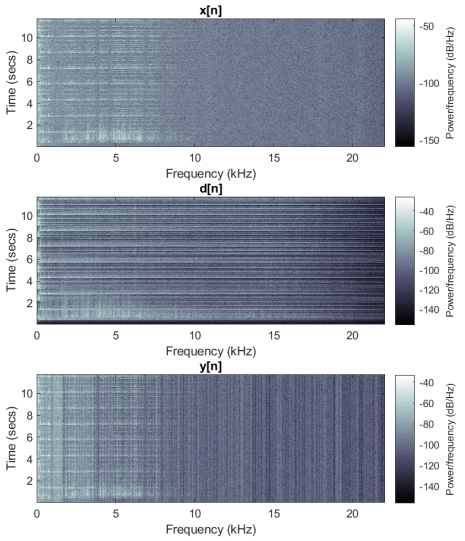


(d) *The Blue Danube*

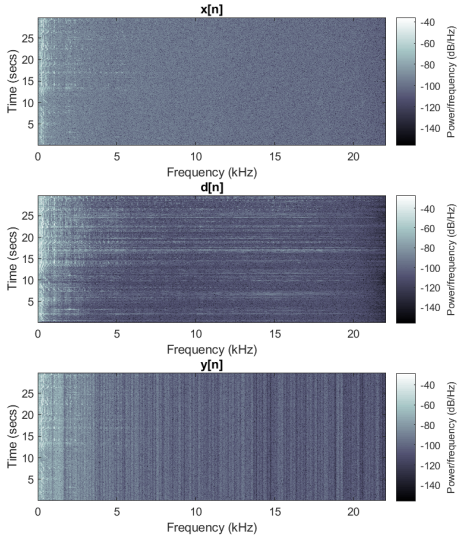
Figura 155: Señales del filtrado de clips de varios instrumentos por la gamma con activación sinusoidal.



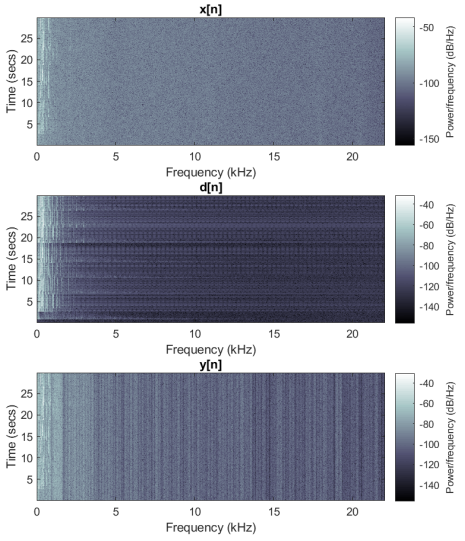
(a) *Bohemian Rhapsody*



(b) *Cadence*

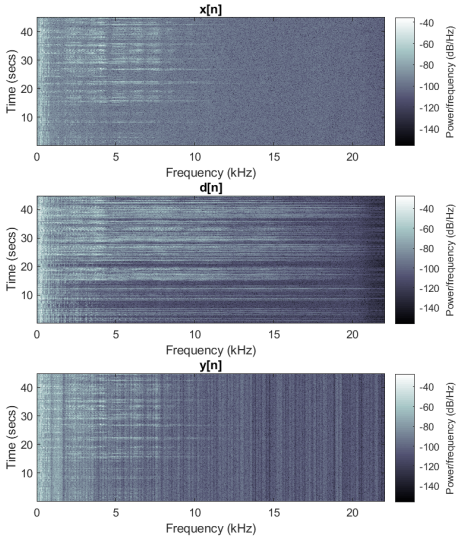


(c) *Peru*

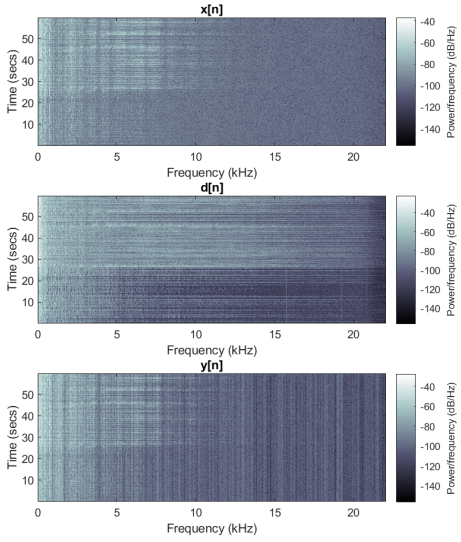


(d) *Week No. 8*

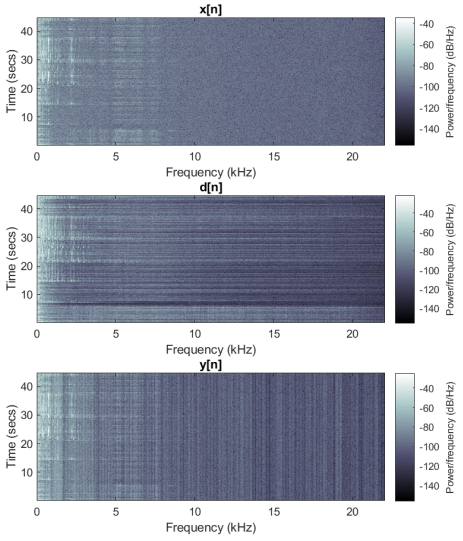
Figura 156: Espectrogramas del filtrado de clips de un solo instrumento por gamma con activación sinusoidal.



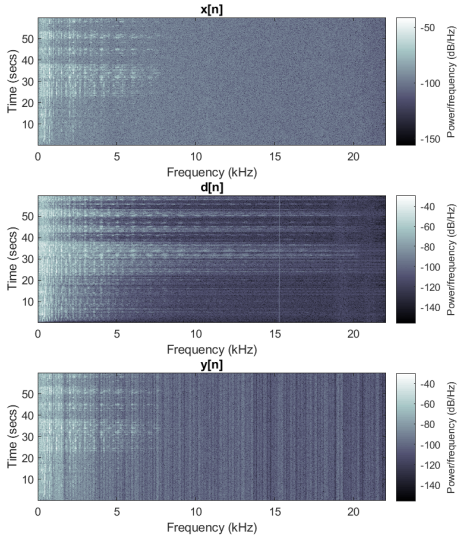
(a) *Lonely Cat*



(b) *Karma Police*



(c) *Atlantic Limited*



(d) *The Blue Danube*

Figura 157: Espectrogramas del filtrado de clips de varios instrumentos por gamma con activación sinusoidal.

Señal	Error promedio
Bohemian Rhapsody	0.034049
Cadence	0.047798
Peru	0.048914
Week No. 8	0.033494
Lonely Cat	0.049778
Karma Police	0.1325
Atlantic Limited	0.12301
The Blue Danube	0.064136
Promedio	0.0667

Cuadro 18: Errores promedios de los clips musicales por las redes gamma con activación sinusoidal.

10.3. Discusión de resultados

10.3.1. Investigación anterior

Los resultados aquí presentados muestran la eficiencia del algoritmo con señales determinísticas sencillas, tal como se expuso en la investigación hecha por Chang et al [3]. Recuérdese la Figura 1, en la que se muestra la señal de prueba $x(t) = 0.3(\sin(1.28\omega t))+0.5(\cos(3.93\omega t))$, con $\omega = 200\pi$ a la cual aplicaron deconvolución al introducirla en una red TDNN. Dicha implementación se contrasta con la puesta a prueba en este trabajo, en relación a la arquitectura y resultados generados.

Las arquitecturas fueron diferentes, así como la tasa de aprendizaje, pero también hubo elementos en común. Como primera instancia está la cantidad de nodos en las distintas capas. Chang *et al.* emplearon una red TDNN de 100 nodos en la capa de entrada y 30 nodos en la capa oculta, cantidades menores a los 250 nodos aquí aplicados. La señal de prueba y el par de sinusoides también fue diferente. Como último punto está el entrenamiento y función de activación, el cual fue *backpropagation* y tanh. Sin embargo, una diferencia considerable fue que ellos usaron el algoritmo *delta-bar-delta* para cambiar la tasa de aprendizaje, en vez de la tasa constante aquí usada.

A pesar de los discrepancias y simuilitudes entre el intento por Chang y este trabajo, en ninguno se logra la deconvolución aceptable de pistas musicales. En el primero, no se puso a prueba la red TDNN bajo esta señales más complejas. Por otro lado, en este trabajo con dichas señales no se obtuvo la fidelidad en audio deseado. Nótese los espectrogramas por ambas arquitecturas (Figuras 122, 123, 156 y 157) en donde la señales $x[n]$ y $d[n]$ en el dominio de la frecuencia no son comparables.

10.3.2. Desempeño de las diferentes funciones de activación

Las diferencias de implementación entre las funciones de activación son palpables en los resultados generados al dejar pasar las señales determinísticas por las redes TDNN. Se observa de primero que para señales sencillas como el sinusoide de 250Hz, las tres funciones

de activación dieron a simple vista el mismo resultado (Figuras 73, 88 y 103). Ya para una señales con mayor cantidad de armónicos, como el diente de sierra de 1kHz, el mejor filtrado fue para las activaciones sinusoidal y la de tanh (Figuras 94 y 109). Esto fue por la pendiente de las funciones, en la que la función sigmoide tiene una pendiente menor en el centro que las otras dos. Como detalle final, obsérvese de nuevo los espectrogramas para el caso anterior. Se distingue que la reconstrucción de las señales por parte de las redes dependió de las frecuencias rescatadas del ruido.

En donde tuvieron problemas las tres funciones de activación en la TDNN fue con la señal *chirp*. Para costo calculado de esta señal con las tres funciones de activación, los tres casos se establecieron a un costo medio de 0.5 durante el entrenamiento (Figuras 84, 99 y 114). Este valor se refleja en el dominio temporal, en donde para el tramo escogido, ninguna de las señales de salida ni siquiera coincidieron en amplitud con $d[n]$ (Figuras 85, 100 y 115). A pesar de esto, los espectrogramas generados indican que los componentes frecuenciales se mantuvieron (Figuras 86, 101 y 116). Esto pudo haber ocurrido porque ya de por sí estos componentes eran lo suficientemente prominentes en la señal perturbada $x[n]$.

10.3.3. TDNN contra la *Focused Gamma*

La red neuronal TDNN contra su generalización *Focused Gamma* demostró la capacidad de realizar un mejor filtrado para señales determinísticas. El efecto de introducir un factor de memoria μ se mostró contraproducente para los hiper-parámetros escogidos. Compárese primero la diferencia de salidas por parte de las dos arquitecturas con los sinusoides de 250Hz para la función de activación sinusoidal (Figuras 103 y 143). Inmediatamente se nota un desfase temporal por parte de la arquitectura *Gamma*. Después comparando el resultado anterior de la *Gamma* con la salida generada por la función de activación sigmode (Figura 125) se nota que un atraso mayor por parte de $y[n]$. De esto se extrae el hecho que la introducción de un factor de memoria en la capa de entrada solo hizo el proceso de aprendizaje más lento.

10.3.4. Redes con las pistas musicales

Tanto para la arquitectura TDNN como *Focused Gamma*, al momento de entrenar la red con todas las pistas musicales y filtrar cada una de las canciones, el efecto generado fue un ordenamiento en el ruido. Este efecto se distingue en el dominio de la frecuencia, en donde se pasó de un ruido aleatorio a bandas ordenadas de frecuencia. Cada una de las funciones de activación presentaron casos meramente idénticos, únicamente cambiando la posición de las susodichas bandas. Al ver cómo cambió el costo durante el aprendizaje, de nuevo se nota que el mayor cambio únicamente sucede al principio. Este apunta a que estas bandas son definidas por la red en el primer *batch* del entrenamiento y que cambian levemente en los *batches* consecutivos.

10.3.5. Ruido

Un factor principal en el funcionamiento de la redes fue la cantidad de ruido presente en la señal perturbada $x[n]$. Parte del contenido espectral de la señal grabada fue ahogada por un suelo de ruido, especialmente en las frecuencias altas. Esto dependió directamente de la respuesta en frecuencia del sistema (Figura 14). Hay que tomar en cuenta que también dependió de la relación señal-ruido que existió en tal señal grabada.

Hay que considerar las figuras que se están visualizando, para saber el papel que está jugando el ruido. Se recuerda que los espectrogramas emplean la FFT en una ventana definida que barre la duración total de la señal a analizar. Otra herramienta de visualización sería el PSD (*Power spectral density*) que se define como la autocorrelación de la señal en el dominio de Fourier. En una PSD, por ejemplo, ruido blanco no correlacionado no se manifiesta, únicamente mostrando la intensidad (energía) con la que varían las frecuencias de interés [23].

10.3.6. Sobre el aprendizaje de las redes

Un detalle que se presentó en algunas curvas de aprendizaje fue una serie de picos. En el aprendizaje de las señales cuadradas por la red TDNN con activación sinusoidal (Figura 105), estos picos fueron de magnitud considerable. Esto se debió primordialmente a que la señal fue capturada por el micrófono con cierto delay. Por lo que en un tramo inicial de $x[n]$, no se manifestó la señal de prueba. Es por esto que los picos en el costo coinciden con las epochs escogidas (Cuadro 4).

En el aprendizaje de todas las pistas musicales, las arquitecturas TDNN y *Focused Gamma* parecen estancarse. En las diferentes gráficas de aprendizaje (Figuras 117-119 y 151-153), la evolución del costo se repitió cada vez que se entrenaba la red con la lista de canciones. Esto se traduce a que la red dejó de aprender. Ya que este no fue el caso con señales determinísticas, esto indica que estas redes necesitan varias pasadas del mismo set de frecuencias antes de que el contenido espectral cambie. Esto no es posible para una pista de audio, cuyo contenido espectral cambia cada instante.

10.3.7. Redes neuronales contra los filtros adaptativos

Las arquitecturas e hiper-parámetros escogidos para las redes neuronales aquí implementadas mostraron diferencias con respecto a los filtros adaptativos puestos a prueba. Estas discrepancias se analizaron en el filtrado de los distintos clips musicales. La diferencia primordial fue el contenido espectral obtenido y calidad en las señales de salida $y[n]$. Esto invita a analizar la información que emplea cada algoritmo, el costo de implementación y el objetivo original de prescindir de la señal $d[n]$.

Una de las características más prominentes de los filtros adaptativos es el uso de una línea de feeback, a diferencia de las redes neuronales. Las variantes LMS son las que utilizan el error de forma explícita para encontrar los pesos. Para el RLS este error es implícito en el vector $\mathbf{p}$ de correlación cruzada. Lo importante es que los filtros adaptativos realizan

correcciones en tiempo real con base en la salida, para así corregir el vector de pesos **w**. Por otra parte, las redes neuronales únicamente emplean el error durante el entrenamiento. Así que al momento en que una red neuronal filtre una pista nunca antes vista, el error será mucho mayor en una pista utilizada durante el entrenamiento.

Las redes neuronales solo emplearon el contenido espectral visible por encima del suelo de ruido, mientras los filtros adaptativos no. De todas las pistas de audio filtradas por las diferentes redes, estas no introdujeron nuevo contenido espectral. Por otro lado, los filtros adaptativos sí amplificaron las frecuencias de toda la banda espectral mostrada en los distintos espectrogramas. A pesar des esto, las redes no distorsionaron el contenido espectral de interés, lo cual sí hicieron ambas variaciones del filtro LMS.

Sopesando las redes neuronales contra el mejor de los filtros adaptativos, el RLS, se observan las implicaciones de implementar cada uno. Lo primero a considerar es el costo computacional, el cual es únicamente alto para la redes durante el entrenamiento. Ya al usar las redes en *feedforward*, el costo es menor que el lema de inversión de matriz implementado por el RLS. En este punto, se recalca de nuevo el hecho de que las redes entrenadas ya no necesitan de de la señal $d[n]$, a diferencia de un filtro adaptativo. En síntesis, las redes neuronales lograron una reducción considerable del ruido en diferentes bandas, sin la necesidad de la señal deseada, pero con una fidelidad en audio menor que la obtenida por el filtro RLS.

10.3.8. Posibles alternativas

RNN (*Recurrent Neural Network*)

Una RNN puede ser una alternativa a las arquitecturas aquí propuestas. En una red neuronal *Focused Gamma* se introdujo un grado de memoria μ en la capa de entrada. Tal como se mencionó anteriormente, este modificó el input original y obtuvo una salida con mayor error que la TDNN. Por otra parte, una RNN tradicional relaciona data anterior con la actual sin modificar el input, lo cual hace manifiesto en un vector de estado oculto. Como otro punto a favor, uno de los modelos RNN toma las salidas anteriores como entradas [24].

Redes neuronales convolucionales

Las redes convolucionales son un tipo de redes que se usan con imágenes, sin embargo estas podrían ser utilizadas para el procesamiento de audio. El empleo de estas redes requiere de una imagen que represente el audio. A diferencia de las pruebas aquí realizadas, lo mejor sería la representar la información en el dominio de la frecuencia. Así que los espectrogramas generados serían los introducirían a la red [25]. Se aleja de lo aquí aplicado, en donde la data de entrada se mostraba a la red en el dominio del tiempo. Lo cual señala a que una de las posibles razones de la baja fidelidad en audio de la redes TDNN fue en cuanto extracción de características, en este caso, información temporal en vez de espectral.

CAPÍTULO 11

Conclusiones

Los filtros adaptativos y redes neuronales implementadas lograron cada uno en cierto grado la deconvolución y la eliminación del ruido. De los distintos filtros adaptativos, el RLS convencional fue el que tuvo el mejor desempeño. En cuanto a la redes neuronales, estas se entrenaron cada una con 8 pistas musicales de distinta complejidad y género musical. Sin embargo, ya entrenadas solo lograron atenuar parte del ruido de fondo.

De los tres filtros adaptativos implementados, el que presentó el menor error para todas las señales de prueba fue el RLS convencional (Cuadros 9 y 10). Ambas variedades del LMS lograron la atenuación de ruido de fondo, pero introdujeron distorsiones indeseadas. El filtro RLS al tener el menor error (1.9240e-12), logró la mayor fidelidad en audio, generando una señal de salida $y[n]$ indistinguible de la señal deseada $d[n]$.

Se escogieron distintas señales de audio, seleccionadas para entrenar las distintas redes neuronales. Los Cuadros 1 y 2 muestran las señales usadas para el entrenamiento de las redes TDNN y *Focused Gamma*. De primero se usaron señales determinísticas como base. Luego se emplearon clips musicales que representaron casos donde solo se escuchaba un único instrumento o varios instrumentos a la vez. Además, se incluyó diferentes géneros tales como *rock*, *jazz* y clásica.

Las arquitecturas TDNN y *Focused Gamma* lograron en cierto grado disminuir del ruido de fondo. Esto es palpable en los diferentes espectrogramas (Figuras 122, 123, 156 y 157) en los cuales se observa un efecto de atenuación en distintas bandas de frecuencia. A pesar de esto, su fidelidad en audio fue mucho menor que el del filtro RLS convencional.

CAPÍTULO 12

Recomendaciones

Una opción sería el cambio en la arquitectura, probar con una red RNN o con una red convolucional. La red RNN no aprende solo durante el entrenamiento, sino que también extrae información de inputs históricos. Este tipo de red, adicionalmente, involucra un vector de estado oculto en el cada nodo influye sobre sus nodos vecinos. La otra vía es una red neuronal convolucional, en la cual los espectrogramas son los que entrarían en la red. Dicha implementación tendría la ventaja de observar otras características de las señales de audio, en este caso, componentes espectrales.

Se proponen otras opciones para la mejora de los resultados aquí presentados. Primero, estudiar más sobre el entrenamiento que requieren las redes neuronales (para un determinado set de parámetros) y sobre la extracción de características. Otra alternativa es el modificar la tasa de aprendizaje del descenso por gradiente. En vez de una tasa de aprendizaje constante β, experimentar con una tasa de paso variable. Por último, puede ser conveniente visualizar los resultados bajo una transformación de PSD.

CAPÍTULO 13

Bibliografía

[1] *ERA 4 Reverb Remover User Manual*, Accusonus Inc. dirección: https://accusonus.com/manuals/era-reverb-remover-user-manual.

[2] *Dialogue De-reverb*, iZotope, Inc. dirección: https://www.izotope.com/en/products/rx/features/dialogue-de-reverb.html.

[3] P. Chang, C. G. Lin y B. Yeh, "Inverse filtering of a loudspeaker and room acoustics using time-delay neural networks", *The Journal of the Acoustical Society of America*, vol. 95, n.º 6, págs. 3400-3408, 1994. DOI: 10.1121/1.409959. eprint: https://doi.org/10.1121/1.409959. dirección: https://doi.org/10.1121/1.409959.

[4] F. H. Glanz y W. T. Miller, "Deconvolution and nonlinear inverse filtering using a neural network", en *International Conference on Acoustics, Speech, and Signal Processing,*, 1989, 2349-2352 vol.4.

[5] M. Hayes, *Statistical Digital Signal Processing and Modeling*. Wiley, 1996. dirección: https://books.google.com.gt/books?id=N%5C_VSAAAAMAAJ.

[6] D. Reay, *Digital Signal Processing Using the ARM Cortex M4*. Wiley, 2015, ISBN: 9781118859049. dirección: https://books.google.com.gt/books?id=CdSOCQAAQBAJ.

[7] P. Diniz, *Adaptive Filtering: Algorithms and Practical Implementation*, third edition. Springer, 2008.

[8] T. Rashid, *Make Your Own Neural Network: A Gentle Journey Through the Mathematics of Neural Networks, and Making Your Own Using the Python Computer Language*. CreateSpace Independent Publishing Platform, 2016, ISBN: 9781530826605. dirección: https://books.google.com.gt/books?id=Zli%5C_jwEACAAJ.

[9] M. A. Nielsen, *Neural Networks and Deep Learning*. Determination Press, 2015.

[10] MissingLink.ai. (). 7 Types of Neural Network Activation Functions: How to Choose?, dirección: https://missinglink.ai/guides/neural-network-concepts/7-types-neural-network-activation-functions-right/.

[11] Y. Hu y J. Hwang, *Handbook of Neural Network Signal Processing*, ép. Electrical Engineering & Applied Signal Processing Series. CRC Press, 2018, ISBN: 9781420038613. dirección: https://books.google.com.gt/books?id=ws-fBwAAQBAJ.

[12] N. Pramanik. (sep. de 2019). Kernel Regression — with example and code, dirección: https://towardsdatascience.com/kernel-regression-made-easy-to-understand-86caf2d2b844.

[13] J. Sopena, E. Romero y R. Alquezar, "Neural networks with periodic and monotonic activation functions: a comparative study in classification problems", feb. de 1999, 323-328 vol.1, ISBN: 0-85296-721-7. DOI: 10.1049/cp:19991129.

[14] C. Olah. (ago. de 2015). Calculus on Computational Graphs: Backpropagation, dirección: http://colah.github.io/posts/2015-08-Backprop/.

[15] A. Lai. (nov. de 2018). Gradient Descent for Linear Regression Explained, dirección: https://blog.goodaudience.com/gradient-descent-for-linear-regression-explained-7c60bc414bdd.

[16] V. Suárez-Paniagua, "Deep Learning for Information Extraction in the Biomedical Domain", Tesis doct., jul. de 2019.

[17] J. Principe, B. Vries, J.-M. Kuo y P. Oliveira, "Modeling Applications with the Focused Gamma Net.", ene. de 1991, págs. 143-150.

[18] G. Horváth, "CMAC: Reconsidering an old neural network", 2003.

[19] iHome. (2008). Model iHM9/iHM10 - Stero speaker system plays all MP3 players, dirección: https://cdn.ihomeaudio.com/media/product/files/iHM9_iHM10_IB.pdf.

[20] ASUS. (jul. de 2015). Notebook PC E-Manual, dirección: https://dlcdnets.asus.com/pub/ASUS/nb/X555LA/0409_E10576_X555LA_LD_LN_EM_V4_B.pdf.

[21] Matlab. (). Measure Frequency Response of an Audio Device, dirección: https://la.mathworks.com/help/audio/examples/measure-frequency-response-of-an-audio-device.html.

[22] R. Parekh, *Principles of Multimedia*. Tata McGraw-Hill, 2006, ISBN: 9780070588332. dirección: https://books.google.com.gt/books?id=TaNmc2IdNVwC.

[23] C. R. International. (). What is power spectral density function?, dirección: https://www.cygres.com/OcnPageE/Glosry/SpecE.html.

[24] A. Amidi y S. Amidi. (). Recurrent Neural Networks cheatsheet, dirección: https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks.

[25] A. Koretzky. (feb. de 2019). Audio AI: isolating vocals from stereo music using Convolutional Neural Networks, dirección: https://towardsdatascience.com/audio-ai-isolating-vocals-from-stereo-music-using-convolutional-neural-networks-210532383785.

CAPÍTULO 14

Anexos

El repositorio de los algoritmos implementados en MATLAB se pueden consultar en el siguiente link: `https://github.com/cmlf1598/Deconvolucion-acustica/tree/master`

Los demás resultados generados se pueden consultar en el siguente Drive: `https://drive.google.com/drive/folders/1a-0G70v7OJeoXp6fLPR2dKKlUsyoL4o8?usp=sharing`

CAPÍTULO 15

Glosario

ADR: Automated Dialog Replacement. 4

bit: Unidad mínima de información, la cual puede ser cero o uno. 5, 22

delay: Tiempo de atraso. 111

epoch: Hiperparámetro del descenso por gradiente que determina el número de pasadas completas por la data de entrenamiento. 111

FIR: Finite Impulse Response. 15, 21

MIMO: Multiple-Input Multiple-Output. 18

RK4: Runge-Kutta 4. 4

shallow: De poca profundidad. 11

SISO: Single-Input Single-Output. 18

string: Secuencia finita de caracteres. 5