DISEÑO DE UNA RED NEURONAL ARTIFICIAL PROFUNDA PARA LA IDENTIFICACIÓN DE FALENCIAS RÍTMICAS EN BAILARINES APRENDICES

ANDRES FELIPE PAREDES TAFUR
2141456
CAMILA ANDREA MARIN MARTINEZ
2157287

UNIVERSIDAD AUTÓNOMA DE OCCIDENTE
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE AUTOMÁTICA Y ELECTRÓNICA
PROGRAMA INGENIERÍA MECATRÓNICA
SANTIAGO DE CALI
2021

DISEÑO DE UNA RED NEURONAL ARTIFICIAL PROFUNDA PARA LA IDENTIFICACIÓN DE FALENCIAS RITMICAS EN BAILARINES APRENDICES

ANDRES FELIPE PAREDES TAFUR
CAMILA ANDREA MARIN MARTINEZ

Pasantía de investigación para optar al título de Ingeniero Mecatrónico

Director
DAVID FERNANDO RAMIREZ MORENO
Doctor en Ciencias Biomédicas

UNIVERSIDAD AUTÓNOMA DE OCCIDENTE
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE AUTOMÁTICA Y ELECTRÓNICA
PROGRAMA INGENIERÍA MECATRÓNICA
SANTIAGO DE CALI
2021

Nota de aceptación:

Aprobado por el Comité de Grado, en cumplimiento de los requisitos exigidos por la Universidad Autónoma de Occidente, para optar al título de Ingeniero Mecatrónico

JOSÉ LUIS PANIAGUA
Jurado

Santiago de Cali, 23 de noviembre de 2021

AGRADECIMIENTOS

Queremos agradecer…

A nuestras familias, por la perseverancia, por el amor, por la dedicación y la espera.

A nuestros amigos, que nos apoyaron sin desfallecer.

Al profesor David Ramírez, por su apoyo y disposición incondicional en cada etapa de este proyecto.

A todos las personas de los grupos de baile que nos colaboraron y permitieron que empezáramos a avanzar en este proyecto ya no desde lo teórico, sino desde lo práctico.

CONTENIDO

LISTA DE CUADROS

pág.

LISTA DE FIGURAS

pág.

LISTA DE ANEXOS

pág.

GLOSARIO

COREOGRAFÍA: conjunto pautado de pasos, figuras y movimientos de un baile o una danza[1].

CSV: un archivo csv (valores separados por comas) es un tipo especial de archivo que puede crear o editar en Excel[2].

DATASET: base de datos para el entrenamiento de la red.

DATO: cifra, letra o palabra que se suministra a la computadora como entrada y la máquina almacena en un determinado formato; también definido como cantidad o magnitud que se cita en el enunciado de un problema y que permite hallar el valor de las incógnitas[3].

DESVIACIÓN ESTÁNDAR: es una medida estadística para la dispersión de los datos; cuanto más dispersa está una distribución, mayor será su desviación estándar.[4]

ESPECTROGRAMA: es una representación visual de las variaciones de la frecuencia en el eje vertical, y de la intensidad del sonido mediante niveles de colores a lo largo del tiempo que se representa en el eje horizontal[5].

---

[1] LEXICO BY OXFORD. Coreografía. [en línea] Oxford Languages. [Consultado: 20 de septiembre de 2021] Disponible en: https://www.lexico.com/es/definicion/coreografia

[2] MICROSOFT. Crear o editar archivos .csv para importarlos a Outlook. [En línea] Support Microsoft. [Consultado: 10 de agosto de 2021] Disponible en: https://support.microsoft.com/es-es/office/crear-o-editar-archivos-csv-para-importarlos-a-outlook-4518d70d-8fe9-46ad-94fa-1494247193c7

[3] LEXICO BY OXFORD, Dato. [en línea] Oxford Languages. [Consultado: 20 de septiembre de 2021]. Disponible en: https://www.lexico.com/es/definicion/dato

[4] KHAN ACADEMY. The idea of spread and standard deviation. [En línea] khanacademy.org [Consultado: 12 de junio de 2021] Disponible en: https://www.khanacademy.org/math/statistics-probability/summarizing-quantitative-data/variance-standard-deviation-population/a/introduction-to-standard-deviation

[5] AYUNTAMIENTO DE GRANADA. Información General: Análisis espectral. [En línea] Granada, España. (20 de mayo de 2014) [Consultado 20 de agosto de 2021] Disponible en:

FILTROS: detectores de características.

FOTOGRAMA: un fotograma es cada una de las imágenes impresas químicamente en la tira del celuloide del cinematógrafo; un fotograma por segundo se refiere al número de fotogramas o frames que se muestran por segundo en un video o película[6].

FUNCIONES ANÓNIMAS: es una función que no se encuentra almacenada en ningún archivo de programa y puede aceptar múltiples entradas pero retorna un única salida; pueden contener una sola sentencia ejecutable[7].

INICIALIZACIÓN DE KAIMING HE: es un método de inicialización para redes neuronales que hacen uso de funciones de activación no lineales, como la ReLU, por medio de un método de inicialización adecuado que evita reducir o amplificar las magnitudes de las señales de entrada de forma exponencial[8].

INICIALIZACIÓN DE XAVIER: inicialización mediante la cual los gradientes tienen la misma – o la más similar- varianza antes y después de cada capa, a través de una distribución normal con media cero y una desviación estándar igual a uno sobre raíz m, donde m es la media del fa-in y del fan-out[9].

---

https://www.granada.org/inet/sonidos.nsf/d483b298c3f6a1b9c1257cdd00384c53/3fdfc36a7489b607c1257cde0024bb34!OpenDocument

[6] MARTIN, Eugenia. ¡Conoce qué es un fotograma y cómo están hechas las películas! [en línea] Crehana. (9 de marzo de 2021) [Consultado: 20 de agosto de 2021] Disponible en: https://www.crehana.com/co/blog/video/que-es-un-fotograma/

[7] MATLAB. Anonymous Functions. [en línea] Mathworks. [Consultado 2 de julio de 2021] Disponible en: https://www.mathworks.com/help/matlab/matlab_prog/anonymous-functions.html

[8] HE, Kaiming, et al. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. [En línea] USA. (6 de febrero de 2015). [Consultado: 15 de julio de 2021]. Disponible en: https://arxiv.org/pdf/1502.01852v1.pdf

[9] BERZAL, Fernando. Entrenamiento de redes neuronales. [diapositivas] Universidad de Granada. Granada, España. 37 diapositivas. [Consultado 20 de agosto de 2021]. Disponible en: https://elvex.ugr.es/decsai/deep-learning/slides/NN4%20Training.pdf

INTERPOLACIÓN LINEAL: es un método de ajuste de datos que pasa por todos los puntos, es la interpolación más simple y busca trazar una recta entre dos puntos a partir de dos coordenadas dadas.

MATLAB: lenguaje para alto rendimiento para computación técnica. Integra computación, visualización y programación en un ambiente fácil de usar donde los problemas y soluciones son expresada en notación matemática[10].

MATRIZ: es un arreglo rectangular bidimensional de elementos de datos dispuestos en filas y columnas; los elementos pueden ser números, valores lógicos, fechas y horas o cadena de texto[11].

OHE (ONE HOT ENCODING): técnica que convierte una características categórica en una serie de características numéricas binarias, una por categoría[12].

PROMEDIO: cociente de dividir la suma de varias cantidades por el número de ellas[13].

PYTHON: es un lenguaje interpretado, orientado a objetos y de alto nivel de programación con semántica dinámica.[14]

RNA: acrónimo de Red Neuronal Artificial.

---

[10] COOPERATIVE INSTITUTE FOR METEOLOGICAL SATELLITE STUDIES. What is Matlab? [En línea] University of Wisconsin – Madison. CIMSS. [Consultado: 2 de julio de 2021] Disponible en: https://cimss.ssec.wisc.edu/wxwise/class/aos340/spr00/whatismatlab.htm

[11] MATLAB. Creating, Concatening and Expanding Matrices. Op. cit., Disponible en: https://la.mathworks.com/help/matlab/math/creating-and-concatenating-matrices.html?lang=en

[12] MISHRA, Abhishek. Data Exploration and Preprocessing. Machine Learning for iOS Developers. [en línea] John Wiley & Sons, Ltd. 2020. [Consultado: 30 de agosto de 2021] Capítulo 3. Disponible en https://onlinelibrary.wiley.com/doi/epub/10.1002/9781119602927

[13] GAMBOA DE BUEN, Berta. El uso de la palabra PROMEDIO y su significado matemático. [En línea] Ciencia y Desarrollo. México [Consultado 10 de agosto de 2021] Disponible en: https://www.cyd.conacyt.gob.mx/?p=articulo&id=127

[14] PYTHON. What is Python?: Executive Summary. [En línea] Python.org. [Consultado: 10 de agosto de 2021] Disponible en: https://www.python.org/doc/essays/blurb/

SEÑAL: es una función representando una variable o cantidad física, y comúnmente contiene información acerca del comportamiento o la naturaleza del fenómeno[15].

VARIABLE: se define, como las cualidades, propiedades o características de los sujetos de estudio que pueden ser enumeradas o contadas (sexo, raza) o medidas cuantitativamente (peso, estatura) y cuyo valor varía de una a otra.[16]

[15] HSU, Hwei P. Signals and Systems. 1ra ed. USA: McGraw-Hill. 1995. 470 p. ISBN: 0070306419.

[16] CUESTAS, Eduardo. Variables. En: *Revista de la Facultad de Ciencias Médicas.* [En línea] Argentina: Universidad Nacional de Córdoba, agosto-septiembre 2009. vol. 66, nro. 3. p. 118. [Consultado: 15 de agosto de 2021] Disponible en: http://www.revista2.fcm.unc.edu.ar/Rev.2009.3/Variables_Cuesta.pdf

RESUMEN

La Inteligencia Artificial (IA) se ha proyectado en la última década como una herramienta innovadora, implementada en diversas áreas como la ingeniería y la medicina entre otras disciplinas, para problemas de gran complejidad cuya solución se puede hallar, con base en algoritmos matemáticos inspirados en la estructura y comportamiento neuronal de los seres vivos, con el fin de resolver los problemas analógicamente a la resolución natural de los seres simulados.

Para dar solución al proceso de mejora y control en la dinámica del baile, se presenta una alternativa que evalúa la ejecución de los movimientos correspondientes a diferentes modalidades de baile, haciendo uso de Redes Neuronales Artificiales; a su vez, se refieren como parámetros técnicos un conjunto diverso de datasets, que permiten la evaluación precisa en la ejecución de cada baile. Los datasets implementados se generaron con la herramienta de tf pose estimation, luego, fueron procesados y presentados a las diferentes estructuras de red elaboradas en los softwares de Matlab y Python, buscando el mejor rendimiento e interpretación del baile en las estructuras neurales. Posteriormente, un análisis comparativo determinó qué tipo de red se adecúa más a la necesidad presentada, este ejercicio se realizó con el uso de matrices de confusión para implementar criterios de evaluación generalizados que comprenden la exactitud, la precisión y la sensibilidad.

Finalmente, se espera que los resultados obtenidos sirvan de base para futuras investigaciones relacionadas al desarrollo tecnológico, concerniente al área del entretenimiento y permitan una profundización investigativa en la clasificación de imágenes mediante redes neuronales.

Palabras clave: RNA, MLP, CNN, identificación de baile, tf- pose- estimation, Python, Matlab.

ABSTRACT

Artificial Intelligence (AI) has been projected in the last decade as an innovative tool, implemented in various areas such as engineering and medicine among other disciplines, for highly complex problems whose solution can be found, based on mathematical algorithms inspired by the neuronal structure and behavior of living beings, in order to solve problems analogously to the natural resolution of simulated beings.

To solve the process of improvement and control in the dynamics of dance, an alternative is presented that evaluates the execution of the movements corresponding to different dance modalities, making use of Artificial Neural Networks; in turn, a diverse set of datasets are referred to as technical parameters, which allow precise evaluation in the execution of each dance. The implemented datasets were generated with the tf pose estimation tool, then, they were processed and presented to the different network structures elaborated in the Matlab and Python software, looking for the best performance and interpretation of the dance in the neural structures. Subsequently, a comparative analysis determined which type of network best suits the presented need. This exercise was carried out with the use of confusion matrices to implement generalized evaluation criteria that include accuracy, precision and sensitivity.

Finally, it is expected that the results obtained will serve as the basis for future research related to technological development, concerning the area of entertainment and allow an investigative deepening in the classification of images through neural networks.

Keywords: RNA, MLP, CNN, dance identification, tf- pose- estimation, Python, Matlab.

INTRODUCCIÓN

Uno de los retos más importantes a los que se enfrenta el ser humano en nuestra generación es la construcción de sistemas inteligentes, entendiendo por sistema, un dispositivo con capacidad de realizar la tarea que le sea requerida[17]; el desarrollo de dichas maquinarias se conoce como Inteligencia Artificial. Según Buduma[18], esta disciplina es la representación de los sueños de personas que desean construir máquinas con cerebros análogos al humano, aún así, el proceso para lograr que esos anhelos sean realidad, requiere de una implementación matemática compleja, ya que se debe simular un sistema de procesamiento informático altamente complejo, no lineal y paralelo, como lo es el cerebro.

Los avances tecnológicos dados en la interdisciplinariedad de la Inteligencia artificial han demostrado que la implementación de técnicas de programación específicas facilita la ejecución de tareas determinadas, tras un correcto entrenamiento y uso posterior de una herramienta seleccionada. Esta rama de la Inteligencia artificial se conoce como Deep Learning y se encuentra en auge en estos momentos; representada en procesos de traducción automática, identificación de lenguaje, visión computacional y análisis de datos[19], es implementada en áreas como medicina, clasificación de imágenes satelitales, ingeniería y entretenimiento.

Dentro del entretenimiento, se ha hecho uso del Deep Learning para las redes sociales, en temas como la búsqueda en archivos de vídeo, análisis de datos y de lenguaje para subtitulación[20]. No obstante, aún existen subdivisiones de este ámbito donde la implementación de la Inteligencia Artificial es poco usada.

La disciplina del baile carece de implementaciones significativas en el ámbito de Deep Learning, pues si bien se han desarrollado investigaciones y productos, éstos van mayormente enfocados a la manipulación de imágenes o generación de pasos

---

[17] ISASI VIÑUELA, Pedro e GALVÁN LEÓN, Inés M. Redes de neuronas artificiales: Un enfoque práctico. Madrid: Pearson Prentice Hall. 2004. ISBN: 8420540250

[18] BUDUMA, Nikhil. Fundamentals of Deep Learning. 1ra Edición. USA: Sebastopol. O'Reilly. 2017. ISBN: 9781491925614.

[19] VARIZANI, JYOTI. 10 Real World Examples of Deep Learning Models & AI. [En línea] Futran Solutions. [Consultado: 8 de septiembre de 2021]. Disponible en: https://www.futransolutions.com/10-real-world-examples-of-deep-learning-models-ai/

[20] AMAZON. Aprendizaje automático para aplicaciones multimedia. [En línea] USA. [Consultado: 9 de septiembre de 2021] Disponible en: https://aws.amazon.com/es/media/tech/machine-learning-for-media-applications/

a partir de secuencias o sonidos, pero pocos se centran en el análisis de la coreografía a realizar y - aún menos-, en el del ritmo ejecutado. Esta situación genera un ámbito de investigación que puede ser beneficioso tanto en lo educativo como en lo social, ya que el baile es visto como una forma de actividad social que brinda la oportunidad de relacionarse con el entorno.

La presente investigación se enfoca en la creación de un mecanismo de apoyo a los docentes en academias de baile, área que no se ha visto permeada por el avance tecnológico a nivel local, a la vez que, promueve la inmersión de la tecnología y la investigación en el sector de cultura y entretenimiento desde el ámbito académico, para su mayor potenciación y evitar su reducción a proyectos enfocados al ocio.

Para ello, se realizó la grabación y la recolección de vídeos de múltiples bailarines, tanto de forma presencial como virtual, ejecutando secuencias de pasos de los ritmos seleccionados; la duración establecida para los vídeos fue seleccionada a partir de una prueba visual evaluada de forma experimental con el grupo de investigación. Seguidamente, se implementó la herramienta de pose estimation, la cual, permitió la estimación de la posición física de los bailarines con el objetivo de elaborar un conjunto de datos que permitiera la generación de datasets específicos para la implementación de los ejemplos de redes neuronales pre seleccionadas, dichos datasets se encuentran generados en diferentes tipos de formatos, acorde a las necesidades que presenta cada red.

Durante la implementación de la herramienta de pose estimation, los datos e imágenes fueron procesados para su aplicación durante la creación de los datasets a utilizar en las diferentes estructuras de redes neuronales tipo MLP y CNN, a través de un algoritmo diseñado que organizó valores en formato de arreglo y de imágenes. Las redes implementadas se elaboraron en los softwares de Python y Matlab, con la finalidad de darle versatilidad a la investigación, puesto que Matlab es un software de uso científico cuyo uso es pago, por lo que su accesibilidad es restringida, mientras el software de Python es utilizado a nivel mundial por su concepto de licencia libre y, por consiguiente, es asequible al público para el cual está dirigido este proyecto, aunado al factor de librerías de Python, que podían brindar una variación significativa en los resultados a adquirir.

Luego, se entrenaron redes neuronales con variaciones en cantidad de capas, neuronas y funciones de activación para la presentación de diferentes estructuras, a las cuales, se les validó su funcionamiento y, con ayuda tanto de parámetros evaluativos concretos como de las métricas, se realizaron comparaciones entre estructuras y tipos de redes que permitieran definir la RNA cuyo rendimiento fue más favorable.

Finalmente, por medio de este trabajo se espera sentar precedentes investigativos para el área del entretenimiento (baile), desde el punto de vista académico, para que se continúe con la implementación de algoritmos óptimos que sirvan como base para la creación de mecanismos de aprendizaje alternos y profundos en dinámicas culturales.

1. PLANTEAMIENTO DEL PROBLEMA

El uso de Redes Neuronales Artificiales (RNAs en plural, RNA en singular) se ha convertido en uno de los avances más exitosos que tienen los algoritmos computacionales. Actualmente, las múltiples estructuras que tienen estas redes, permiten mejorar el desempeño de aplicaciones en diferentes áreas-objetivo, al procesar de forma más precisa los valores de las variables que determinan el desempeño y la calidad de las funciones realizadas por diversos sistemas. Esto ha conducido a que el uso de RNAs, gracias a su facilidad para resolver problemas de alta complejidad en comparación con técnicas convencionales, haya migrado de campos netamente académicos hacia sectores como mecánica, economía y medicina.

En medio del crecimiento interdisciplinario logrado con este mecanismo, todavía existen espacios en los que su intervención es nula, tal es el caso del baile, disciplina que, en palabras de Gardner[21], Jaramillo y Murcia[22], cumple tanto como actividad de recreación como de educación, a la vez que, se vuelve necesario para las personas como mecanismo de comunicación ante la sociedad. Para Ereño[23], su enseñanza está centrada en los modelos de ejecución, basándose en la repetición fragmentada, analítica y continua, indispensable para lograr un cierto grado de automatización en las mismas. Ahora bien, en una clase de baile donde un instructor observa, corrige y enseña a varias personas simultáneamente, el aprendizaje no es equitativo, ya que el docente no tiene la capacidad ni el tiempo de reconocer las características y destrezas de cada persona. Por lo tanto, buscando una iniciativa que sirva como apoyo a las personas que se desempeñan en ésta modalidad, se planteó la pregunta: ¿Qué red neuronal artificial puede desarrollarse para servir como auxiliar en el trabajo de mejoramiento progresivo de personas que bailan ritmos populares?

[21] GONZÁLEZ, Mónica. Bailar, ¿una profesión o una afición?. [blog]. Blogs EMagister. [Consultado 17 de mayo de 2020] Disponible en: https://www.emagister.com/blog/bailar-una-profesion-o-una-aficion/

[22] JARAMILLO ECHEVERRY, Luis Guillermo y MURCIA PEÑA, Napoleón. La danza y el baile. [En línea] Lecturas: Educación Física y Deportes. No. 46. Buenos Aires. 2002. [Consultado el 14 de mayo de 2020]. Disponible en: https://www.efdeportes.com/efd46/baile.htm

[23] EREÑO Álvarez, C. El proceso de enseñanza-aprendizaje en la danza tradicional. Alternativas metodológicas. Reflexiones desde la propia práctica [En línea] Lecturas: Educación Física y Deportes. No. 46. Buenos Aires. 2002. [Consultado el 16 de mayo de 2020]. Disponible en: https://www.efdeportes.com/efd46/baile.htm

2. JUSTIFICACIÓN

Las RNA son mecanismos programados que, en palabras de García[24], se encargan de emular ciertos rasgos del comportamiento humano, realizando tareas que reducen costos, mejoran el desempeño del personal inexperto, y el control de calidad en el área designada. A través de las RNAs se puede obtener una alta efectividad condicionada únicamente por cuestiones de estructura y rendimiento en las redes, tal es el caso de redes como el Perceptrón Multicapa (MLP) de Toro y Lizarazo[25], usada para la clasificación de imágenes satelitales, donde se obtuvieron resultados con un 95.10% de asertividad, también aplica para la MLP y la Red Neuronal Artificial Recurrente (RNN), para el desarrollo de un sistema humano - humanoide mediante el reconocimiento y aprendizaje del lenguaje corporal de González[26], que logró la creación de una interfaz de comunicación escrita e interacción física con el humanoide.

Ahora, Jarque afirma que la velocidad de procesamiento en humanos "*es una capacidad cognitiva que consiste en la relación entre la respuesta a una demanda cognitiva y el tiempo invertido en esa operación*."[27], ésta posee cuatro rangos: lentos exactos, rápidos exactos, lentos inexactos y rápidos inexactos, ninguno de ellos relacionado con el coeficiente intelectual. Lo que implica, que la aleatoriedad de esta característica es demasiado alta, y puede afectar la desenvoltura de la persona en cualquier entorno, incluyendo tanto el ámbito del aprendizaje, como el de la enseñanza.

---

[24] GARCIA FERNANDEZ, Luis Alberto. Usos y aplicaciones de la inteligencia artificial. En: La Ciencia y el Hombre. México. 2004. [Consultado el 15 de mayo de 2020]. Disponible en: https://www.uv.mx/cienciahombre/revistae/vol17num3/articulos/inteligencia/

[25] TORO BAYONA, Guillermo Antonio y LIZARAZO SALCEDO, Iván Alberto. Evaluación de las Redes Neuronales Artificiales Perceptrón Multicapa y Fuzzy-Artmap en la Clasificación de Imágenes Satelitales. [2012] En: Ingeniería, Vol.17, No. 1, pág. 61 - 72

[26] GONZALEZ LOPEZ, Francisco Javier. Desarrollo de un sistema de interacción humano-humanoide mediante el reconocimiento y aprendizaje del lenguaje corporal [en línea]. Trabajo de grado Ingeniero Mecatrónico. Santiago de Cali. Universidad Autónoma de Occidente. Facultad de Ingeniería. Departamento de Ingeniería. 2018. 54 - 64 p. [Consultado: 20 de febrero de 2020]. Disponible en: Biblioteca - Recursos Electrónicos. http://hdl.handle.net/10614/10162

[27] FOMINAYA, Carlota. Velocidad de procesamiento ¿Qué es y cómo influye en tu hijo? [En línea] En: ABC Educación. España. 2019. [Consultado el 16 de mayo de 2020]. Disponible en: https://www.abc.es/familia/educacion/abci-velocidad-procesamiento-y-como-influye-inteligencia-hijo-201903250148_noticia.html

Adicionalmente, según Alberich, Gómez y Ferrer[28], sólo la zona central del campo visual es nítida -sabiendo que la percepción no desenfocada de la realidad responde al movimiento del ojo y a la composición que hace el cerebro de la información recibida-, lo que conlleva a que las estimaciones de movimiento que realiza una persona se vean sesgadas por limitantes biológicas, siendo un inconveniente para una práctica de un grupo grande de personas.

Por ello, el aporte propuesto está enfocado hacia las personas que inician o que tienen falencias en el baile con la aspiración de apoyar su mejora continua en la disciplina artística, ya que, según una encuesta realizada por MGM Resorts[29], desde la perspectiva colectiva americana en 2017, el entretenimiento -una de los conceptos de baile para Gardner[30]- se percibía como esencial para la salud y la felicidad en un 93%, mientras el 92% lo percibió como una necesidad básica, reflejando de esta manera, la importancia de realizar proyectos desde este enfoque.

El baile es una forma de expresión y comunicación que puede pasar de una afición a una profesión, dependiendo de la finalidad con la que se practique[31], para esto, existen centros y escuelas de baile especializadas, donde docentes enseñan a personas las técnicas correspondientes a cada tipo de baile, al mismo tiempo que evalúan parámetros tales como el ritmo, el compás y el tempo[32]. La correcta percepción de estas variables está íntimamente relacionada con la ejecución de la coreografía o la danza de manera armónica y precisa, sin embargo, aspectos como

---

[28] ALBERICH, Jordi; GÓMEZ FONTANILLS, David y FERRER FRANQUESA, Alba. Percepción Visual. [En línea] España: Universitat Oberta de Catalunya. 2013. 66 p. [Consultado el 16 de mayo de 2020]. Disponible en: https://www.exabyteinformatica.com/uoc/Disseny_grafic/Diseno_grafico/Diseno_grafico_(Modulo_1).pdf

[29] MGM RESORTS INTERNATIONAL. The Truth About Entertainment [infografía]. Disponible en: https://www.mgmresorts.com/content/dam/MGM/corporate/corporate-initiatives/welcome-to-the-show/mgm-resorts-truth-about-entertainment-infographic.pdf

[30] GARDNER, Howard. La danza. En: *Revista Kinesis* Vol 2 Nº. 6. Bogotá. D. E. 1991. Res. Min. Gobierno 2113/89.

[31] GONZÁLEZ, Mónica. Bailar, ¿una profesión o una afición?. [blog]. Blogs EMagister. [Consultado 17 de mayo de 2020] Disponible en: https://www.emagister.com/blog/bailar-una-profesion-o-una-aficion/

[32] Anónimo. El ritmo, un elemento esencial del baile. About Español. [2019]. [Consultado el 16 de mayo de 2020]. Disponible en: https://www.aboutespanol.com/el-ritmo-un-elemento-esencial-del-baile-297914

la cantidad de alumnos -salones con 20 alumnos promedio en grupos de salsa[33], el tiempo de duración de la clase -mínimo de una hora- además de las habilidades y destrezas de cada persona, complican la tarea de enseñanza y evaluación al docente, ya que no se produce una clase equitativa en donde todos tengan el mismo nivel de aprendizaje.

Con base a lo anteriormente expuesto, se propuso el desarrollo de una RNA que tiene como propósito reconocer características y aspectos durante los distintos bailes en cada alumno, buscando proporcionarle al docente una herramienta que facilita la evaluación crítica e individual en un gran grupo de personas, para así identificar y diseñar metodologías de aprendizaje equitativas más ajustadas a las necesidades individuales.

[33] ÁZUCAR COMPAÑÍA ARTÍSTICA & ESCUELA DE BAILE. [vídeo] Santiago de Cali. Facebook. Azúcar compañía artística & escuela de baile. (29 de marzo de 2020). 0:11 minutos. [Consultado: 17 de mayo de 2020]. Disponible en: https://www.facebook.com/azucarescueladebaile/videos/2533679163403777/

3. OBJETIVOS

3.1 OBJETIVO GENERAL

Diseñar una RNA que permita identificar características en la ejecución del baile de los ritmos populares: Salsa, Bachata, Merengue y Chachacha, para la mejora de los practicantes en los ritmos seleccionados.

3.2 OBJETIVOS ESPECÍFICOS

- Diseñar diferentes tipos de RNAs profundas para tener la identificación de los ritmos seleccionados.

- Entrenar cada una de las RNAs con su respectivo dataset.

- Comparar las diferentes respuestas de las RNAs mediante los índices de tiempo de entrenamiento, asertividad y estructura de las mismas.

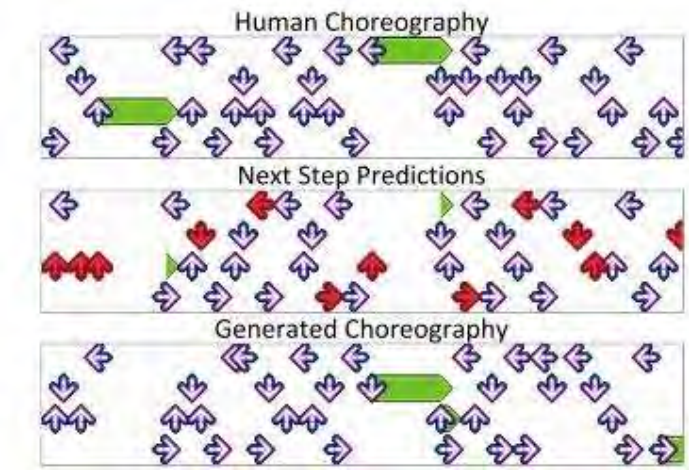
- Validar el funcionamiento de la RNA para la función especificada.

4. ANTECEDENTES

4.1 DANCE DANCE CONVOLUTION

Esta propuesta de Chris Donahue, Zachary Lipton y Julian McAuley se relaciona con el conocido juego Dance Dance Revolution, donde los jugadores realizan una serie de pasos programados de acuerdo al catálogo disponible en el programa. Ellos, proponen la generación de coreografías implementando un análisis auditivo de determinada pista para establecer dos tareas: decidir cuándo situar los pasos y cuáles pasos poner para la canción. La primera labor fue definida combinando redes recurrentes y convolucionales para crear espectrogramas de características de audio de bajo nivel para predecir los pasos, acorde a la dificultad y para la selección de pasos, se presenta un modelo de LSTM condicional.

Figura 1. Gráfica comparativa Dance Dance Convolution



Fuente: DONAHUE, Chris, LIPTON, Zachary y MCAULEY, Julian. Dance Dance Convolution. [imagen] Cornell University . USA. ICML. Conference paper. 2017. p. 7. [Consultado: 10 de marzo de 2021]. Disponible en: https://arxiv.org/pdf/1703.06891.pdf

4.2 EVERYBODY DANCE NOW

Presentado por Caroline Chan, Shiry Ginosar, Tinghui Zhou y Alexei Efros, Everybody Dance Now, utiliza un concepto de transferencia de movimiento, teniendo como fuente un vídeo donde aparece un profesional de cualquier disciplina bailando y el performance es transferido a un amateur por medio del reconocimiento de la imagen corporal. Gracias a la identificación de poses se logra duplicar los movimientos realizados con la imagen de la persona objetivo, siendo un vídeo de pasos básicos del amateur lo que permite replicar la coreografía con una ejecución idéntica a la del profesional.

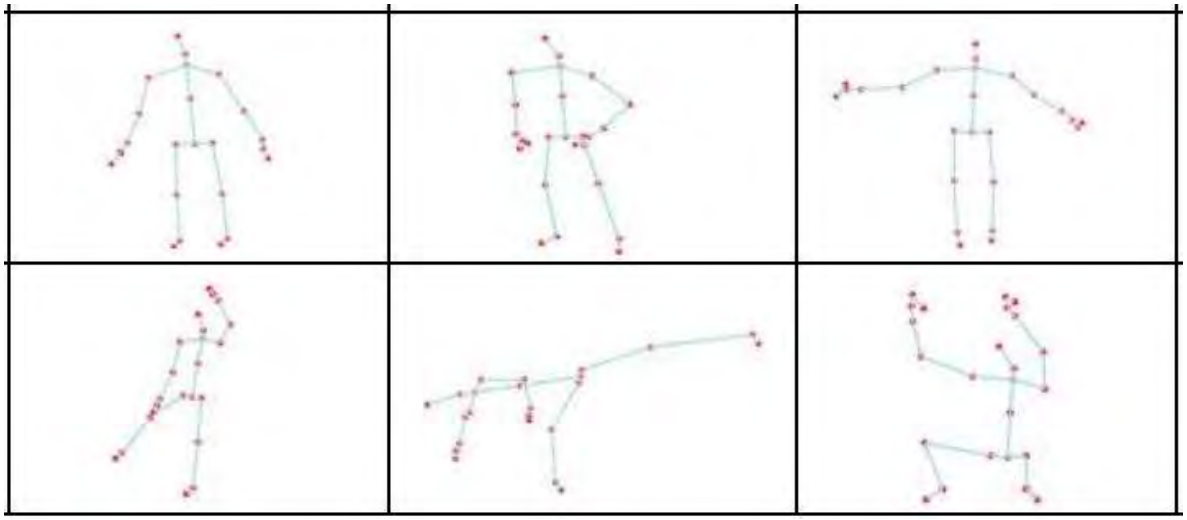
Figura 2. Resultados de Everybody Dance Now

Fuente: CHAN, Caroline, *et al*. “Do as I Do”. [imagen] Everybody Dance Now. USA. ICCV. 2019. p. 1. [Consultado: 10 de marzo de 2021]. Disponible en: https://arxiv.org/pdf/1808.07371.pdf

4.3 GENERATIVE CHOREOGRAPHY USING DEEP LEARNING

Luka Crnkovic y Louise Crnkovic, construyeron una red neuronal recurrente que permite la producción de coreografías con ejecuciones de una cohesión alta y prometedora para un bailarín en solitario. Inicia con un proceso de aprendizaje dependiente del coreógrafo original, a partir del cual, se elabora una coreografía subsecuente que encaja con la inicial y simultáneamente se construye en concordancia con el resto de material que se provea.

Figura 3. Pruebas de Generative Choreography using Deep Learning

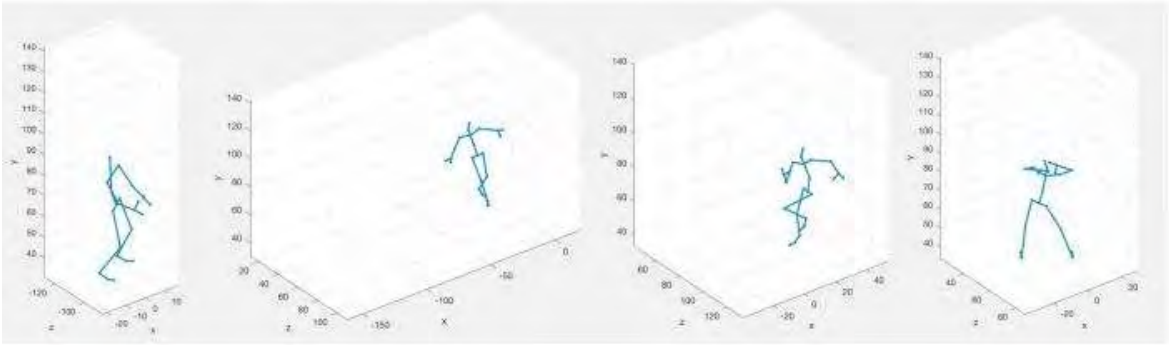


Fuente: CRNKOVIC-FRISS, Luka y CRNKOVIC-FRISS, Louise. "Example results over time". [imagen] Generative Choreography using Deep Learning . USA. 7th International Conference on Computacional Creativity. 2016. p. 4 .[Consultado: 10 de marzo de 2021]. Disponible en: https://arxiv.org/ftp/arxiv/papers/1605/1605.06921.pdf

4.4 GROOVENET

Este sistema que se encuentra en desarrollo, pretende realizar un aprendizaje sintetizado de los movimientos de baile para una pista de audio en tiempo real, haciendo uso de las Factored Conditional Restricted Boltzmann Machines (FCRBM) y redes neuronales recurrentes (RNN), para el caso de las canciones con las que se entrena funciona de buena forma. Por otra parte, se encuentra en mejora el desempeño de la red ante canciones desconocidas.

Figura 4. Groovenet en Matlab



Fuente: ALEMI, Omid, FRANÇOISE, Jules y PASQUIER, Phipippe. “Some still frames from the generated movement patterns”. [imagen] GrooveNet: Real-Time Music Driven Dance Movement Generation using Artificial Neural Networks. Canadá. ML4Creativity. 2017. p. 5. [Consultado: 10 de marzo de 2021]. Disponible en: https://omid.al/docs/groovenet-ml4c-2017.pdf

4.5 WEAKLY – SUPERVISED DEEP RECURRENT NEURAL NETWORKS FOR BASIC DANCE STEP GENERATION

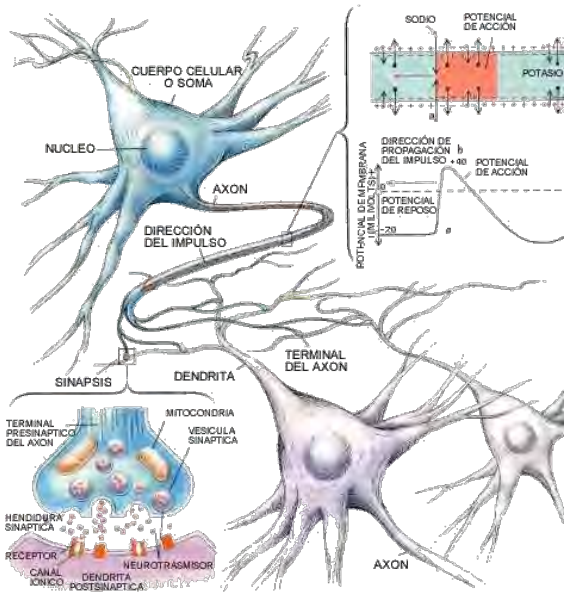
Para esta propuesta, se implementó un modelo que consta de etapa convolucional y modelo de memoria a corto-largo plazo para el procesamiento del audio, y también se utiliza este último para la decodificación del baile. Se obtuvo un modelo que generaba movimientos básicos con un resultado óptimo.

5. MARCO TEÓRICO

5.1 LA NEURONA

Las neuronas son células especializadas en el procesamiento de la información en el cerebro que se pueden comunicar entre ellas con la meta de compartir información; el proceso de contacto entre la neurona que envía la señal (presináptica) y neurona que la recibe (postsináptica) se conoce como sinapsis, y la transferencia de la misma logra que una variación eléctrica de la neurona presináptica afecte el estado de la neurona postsináptica. Este cambio es generado por un pulso eléctrico conocido como potencial de acción, encargado de estimular la neurona postsináptica por medio de las dendritas. El esquema que representa la conexión entre neuronas se muestra en la siguiente imagen.

Figura 5. Estructura de una red neuronal



Fuente: FISCHBACH, Gerald D. Mind and brain. En: Scientific American, [En línea] USA: Scientific American. septiembre, 1992, p. 48–59. [Consultado: 20 de marzo de 2021]. Disponible en: https://www.jstor.org/stable/24939212

Ramírez y Hurtado definen la analogía digital a la neurona como una unidad de cómputo que contiene:

- Un conjunto de canales de entrada o ingreso de señales, caracterizados por un peso sináptico que cuantifica la eficiencia en la transmisión de la señal. Estos canales modelan la ramificación dendrítica de una neurona real.

- Una entrada neta o potencial postsináptico o nivel de activación, calculada generalmente como una suma ponderada de las señales provenientes de los canales de entrada.

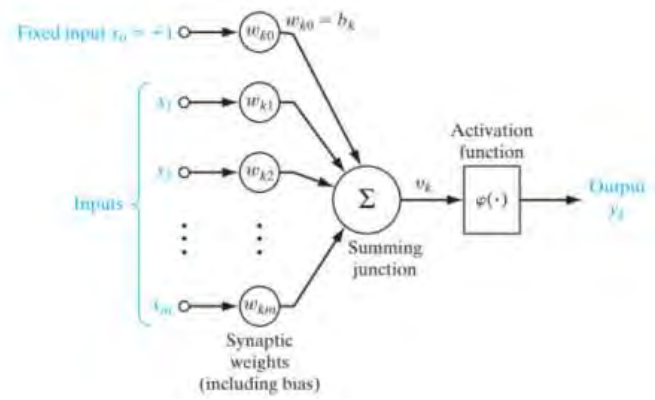
- Una función de activación, de ganancia o de transferencia, que toma la entrada neta y la computa, usualmente de manera no lineal, para producir el estado de salida o de actividad de la neurona. Esta es la modelación compleja de la acción del soma y su membrana sobre la integración de todos los potenciales postsinápticos aferentes.

- Una función de salida de carácter probabilístico, que señala la probabilidad que la neurona modelo transmita o proyecte su estado de salida a las siguientes neuronas postsinápticas. Este es el modo de interpretar la acción del segmento inicial del axón a la computación.[34]

La analogía computacional a la neurona se encuentra representada en la siguiente imagen.

[34] RAMIREZ MORENO, David F. y HURTADO LOPEZ, Julián. Modelamiento y simulación de circuitos sinápticos sensoriomotores: Introducción a la neurobiología computacional. Colombia: Universidad Autónoma de Occidente. 2014. p 29.

Figura 6. Esquema de un Perceptrón Simple



Fuente: HAYKIN, Simon Nonlinear model of a neuron, labeled k. [imagen]. Neural Networks and Learning Machines. 2da Edición. New Jersey: Pearson Education. 1999. p. 11.

5.2 MULTI-LAYER PERCEPTRON (MLP)

Inspirada por el reconocimiento de que el cerebro humano trabaja de forma completamente diferente del computador convencional tanto a nivel de velocidad como en su capacidad computacional para el procesamiento de información compleja, no lineal y paralela, una red neural es una máquina diseñada para modelar la forma en que el cerebro ejecuta una tarea particular o una función de interés. Descrita como un procesador masivo distribuido en paralelo, hecho de unidades de procesamiento simple, tiene una tendencia natural a almacenar conocimiento experimental, hacerlo disponible para su uso y presenta similitud en dos aspectos:

- El conocimiento es adquirido por la red neural de su entorno mediante un proceso de aprendizaje.

- Las fuerzas de la conexión interneuronal, conocida como pesos sinápticos, son usadas para hacer acopio del almacenamiento adquirido.

La primera red neural descrita algorítmicamente, conocida como perceptrón, es la forma más simple de red neuronal usada para clasificar patrones linealmente separables, es decir, patrones que se encuentra en lados opuestos del hiperplano. Básicamente, consiste de una única neurona con pesos sinápticos y bias ajustables,

con un algoritmo de ajuste diseñado por Rosenblatt capaz de demostrar que si los patrones con los que se entrena el perceptrón están dibujados a partir de dos clases linealmente separables, el algoritmo del perceptrón converge y posiciona la superficie de decisión en forma de hiperplano entre las dos clases.

Haykin sostiene que, con el fin de sobreponerse a la limitaciones prácticas debido a la estructura del perceptrón, se desarrolló una red neuronal conocida como Perceptrón Multicapa o MLP por sus siglas en inglés. La MLP contiene en su estructura las siguientes características básicas:

- "El modelo de cada neurona en la red incluye una función de activación no lineal, que es diferenciable.

- La red contiene una o más capas que están ocultas de los nodos de entrada y salida.

- La red exhibe un alto grado de conectividad, que se extiende a ser determinado por los pesos sinápticos de la red"[35].

Además de estas características, existe un algoritmo popular para el entrenamiento de MLP conocido como backpropagation, el cual, consta de dos fases:

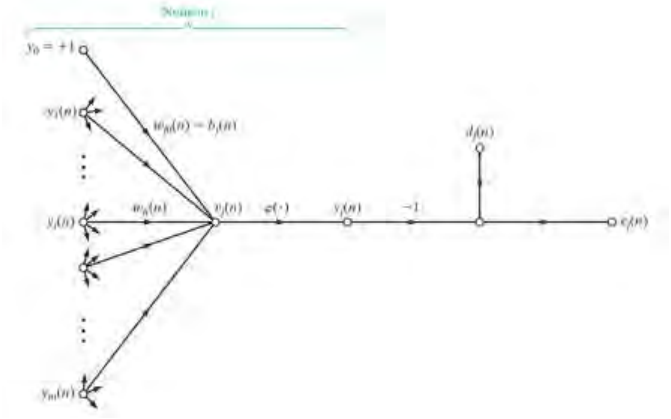
Haykin[36] explica que en la forward o feedforward phase, los pesos sinápticos de la red son fijados y la señal de entrada es propagada a través de la red, capa a capa, hasta la salida, los cambios dependen de los potenciales de activación y las salidas de las neuronas. La siguiente ecuación representa el concepto donde l inicia en 0, hasta la capa L.

$$v_j^l = \sum_{i=0}^{I} \omega_{ji}^l y_i^{(l-1)} = \boldsymbol{\omega}_{ji}^l \boldsymbol{y}_i^{(l-1)} \qquad [1]$$

[35] HAYKIN, Simon. Neural Networks and Learning Machines. 2da Edición. New Jersey: Pearson Education. 2009. p. 123.

[36] Ibíd. p. 123 - 124

Figura 7. Esquema de Feedforward de una MLP



Fuente: HAYKIN, Simon. Signal-flow graph highlighting the details of output neuron j. [imagen]. Neural Networks and Learning Machines. 2da Edición. New Jersey: Pearson Education. 1999. p. 129.

La iteración actual es referenciada en la gráfica con la letra n

$$v_j = \sum_{i=0}^{m} \omega_{ji} y_i \qquad [2]$$

$$y_j = \phi_j(v_j) \qquad [3]$$

En la backward phase, una señal de error es producida comparando la salida con la respuesta deseada; la señal de error resultante es propagada nuevamente pero en sentido inverso, para ello, se hace uso de la función del error cuadrático medio:

$$e_j = d_j - y_j \qquad [4]$$

$$L = \frac{1}{2}\sum_{n=1}^{N} e_j^2 = \frac{1}{2}\sum_{n=1}^{N}(d_j - y_j)^2 \qquad [5]$$

Para la minimización de la función anterior se varían los pesos sinápticos, hecho similar al uso de la regla delta en el perceptrón, se actualiza ω_{ji} con $\Delta\omega_{ji}$, que es proporcional a la $\frac{\partial L}{\partial \omega_{ji}}$, de acuerdo con la regla de la cadena:

$$\frac{\partial L}{\partial \omega_{ji}} = \frac{\partial L}{\partial e_j}\frac{\partial e_j}{\partial y_j}\frac{\partial y_j}{\partial v_j}\frac{\partial v_j}{\partial \omega_{ji}} \qquad [6]$$

$$\frac{\partial L}{\partial e_j} = e_j \quad \frac{\partial e_j}{\partial y_j} = -1 \quad \frac{\partial y_j}{\partial v_j} = \phi_j'(v_j) \quad \frac{\partial v_j}{\partial \omega_{ji}} = y_i \qquad [7]$$

$$\frac{\partial L}{\partial \omega_{ji}} = -e_j \phi_j'(v_j) y_i \qquad [8]$$

Aplicado $\Delta\omega_{ji}$ a ω_{ji} se define por:

$$\Delta\omega_{ji} = -\eta \frac{\partial L}{\partial \omega_{ji}} \qquad [9]$$

$$\Delta\omega_{ji} = \eta e_j \phi_j'(v_j) y_i \qquad [10]$$

Donde el error o error local de la capa, es lo que se denomina el error acreditado. Este va desde la neurona j y se propaga hasta la salida de la red.

$$\delta_j = -e_j \phi_j'(v_j) \qquad [11]$$

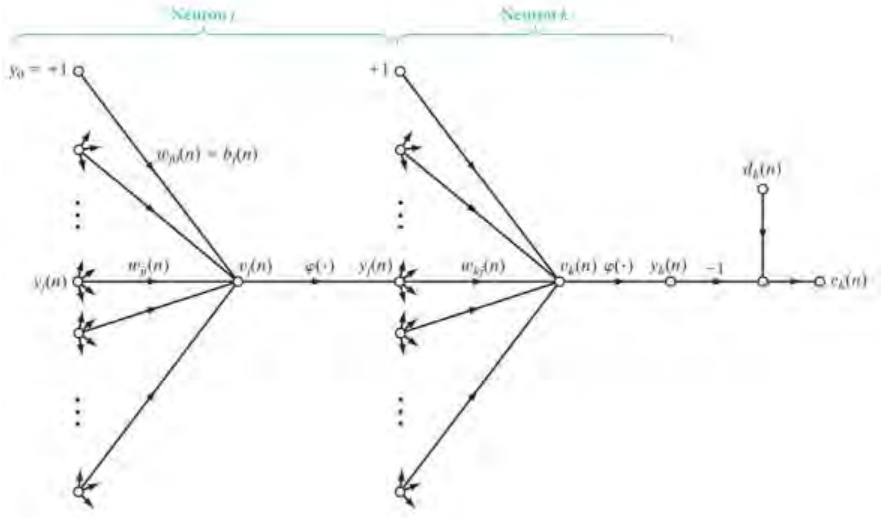
$$\Delta\omega_{ji} = \eta \delta_j y_i \qquad [12]$$

Para tener presente, se expande el término δ_j:

$$\delta_j = \frac{\partial L}{\partial v_j} = \frac{\partial L}{\partial e_j}\frac{\partial e_j}{\partial y_j}\frac{\partial y_j}{\partial v_j} = \frac{\partial L}{\partial y_j}\frac{\partial y_j}{\partial v_j} \qquad [13]$$

Haykin[37] hace un énfasis en apreciar la importancia de e_j en la actualización de los pesos, ya que es posible de adquirir directamente de la comparación entre d_j y y_j en la capa de salida, sin embargo, cuando se tiene un error en una capa oculta surge la duda: ¿con qué se compara la salida de la neurona? ¿Cómo se atribuye parte del error obtenido a la salida de la red?. Esto se logra, recurriendo a la adición de una capa oculta para poder deducir el término para la actualización de los pesos, permitiendo un ajuste sucesivo y en dirección inversa al término de señal de error, aplicable a todas las capas y neuronas.

Figura 8. Estructura de una MLP



Fuente: HAYKIN, Simon. Signal-flow graph highlighting the details of output neuron k connected to hidden neuron j [imagen]. Neural Networks and Learning Machines. 2da Edición. New Jersey: Pearson Education. 1999. p. 132.

$$L = \frac{1}{2}\sum_{k=1}^{C_k} e_k^2 \qquad [\ 14\ ]$$

$$e_k = d_k - y_k \qquad [\ 15\ ]$$

[37] Ibíd. p. 131

$$y_k = \phi_k(v_k) \quad [16]$$

$$v_k = \sum_{j=0}^{m} \omega_{kj} y_j \quad [17]$$

$$\frac{\partial L}{\partial \omega_{kj}} = \frac{\partial L}{\partial e_k}\frac{\partial e_k}{\partial y_k}\frac{\partial y_k}{\partial v_k}\frac{\partial v_k}{\partial \omega_{kj}} \quad [18]$$

Para la actualización de los pesos ω_{kj}:

$$\Delta\omega_{kj} = -\eta \frac{\partial L}{\partial \omega_{kj}} \quad [19]$$

$$\frac{\partial L}{\partial \omega_{kj}} = -e_k \phi_k'(v_k) y_j = -\delta_k y_j \quad [20]$$

$$\delta_k = e_k \phi_k'(v_k) \quad [21]$$

$$\frac{\partial L}{\partial \omega_{kj}} = -\delta_k y_j \quad [22]$$

$$\Delta\omega_{kj} = \eta \delta_k y_j \quad [23]$$

La necesidad es actualizar pesos de las capas ocultas

$$\frac{\partial L}{\partial \omega_{ji}} = \frac{\partial L}{\partial e_k}\frac{\partial e_k}{\partial y_k}\frac{\partial y_k}{\partial v_k}\frac{\partial v_k}{\partial \omega_{ji}} = -\delta_k \frac{\partial v_k}{\partial \omega_{ji}} \quad [24]$$

$$\frac{\partial L}{\partial \omega_{ji}} = -\delta_k \frac{\partial v_k}{\partial y_j}\frac{\partial y_j}{\partial \omega_{ji}} = -\delta_k \frac{\partial v_k}{\partial y_j}\frac{\partial y_j}{\partial v_j}\frac{\partial v_j}{\partial \omega_{ji}} \quad [25]$$

$$\frac{\partial v_k}{\partial y_j} = \omega_{kj} \quad [26]$$

Se excluye el parámetro del bias, que es el término ω_{k0}, bias de la neurona k, debido a que y_0 en j es 1.

$$\frac{\partial y_j}{\partial v_j} = \phi_j'(v_j) \quad [27]$$

$$\frac{\partial y_j}{\partial \omega_{ji}} = y_i \quad [28]$$

$$\frac{\partial L}{\partial \omega_{ji}} = -\delta_k \omega_{kj} \phi_j'(v_j) y_i \quad [29]$$

Si se compara con:

$$\frac{\partial L}{\partial \omega_{kj}} = -e_k \phi_k'(v_k) y_j \quad [30]$$

Se evidencia que:

$$e_j = \delta_k \omega_{kj} \quad [31]$$

$$\frac{\partial L}{\partial \omega_{ji}} = -e_j \phi_j'(v_j) y_i \quad [32]$$

$$\delta_j = e_j \phi_j'(v_j) \quad [33]$$

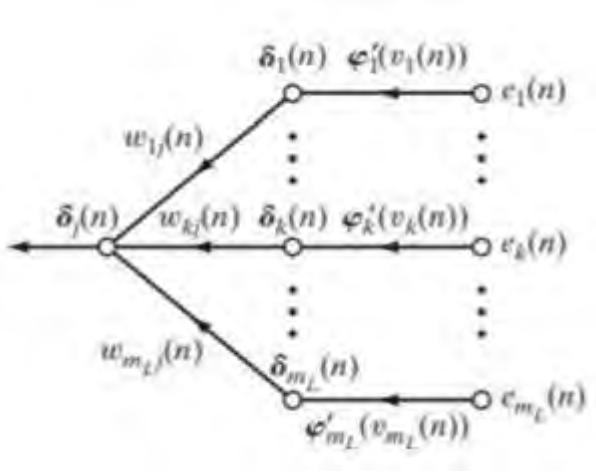
$$\frac{\partial L}{\partial \omega_{kj}} = -\delta_k y_j \quad [34]$$

La actualización de los pesos ω_{ji}

$$\Delta\omega_{ji} = \eta\delta_j y_i \qquad [\ 35\ ]$$

El flujo de la señal de error desde una neurona k la capa K a una neurona j ubicada en la capa J

Figura 9. Esquema de backpropagation de una MLP



Fuente: HAYKIN, Simon. Signal-flow graph of a part of the adjoint system pertaining to backpropagation of error signals [imagen]. Neural Networks and Learning Machines. 2da Edición. New Jersey: Pearson Education. 1999. p. 134.

5.3 CONVOLUTIONAL NEURAL NETWORK (CNN)

En su libro, Haykin describe una red convolucional como:

> Un perceptrón multicapa diseñado específicamente para reconocer formas bidimensionales con alto grado de invarianza en el traslado, escalamiento, sesgado y otras formas de distorsión. Esta difícil tarea es aprendida de manera supervisada por medio de una red cuya estructura incluye las siguientes limitaciones:
>
> - Extracción de características: Cada neurona toma sus entradas sinápticas de campo local receptivo en la capa anterior, forzando de este modo las características locales. Una vez la característica ha sido extraída, su localización exacta se vuelve menos significativa, siempre que su posición relativa a otras características sea aproximadamente preservada.

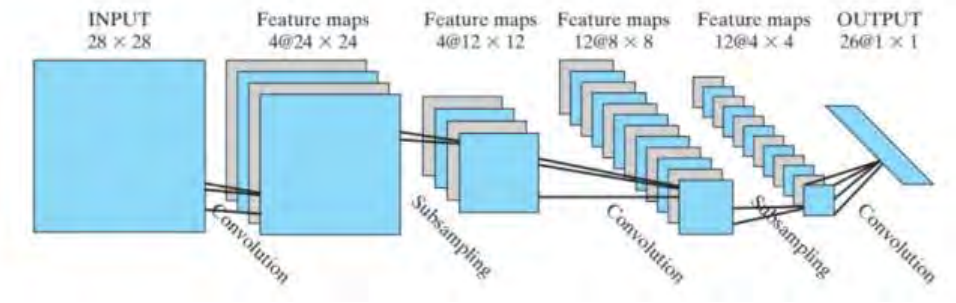
- Mapeado de características: Cada capa computacional de la red está compuesta por múltiples mapas de características, con cada mapa de característica estando en la forma de un plano dentro del cual las neuronas individuales son limitadas a compartir el mismo set de pesos sinápticos.

- Muestreo parcial: Cada capa convolucional es seguida por una capa computacional que realiza un promedio local y un muestreo parcial, mediante el cual el mapa de características es reducido. Esta operación tiene el efecto de reducir la sensibilidad de la salida del mapa de características a desplazamientos u otras formas de distorsión.

Se enfatiza que todos los pesos en todas las capas de una red convolucional se aprenden a través del entrenamiento, además, la red aprende a extraer sus propias características automáticamente.[38]

La representación gráfica del procesamiento de una red convolucional se presenta en la siguiente imagen.

Figura 10. Estructura de una red convolucional



Fuente: HAYKIN, Simon. Convolutional network for image processing such as handwriting recognition [imagen]. Neural Networks and Learning Machines. 2da Edición. New Jersey: Pearson Education. 1999. p. 202.

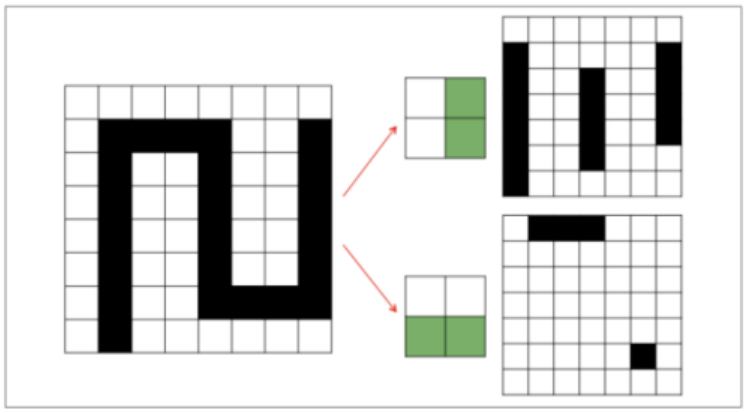
Esta red está diseñada para el procesamiento de imágenes e implementando la alternancia sucesiva entre convolución y muestreo parcial en las capas, se obtiene un efecto “bipiramidal”, lo que significa que por cada capa convolucional o sub mapeada el número de mapas de características incrementa, mientras la resolución espacial disminuye, comparada con la capa anterior correspondiente. La idea de la

[38] Ibíd. p. 201

convolución seguida del mapeo parcial está inspirada por la noción de células “simples”, seguidas por células “complejas”.

Adentrándose en el proceso, se usan filtros que determinan un parámetro específico de la imagen: si hay coincidencia, se refleja en el mapa de características, en caso contrario, no. Esto genera la identificación de particularidades para diseñar los mapas, dicha operación se conoce como convolución y es la encargada de representar combinaciones de conexiones que son replicadas a través de toda la entrada, conllevando a la activación de las neuronas en el mapa de características, si el filtro generado concuerda apropiadamente en la posición correspondiente que se evalúa. En la siguiente imagen se aprecia como se extraen dos características utilizando de los filtros aplicados al ejemplo.

Figura 11. Esquema de backpropagation de una MLP



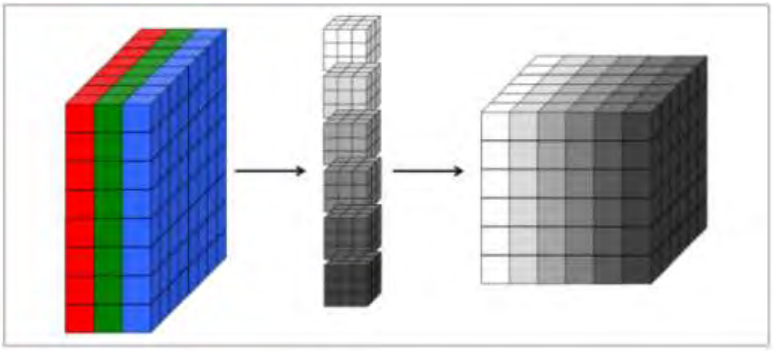
Fuente: BUDUMA, Nikhil. Applying filters that detect vertical and horizontal lines on our toy example [imagen]. Fundamentals of Deep Learning. Sebastopol: O’Reilly. 2017. p. 92.

Para denotar un mapa de características *k* en una capa *m* se denomina m^k y su filtro correspondientes por los valores de sus pesos *W*, asumiendo que las neuronas en el mapa de características tengan un bias b^k -que se mantiene para todas las neuronas del mapa de características- puede ser expresado matemáticamente de la siguiente forma:

$$m_{ij}^{k} = f((W * x)_{ij} + b^{k}) \qquad [\ 36\ ]$$

Tal descripción matemática es simple pero no describe completamente los filtros, ya que estos no operan en un único mapa de características, sino como un volumen de mapas generados por una capa en particular, como resultado, los mapas de características deben ser capaces de operar en volumen y no en un área. Una capa convolucional -que consiste de un set de filtros- convierte un volumen de valores en otro, mientras la profundidad del filtro corresponde a la de entrada, así el filtro combina la información de las características que aprendió.

Figura 12. Resultado de una CNN luego de la aplicación de diversos filtros



Fuente: BUDUMA, Nikhil. A three-dimensional visualization of a convolutional layer, where each filter corresponds to a slice in the resulting output volume [imagen]. Fundamentals of Deep Learning. Sebastopol: O'Reilly. 2017. p. 92.

Profundizando en el área matemática, Zhou[39] describe que la CNN consiste en capas convolucionales caracterizadas por una mapa de entrada ***I***, un banco de filtros **K** -de dimensiones k_1 x k_2 y unos bias ***b***, donde el forward pass, se encuentra definido por una entrada consistente de N puntos, cada uno con C canales, H altura y W ancho, se realiza una convolución a cada entrada con cada filtro, obteniendo la siguiente ecuación.

$$(I * K)_{ij} = \sum_{m=0}^{k_1-1} \sum_{n=0}^{k_2-1} \sum_{c=1}^{C} K_{m,n,c} \cdot I_{i+m,j+n,c} + b \qquad [\ 37\ ]$$

[39] ZHOU, Victor. CNNs, Part 2: Training a Convolutional Neural Network. [En línea] Victorzhou.com. 29 de mayo de 2019. [Consultado: 4 de mayo de 2021] Disponible en: https://victorzhou.com/blog/intro-to-cnns-part-2/

Aplicada la ecuación, se obtienen las características del uso de la convolución, se implementan para la clasificación en la capa de subsampling donde, por medio del pooling -average (promedio) o max (máximo)- de acuerdo a la necesidad presentada, se extraen los datos más representativos para reducir la dimensionalidad de los mapas; es decir, condensar la información del mapa de características, mientras se reduce la sensibilidad de la salida a cambios y distorsiones.

Luego de producirse la salida, se realiza una evaluación para un total de predicciones P, entre los datos calculados $\boldsymbol{y_p}$ y el producto objetivo $\boldsymbol{t_p}$. El error cuadrático medio aplicado se encuentra dado por la ecuación.

$$E = \frac{1}{2}\sum_{p}(t_p - \ y_p)^2 \qquad [\ 38\ ]$$

Durante la forward pase, cada capa deja un caché (como entradas, valores intermedios, entre otros) que será necesario para el backward y, en esta última fase, cada capa recibirá un gradiente y también devolverá un gradiente. Recibirá un gradiente de pérdida respecto a las salidas $\frac{\partial L}{\partial out}$ y otro respecto a la entrada. $\frac{\partial L}{\partial in}$.

Por ello, para el backpropagation de la red, se inicia por pérdida cross - entropy, que se define como:

$$L = \ -\ln(p_c) \qquad [\ 39\ ]$$

Donde p_c es la probabilidad estimada para la clase correcta, en otras palabras, el dígito en el cual está la imagen actualmente. Lo primero que se necesita calcular es la entrada a la capa Softmax del backward, □□□□□□□,donde out_s es la salida de la capa Softmax, lo que se traduce como la evaluación de la ecuación de pérdida así:

$$\frac{\partial L}{\partial out_s(i)} = \begin{cases} 0 & si\ i\ \neq c \\ -\dfrac{1}{p_i} & si\ i = c \end{cases} \qquad [\ 40\ ]$$

Donde c es la clase correcta. De la etapa ejecutada, hay 3 datos caché resultantes, que se implementan posteriormente y son: las dimensiones de la entrada antes de redimensionarla (a un vector 1xn), la entrada después de redimensionarla y los

totales, que son los valores evaluados en la función de activación softmax. Basados en ello, se procede a derivar el gradiente de $out_s(c)$ con respecto a los totales (los valores evaluados en la función de activación softmax), ahora, si t_i sea el total para la clase *i* y $S = \sum_i e^{t_i}$, así se puede escribir $out_s(c)$ como:

$$out_s(c) = \frac{e^{t_c}}{\sum_i e^{t_i}} = \frac{e^{t_c}}{S} \quad [41]$$

Ahora, asumiendo alguna clase *k* como $k \neq c$. Se puede reescribir $out_s(c)$ como:

$$out_s(c) = e^{t_c} S^{-1} \quad [42]$$

Y usar la regla de la cadena para derivar:

$$\frac{\partial out_s(c)}{\partial t_k} = \frac{\partial out_s(c)}{\partial S}\left(\frac{\partial S}{\partial t_k}\right) \quad [43]$$

$$= e^{t_c} S^{-2}\left(\frac{\partial S}{\partial t_k}\right) \quad [44]$$

$$= e^{t_c} S^{-2}(e^{t_k}) \quad [45]$$

$$= -\frac{e^{t_c} e^{t_k}}{S^2} \quad [46]$$

Eso fue asumiendo que $k \neq c$, para hacer la derivación de *c*, esta vez se usa la regla del cociente (porque se tiene un e^{t_c} en el numerador de $out_s(c)$):

$$\frac{\partial out_s(c)}{\partial t_k} = \frac{Se^{t_c} - e^{t_c}\frac{\partial S}{\partial t_c}}{S^2} \quad [47]$$

$$= \frac{Se^{t_c} - e^{t_c} e^{t_c}}{S^2} \quad [48]$$

$$= \frac{e^{t_c}(S - e^{t_c})}{S^2} \quad [49]$$

Ahora, se busca *c* para un gradiente no cero, posteriormente, se calcula el gradiente $\frac{\partial out_s(i)}{\partial t}$ usando las ecuaciones derivadas antes, de la siguiente forma:

$$\frac{\partial out_s(k)}{\partial t} = \begin{cases} -\frac{e^{t_c}e^{t_k}}{S^2} & si\ k \neq c \\ \frac{e^{t_c}(S - e^{t_c})}{S^2} & si\ k = c \end{cases} \quad [50]$$

Para obtener los gradientes de pérdida de los pesos, bias y entradas:

- Se usa el gradiente de pesos, $\frac{\partial L}{\partial \omega}$, para actualizar los pesos de las capas.

- Se usa el gradiente de biases, $\frac{\partial L}{\partial b}$, para actualizar los biases de las capas.

- Y se retorna el gradiente de entrada, $\frac{\partial L}{\partial input}$, del método de backpropagation así se puede usar en la próxima capa.

Para calcular esos tres gradientes de pérdida, primero se necesita derivar tres resultados más: el gradiente de totales contra los pesos, biases y la entrada. La ecuación relevante es:

$$t = \omega * input + b \quad [51]$$

$$\frac{\delta t}{\delta \omega} = input \quad [52]$$

$$\frac{\partial t}{\partial input} = \omega \quad [53]$$

Reemplazando:

$$\frac{\delta L}{\delta \omega} = \frac{\delta L}{\delta out} * \frac{\delta out}{\delta t} * \frac{\delta t}{\delta \omega} \quad [54]$$

$$\frac{\delta L}{\partial b} = \frac{\partial L}{\partial out} * \frac{\partial out}{\partial t} * \frac{\partial t}{\partial b} \quad [55]$$

$$\frac{\partial L}{\partial input} = \frac{\partial L}{\partial out} * \frac{\partial out}{\partial t} * \frac{\partial t}{\partial input} \quad [56]$$

Se procede al entrenamiento de la capa softmax, usando el gradiente descendiente descrito anterior y se regresan los valores de la entrada redimensionados, que tienen que volver a su tamaño original. Para la parte del pooling, se debe tener en cuenta que, a pesar de no ser entrenado, se debe pasar por esta capa para el backpropagation, por lo tanto, se sitúa el valor de cada gradiente donde estaba el valor máximo original de acuerdo a la dimensionalidad inicial. Es decir, si el valor máximo de bloque 2x2 se encontraba en la posición [1,2] dicha posición será la ocupada por el gradiente, ya que anteriormente se guardó este dato; el resto de datos son igualados a cero, ya que un valor no máximo no tendrá efecto marginal en la pérdida. En otras palabras, $\frac{\partial L}{\partial input} = 0$ para píxeles no máximos, mientras un píxel cuyo valor es máximo habrá pasado por la salida, por lo tanto, $\frac{\partial output}{\partial input} = 1$, lo que significa $\frac{\partial L}{\partial input} = \frac{\partial L}{\partial output}$.

Finalmente, en la etapa final del backpropagation, se necesita la actualización de los pesos de los filtros y teniendo $\frac{\partial L}{\partial out}$, hace falta realizar el calculo de $\frac{\partial out}{\partial filters}$, teniendo en cuenta que cualquier cambio en el peso de los filtros afectará toda la imagen de salida de ese filtro, para realizar la derivada de un píxel específico de salida respecto al peso del filtro específico, es igual al valor del píxel correspondiente en la imagen, matemáticamente hablando, la ecuación se describe así:

$$out(i,j) = convolver(image, filter) \quad [57]$$

$$= \sum_{x=0}^{3} \sum_{y=0}^{3} image(i+x, j+y) * filter(x,y) \quad [58]$$

$$\frac{\partial out\ (i,j)}{\partial filter\ (x,y)} = image(i+x, j+y) \qquad [\ 59\ ]$$

Situándolo en conjunto para hallar el gradiente de pérdida para el peso específico del filtro:

$$\frac{\partial L}{\partial filter\ (x,y)} = \sum_i \sum_j \frac{\partial L}{\partial out(i,j)} * \frac{\partial out\ (i,j)}{\partial filter\ (x,y)} \qquad [\ 60\ ]$$

5.4 MATRIZ DE CONFUSIÓN

La matriz de confusión es una herramienta de evaluación de rendimiento que permite visualizar el desempeño de un algoritmo de aprendizaje supervisado[40] [41]; cada columna de la matriz representa el número de predicciones por clase, mientras cada fila representa el número de instancias en la clase real. Gracias a este diseño es posible ver qué tipo de acierto y errores tiene un modelos en el proceso de aprendizaje.

Ahora la clasificación de valores se representa como:

- TP (True Positive o Verdadero Positivo) = Son los valores que el algoritmo clasifica como positivos y son positivos.

- TN (True Negative o Verdadero Negativo) = Son los valores que el algoritmo clasifica como negativos y son negativos.

- FP (False Positive o Falso Positivo) = Son los valores que el algoritmo clasifica como positivos y son negativos. Se conoce como error tipo 1.

[40] BARRIOS ARCE, Juan I. La matriz de confusión y sus métricas. [En línea] Health Big Data. 2019. [Consultado: 12 de agosto de 2021] Disponible en: https://www.juanbarrios.com/la-matriz-de-confusion-y-sus-metricas/

[41] ACEVEDO, Natalia. Matriz de confusión en Machine Learning. Explicado paso a paso. [En línea] Nataliaacevedo.com. 2020. [Consultado: 12 de agosto de 2021] Disponible en: https://nataliaacevedo.com/matriz-de-confusion-en-machine-learning-explicado-paso-a-paso/

- FN (False Negative o Falso Negativo) = Son los valores que el algoritmo clasifica como negativos y son positivos. Se conoce como error tipo 2.

A partir de estos valores se construyen unas métricas de clasificación que permiten valorar los resultados de la red y son:

- Exactitud o Accuracy: Permite calcular de todas las clases evaluadas, cuántas se predijeron correctamente. Es una métrica que se recomienda usar cuando los valores están balanceados, es decir, la misma cantidad de valores por etiqueta. Se expresa en la siguiente ecuación:

$$\frac{TP + TN}{TP + TN + FP + FN} \quad [61]$$

- Precisión: Evalúa qué porcentaje de valores positivos fueron predichos correctamente, se genera este tipo de parámetro por cada clase. La ecuación que representa esta métrica es la siguiente:

$$\frac{TP}{TP + FP} \quad [62]$$

- Sensibilidad o Recall: Indica la proporción de casos positivos que fueron correctamente clasificados. La ecuación que define esta métrica es:

$$\frac{TP}{TP + FN} \quad [63]$$

- F1 Score o Medida F: Es una métrica utilizada en problemas en los que el conjunto de datos a analizar está desbalanceado, utiliza la media armónica para castigar los valores extremos. La métrica se encuentra descrita así:

$$\frac{2 * recall * precision}{recall + precision} \quad [64]$$

5.5 POSE ESTIMATION

Es una técnica de visión por computadora utilizada para detectar la posición física de una figura humana en imágenes y vídeos, a través de un modelo ML, mediante la estimación de las ubicaciones espaciales de las 18 articulaciones claves del cuerpo (Puntos clave). Cabe aclarar, que dicha imagen estima la localización de las articulaciones, sin embargo, no reconoce quién está en la imagen o vídeo.

Figura 13. Imagen comparativa. Esquema en imagen vs. Esquema en solitario

Fuente: ROVAI, MARCELO. [imagen] Realtime Multiple Person 2d Pose Estimation using TensorFlow2.x. USA. ICCV. 2020. [Consultado: 28 de octubre de 2021]. Disponible en: https://towardsdatascience.com/realtime-multiple-person-2d-pose-estimation-using-tensorflow2-x-93e4c156d45f

El modelo consiste en una backend AlexNet de 7 capas con una capa final adicional que genera 2000 coordenadas conjuntas. El problema más significativo es que primero, se debe detectar a una persona antes de aplicar el modelo, en otras palabras, primero se identifica el cuerpo y posteriormente las articulaciones asociadas. Según Cao, *et al*, algunos de los inconvenientes son:

- Cada imagen contiene un número desconocido de individuos que pueden aparecer en cualquier posición o escala.

- Las interacciones entre personas inducen a interferencias espaciales complejas, debido al contacto, oclusión y las articulaciones de las extremidades, dificultando la asociación de partes.
- La complejidad de los tiempos de ejecución tiende a aumentar de acuerdo al número de personas en la imagen, convirtiendo el rendimiento en tiempo real en un desafío. [42]

Los modelos de estimación de pose, toman una imagen de cámara procesada como entrada y emiten información sobre los puntos clave, cada uno tiene un ID y una puntuación de confianza entre 0.0 y 1.0. Esta puntuación indica la probabilidad de que exista un punto clave en esta posición.

Normalmente, se emplea un detector de personas y se reconoce únicamente una persona por detección, el cual ha sido ampliamente aprovechado para las técnicas de estimación existente, pero surge un problema: si el detector falla -algo que ocurre cuando las personas se encuentran demasiado cerca- no existe forma de recuperar los recursos. Así mismo, el tiempo de ejecución es directamente proporcional al número de personas que estén en la imagen.

5.6 MATLAB

Matrix Laboratory, mejor conocido por sus siglas de MATLAB, es una plataforma de programación y cálculo numérico utilizada por ingenieros y científicos para analizar datos, desarrollar algoritmos y crear modelos. El lenguaje MATLAB, se basa en matrices para facilitar la expresión de matemáticas y arreglos computacionales, en un entorno de escritorio que permite su análisis interactivo.

Algunas de las herramientas que se pueden llevar a cabo en el software son gráficas, análisis de datos, formulación de algoritmos, creación de apps, cálculo paralelo, cálculo en la nube, despliegue en escritorio y web, conexión con hardware, entre otros.

Respecto al área de la inteligencia artificial, MATLAB ha generado herramientas para su implementación, para casos de preprocesamiento, diseño y modelamiento

[42] CAO, ZHE, et al. Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. En: CVPR. [En línea] USA. 2017. [Consultado: 25 de octubre de 2021]. Disponible en: https://arxiv.org/pdf/1611.08050.pdf

de Machine Learning y AI, Deep Learning, Reinforcement Learning, Procesamiento de lenguaje natural, entre otros.

5.7 PYTHON

Se define como un lenguaje de programación interpretado, orientado a objetos y de alto nivel con semántica dinámica. Sus estructuras de datos integradas de alto nivel, combinadas con la escritura dinámica y el enlace dinámico, lo hacen muy atractivo para el desarrollo rápido de aplicaciones, así como para su uso como lenguaje de scripting o pegamento para conectar componentes existentes. La sintaxis simple y sencilla de Python enfatiza la legibilidad y, por lo tanto, reduce el costo de mantenimiento del programa. Python admite módulos y paquetes, lo que fomenta la modularidad del programa y la reutilización del código. El intérprete de Python y la extensa biblioteca estándar, están disponibles en forma de fuente o binaria sin cargo para todas las plataformas principales, y se pueden distribuir libremente.

5.7.1 Keras

Keras es una interfaz de programación de aplicaciones de aprendizaje profundo escrita en Python, la cual, se ejecuta sobre TensorFlow. Según la página de Keras[43], algunas de sus características son:

- La reducción de la carga cognitiva del desarrollador para liberarlo para que se concentre en las partes del problema que realmente importan.

- Aplicación del principio de divulgación progresiva de la complejidad del trabajo: los flujos simples son rápidos y fáciles, mientras los flujos arbitrariamente avanzados, son usados por un camino claro que se basa en lo aprendido.

- Proporción de un rendimiento y escalabilidad sólidos en la industria.

También se define como una interfaz accesible y altamente productiva para resolver dificultades de aprendizaje automático, con una orientación en el aprendizaje profundo moderno. Facilita abstracciones esenciales y bloques de construcción para elaborar y enviar soluciones de aprendizaje automático con alta velocidad de iteración.

[43] KERAS. About Keras. [En línea] Keras.io. [Consultado: 28 de octubre de 2021]. Disponible en: https://keras.io/about/

5.7.2 TensorFlow

Es una plataforma de aprendizaje automático de código abierto de extremo a extremo. Según su página oficial, se combinan cuatro habilidades:

"- Ejecución eficiente de operaciones tensoriales de bajo nivel en CPU, GPU o TPU.

- Cálculo del gradiente de expresiones diferenciables arbitrarias.

- Escalado de cálculo a muchos dispositivos, como clústeres de cientos de GPU.

- Exportación de programas ("gráficos") a tiempos de ejecución externos como servidores, navegadores, dispositivos móviles e integrados."[44]

5.7.3 Librería CV2 u Open CV

Es una librería de código abierto que incluye miles de algoritmos de visión computacional. Compuesta de forma modular, algunos de los módulos disponibles son: Funcionalidad central, procesamiento de imágenes, análisis de vídeo, calibración de cámara y reconstrucción 3D, marco de características 2d, detección de objetos, entre otros.

5.7.4 Librería NumPy

En palabras de sus desarrolladores, "Numpy es el paquete fundamental para la computación científica en Python, provee objetos de matriz multidimensional, varios objetos derivados y una variedad de rutinas para operaciones rápidas en matrices, incluyendo manipulación matemática, lógica de formas, selección, transformadas discretas de Fourier, álgebra lineal básica, operaciones estadísticas básicas, entre otros".[45]

[44] Ibíd.

[45] NUMPY. What is NumPy?. [En línea] The NumPy community. 22 de junio de 2021. [Consultado: 28 de octubre de 2021] Disponible en: https://numpy.org/doc/stable/user/whatisnumpy.html

Igualmente, afirman que es necesario el conocimiento de matrices en Numpympar poder usar de manera eficiente gran parte del software científico - matemático basado en Python, debido a la alta implementación de esta herramienta.

5.7.5 Librería OS

Este módulo proporciona una forma portátil de utilizar la funcionalidad dependiente del sistema operativo. Posee múltiples funcionalidades como la lectura o escritura de archivos, manipulación de rutas, lectura de líneas en todos los archivos en la línea de comando, creación de archivos y directorios temporales, manejo de archivos y directorios de alto nivel, entre otros.

5.7.6 Librería Pandas

Patel nos dice que:

> El procesamiento de datos es una parte importante del análisis de los datos, porque los datos no siempre están disponibles en el formato deseado. Se requieren varios procesos antes de analizar los datos, como limpieza, reestructuración o fusión, etc. Numpy, Scipy, Cython y Panda son las herramientas disponibles en Python que se pueden utilizar para procesar rápidamente los datos. Además, los pandas se construyen en la cima de Numpy.
>
> Pandas proporciona un amplio conjunto de funciones para procesar varios tipos de datos. Además, trabajar con Panda es rápido, fácil y más expresivo que otras herramientas. Pandas proporciona un procesamiento de datos rápido como Numpy junto con técnicas de manipulación de datos flexibles como hojas de cálculo y bases de datos relacionales. Por último, pandas se integra bien con la biblioteca matplotlib, lo que la convierte en una herramienta muy útil para analizar los datos.[46]

[46] PATEL, Meher Krishna. 1. Pandas Basic. [En línea] pandasguide.readthedocs.io [Consultado: 28 de octubre de 2021]. Disponible en: https://pandasguide.readthedocs.io/en/latest/Pandas/basic.html

5.7.7 Librería Matplotlib

El equipo de desarrollo de Matplotlib[47] describe la librería como una biblioteca completa para crear visualizaciones estáticas, animadas e interactivas en Python. Permite desarrollar gráficos de calidad de publicación a la vez que las figuras interactivas que puedan hacer zoom, desplazarse, actualizar, entre otras. A la vez que cede el control total sobre los estilos de línea, las propiedades de la fuente, las propiedades de los ejes a través de la exportación e incrustación en varios formatos de archivo y entornos interactivos.

5.7.8 Librería Seaborn

Waskom[48] ha definido a Seaborn como una herramienta para hacer gráficos estadísticos en Python. Basada en matplotlib, se integra estrechamente con las estructuras de datos de pandas, además de ayudar a explorar y comprender los datos.

Tiene funciones de trazado que operan en marcos de datos y matrices, conteniendo conjuntos de datos completos mediante los cuales se realiza un mapeo semántico y la agregación estadística necesarios para producir gráficos informativos. A través de su interfaz de programación de aplicaciones logra la manipulación de los diferentes elementos de sus gráficos para un mejor manejo de los mismos.

5.7.9 Librería Sklearn

Permite implementar funciones que evalúan el error de predicción para propósitos específicos. Las métricas que contiene se dividen en clasificación, clasificación de etiquetas múltiples, regresión y agrupación.

---

[47] MATPLOTLIB DEVELOPMENT TEAM. Matplotlib: Python plotting. [En línea]. Matplotlib.org. 13 de agosto de 2021. [Consultado: 28 de octubre de 2021]. Disponible en: https://matplotlib.org/

[48] WASKOM, Michael. An introduction to seaborn. [En línea]. Seaborn.pydata.org. [Consultado: 28 de octubre de 2021]. Disponible en: https://seaborn.pydata.org/introduction.html

5.7.10 Librería Tkinter

Según la fundación del software de Python, esta herramienta implementada en Python: “Proporciona un conjunto de herramientas de ventanas robusto e independiente de la plataforma, que está disponible para los programadores de Python que usan el paquete tkinter y su extensión, los módulos tkinter.tix y tkinter.ttk.

Las principales virtudes de tkinter, son su rapidez y que normalmente viene incluido con Python. Aunque su documentación estándar es débil, se dispone de buen material, que incluye: referencias, tutoriales, un libro y otros”[49].

[49] PYTHON SOFTWARE FOUNDATION. .Graphical User Interfaces with Tk. [En línea] Python.org. [Consultado: 28 de octubre de 2021]. Disponible en: https://docs.python.org/3/library/tk.html

6. METODOLOGÍA

Para el desarrollo del proyecto se llevó a cabo el siguiente procedimiento: Obtención de datos fílmicos para los ritmos de baile seleccionados, procesamiento de datos, creación de datasets, entrenamiento de RNAs, validación de RNAs, evaluación y validación comparativa de RNAs.

6.1 OBTENCIÓN DE DATOS FÍLMICOS

En la etapa inicial, se realizó la recolección de los datos fílmicos a través de aficionados, vídeos de youtube y tres grupos de baile, tanto uno de género urbano como dos de folklore, que permitieron la grabación de sus integrantes realizando los pasos básicos de salsa, bachata, merengue y cha cha cha, cada vídeo tiene un duración de 1 minuto: tiempo equivalente a 4 series de combos básicos mínimos por ritmo. Estos vídeos fueron grabados en una resolución de 480p x 720p con un fondo neutro para mayor confiabilidad en la detección del esquema, se tomaron 100 vídeos por baile. Las grabaciones realizadas se llevaron a cabo teniendo en cuenta los protocolos de bioseguridad y de las mismas se obtuvo el 40% del total de vídeos.

Respecto al manejo y la cantidad de los datos, se hizo un experimento con 20 personas para concretar la cantidad de fotogramas por segundo que se presentarían a las diferentes redes neuronales, se extrajeron todos los fotogramas de un vídeo y se dispuso a la presentación secuencial de los mismos. Luego, fueron extrayéndose paulatinamente hasta llegar a la mínima cantidad, la disposición de reconocimiento se presenta en el siguiente cuadro.

Cuadro 1. Comparación de análisis visual de fotogramas

Cantidad de fotogramas /seg	Cantidad de personas	Tiempo de reconocimiento (s)
25	20	3
20	20	3
15	15	3
	5	4
10	8	4
	12	5
5	13	5
	7	6
3	10	10
	10	12
2	5	No hubo reconocimiento
	15	No hubo reconocimiento
1	20	No hubo reconocimiento

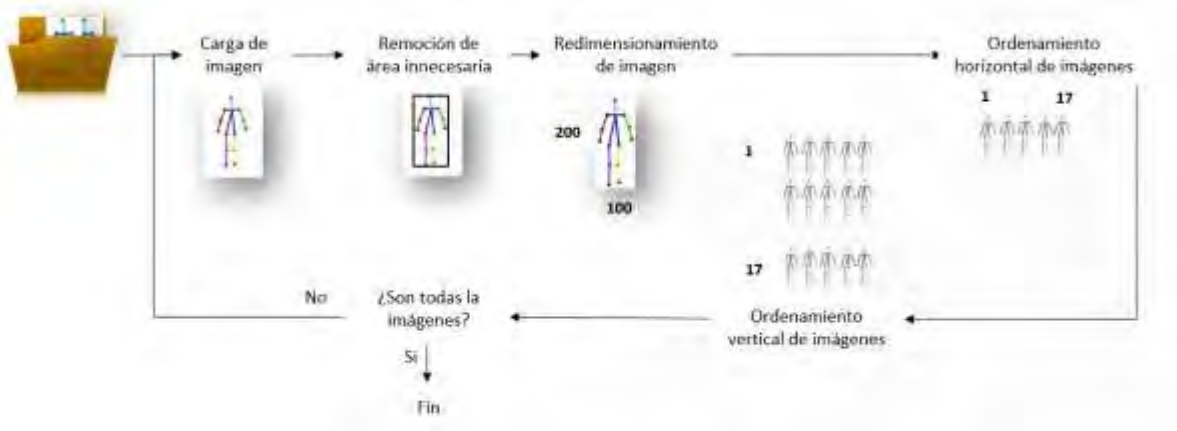
Con base en los datos recolectados, se estableció que el último valor donde el tiempo tuvo una variación mínima en comparación al inmediatamente anterior, se encuentra ubicado en 5 fotogramas por segundo donde, según la percepción de las personas del experimento, aún permanece una sensación de continuidad entre las imágenes; este dato se utiliza en el algoritmo, como la cantidad de muestras por segundo que se obtienen para el dataset de las redes.

El tipo de dataset varía de acuerdo a la estructura neural a implementar: para la red MLP se determinó un dataset de tipo .csv para mayor facilidad de procesamiento, mientras para red convolucional, se determinó una matriz cuadrada de imágenes. De esta manera, se buscó la raíz cuadrada más cercana a 300 -número de fotogramas totales en un minuto, con una muestra de 5 fotogramas por segundo-, obteniendo 17 como resultado, pues 17^2 es igual 289 y, debido a que la cantidad de datos con los que se entrenan las redes deben ser iguales para realizar la comparación, este dato determinó el número de imágenes con las que se entrenarían las RNA.

6.2 PROCESAMIENTO DE DATOS Y CREACIÓN DE DATASETS

En cuanto al procesamiento de datos, se implementó la herramienta tf-pose-estimation, un programa diseñado en Python que permite, a través de la implementación de una CNN, realizar el reconocimiento de los esquemas humanos. De manera posterior a la instalación de la ella, se procede al tratamiento de los diferentes vídeos adquiridos para la generación de diversos dataset, que permitan a las RNAs un mayor margen de trabajo ante las distintas disposiciones de las figuras humanas; el procedimiento fue implementado acorde a la siguiente gráfica.

Figura 14. Procedimiento de elaboración del dataset de la CNN



Para iniciar este proceso, se importaron las librerías necesarias para su desarrollo y los métodos de la librería de tf-pose-estimation, para que la misma no resulte modificada. Una vez establecido esto, se formularon las rutas de acceso tanto de los archivos de origen -vídeos- como de los archivos finales -imágenes y archivo .csv-; la variable cap se fija como el número de capturas de fotogramas más dos -debido a cuestiones prácticas, ya que el individuo en el primer o último fotograma presentaba inconvenientes para ser identificado-.

Posteriormente, se lee la ruta y se detecta si ésta existe o no, en caso de errar la escritura de la misma, el código permite reemplazar la carpeta de acceso para seguir con la ejecución normal. Luego, se constituye un arreglo vacío de centros -variable a utilizar más adelante- y los colores característicos para las diferentes extremidades -establecidos sobre los límites del rango numérico de color para facilidad de reconocimiento-; una vez realizado esto, se leen las carpetas dispuestas en la ruta especificada y se determinan los sufijos de los diferentes tipos de archivo a utilizar, así como las variables a utilizar: contt para contador de ritmos, shot para la posición donde se va a situar el valor para el one hot enconding y tmfin es el tamaño del vector final dispuesto para la MLP.

El algoritmo realiza el siguiente recorrido, en el orden dispuesto:

- Carpeta de ritmos
- Ritmos
- Vídeos del ritmo dispuestos en la carpeta

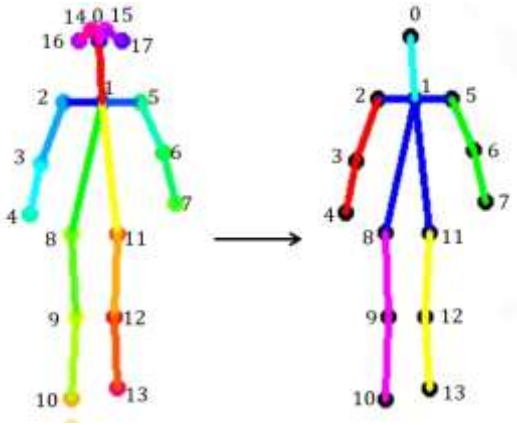
Cuando terminan todas las iteraciones del ciclo más interno, el algoritmo procede con los siguientes ritmos y repite nuevamente la iteración por carpeta hasta culminar. Internamente, para cada ritmo, se dispone de una variable para leer la cantidad de archivos .mp4 que se tienen al igual que se define la combinación binaria que identifica cada ritmo, la cual, sólo se actualiza a través del cambio de carpeta. Simultáneamente, se crea una matriz de ceros con un ancho equivalente a la cantidad de ritmos y un alto equivalente al número de vídeos del ritmo, para luego ser reemplazada por la combinación binaria del baile en cuestión -One Hot Encoding-, antes de pasar al procesamiento de los vídeos de la carpeta.

Por cada vídeo, se obtienen los datos de ancho (weight) y alto (height), buscando mantener una correcta dirección del fotograma, por ello, se establece un condicional que realiza el siguiente proceso: en caso de que el vídeo se encuentre rotado, la imagen se redimensiona a 400px200p y se procede con la respectiva corrección; en

caso contrario, únicamente redimensiona la imagen, con la intención de optimizar el rendimiento de la red y no utilizar datos innecesarios.

Una vez el fotograma fue procesado, se utilizó la herramienta tf-pose-estimation para el reconocimiento de la estructura humana. La disposición articular del esquema base del algoritmo tf-pose-estimation cuenta con 18 puntos característicos, donde los puntos 14 al 17 no son articulaciones sino puntos característicos faciales; dado que podían afectar el entrenamiento de la red por la baja precisión de su detección y no aportaban datos significativos para el reconocimiento de los ritmos -desde la perspectiva de bailarines experimentados-, se descartaron estos puntos para el proceso de dibujo de los esquemas de postura.

Figura 15. Comparativa de esquema estándar vs. Modificado



Cuadro 2. Relación articulación – número asociado

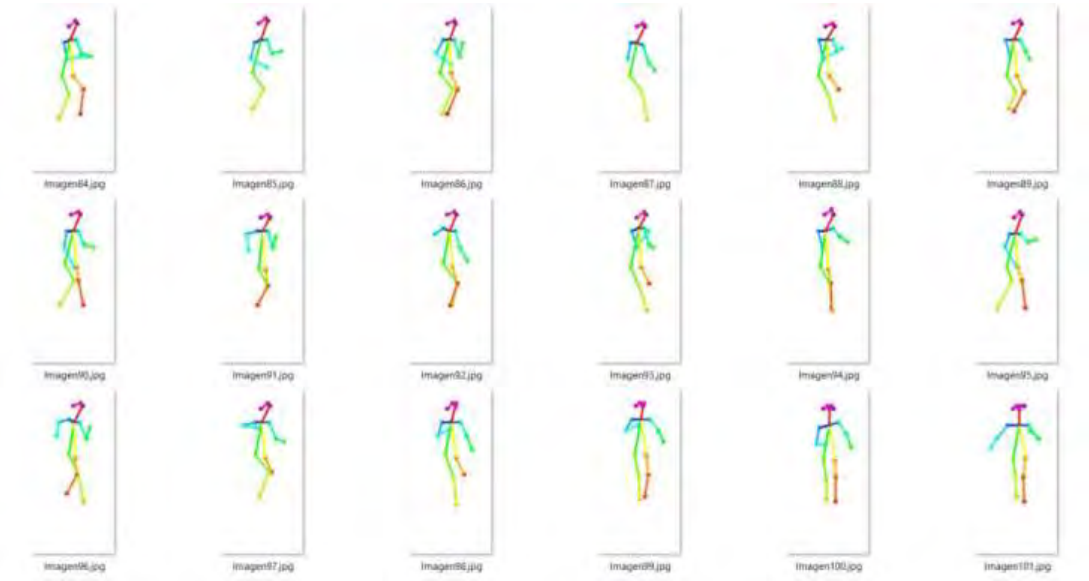
Puntos referidos	Número Asociado
Nariz	0
Cuello	1
Hombro derecho	2
Codo derecho	3
Muñeca derecha	4
Hombro izquierdo	5
Codo izquierdo	6
Muñeca izquierda	7
Cadera derecha	8
Rodilla derecha	9
Tobillo derecho	10
Cadera izquierda	11
Rodilla izquierda	12

Cuadro 2. Continuación

Puntos referidos	Número Asociado
Tobillo izquierdo	13
Ojo derecho	14
Ojo izquierdo	15
Oreja izquierda	16
Oreja derecha	17

Para obtener el esquema humano, se envía la foto para la inferencia por medio de la red convolucional; a partir de ello, se procede al desarrollo de dos tipos de datasets: Imágenes y datos numéricos. Se inicia con los datos, para los cuales se descartan los puntos descritos que no son articulaciones y se aproximan los valores a 3 cifras significativas; al momento de implementar la red se pudo evidenciar que por cada 10 imágenes con una persona desde el punto de vista lateral, se perdían los datos en 3, como se presenta en la siguiente imagen.

Figura 16. Esquemas de postura iniciales



A causa de ello, se buscó un mecanismo que permitiera reflejar una continuidad en los datos con la menor complejidad posible y que produjera los resultados más óptimos; al realizar la comparación de los diferentes tipos de interpolación, se seleccionó la interpolación lineal debido a que la variación de los puntos entre cada fotograma no es grande y, en adición, no tiene comportamientos curvos que pudieran requerir de la implementación de polinomios. En consecuencia, se procedió a una recolección de las coordenadas de las articulaciones por cada

fotograma de todo el vídeo en una tupla que, posteriormente, es iterada con el propósito de corregir los datos de la siguiente forma: si sólo existen pérdidas en el punto a analizar, se calcula el promedio -debido a que la interpolación lineal equilvadría a 0, valor que dañaría por completo la estructura generada- y, en caso de que existan dos puntos seguidos con este daño, utiliza el siguiente valor diferente de cero para realizar la interpolación, causando una proyección de datos acordes al movimiento. Adicionalmente, se revisó la desviación estándar en caso de encontrarse un valor mal calculado.

Después de ser llevada a cabo la corrección, se dibujaron los esquemas en una imagen blanca con un tamaño predeterminado de la siguiente manera:

- Basado en la proporción espacial -tamaño de la imagen-, se calcula la ubicación de los puntos articulares con ayuda de las proporciones de las coordenadas iniciales, producidas por la herramienta tf-pose estimation.

- Se dibujan los circulos donde están las coordenadas calculadas, con un color negro general para todas.

- Se dibujan las líneas conectoras, estas líneas son dibujadas con los colores determinados al inicio de la codificación.

Con las líneas y las articulaciones dibujadas, se guardó imagen a imagen en la carpeta seleccionada por ritmo, simúltaneamente, se guardó un arreglo de 1x8096 -datos correspondientes a 14 articulaciones por 2 coordenadas por 289 fotogramas concatenadas con la matriz de combinación binario que identificó el baile. En el anexo A se encuentra el código desarrollado en Python para este procesamiento y, en el anexo B, el código para la interpolación.

Al terminar las iteraciones referentes a cada ritmo, se completó la generación de datasets para la MLP, sin embargo, faltaba la concatenación de las imágenes para la CNN, por lo que se generó un algoritmo que realizó el siguiente proceso.

Figura 17. Diagrama de flujo de la generación de dataset CNN

De manera más detallada, los pasos realizados en el proceso fueron:

- Lectura de la dirección donde se encontraban las imágenes y validación, en caso de no encontrar la ruta, se solicitaría una nueva.

- Lectura de la cantidad de imágenes que hay en la carpeta.

- Lectura de la dimensionalidad de una imagen para tomar las valores de las variables.

- Calculo del número elevado al cuadrado máximo con el que se podían concatenar las imágenes en la matriz cuadrada.

- Se calcularon los límites verticales y horizontales, estos límites se definieron con la finalidad de optimizar el procesamiento, descartando los espacios donde no existía ningún dato: espacios completamente en blanco que eran innecesarios para la matriz. Ejecutando este procedimiento en una función, se utilizaron unos arreglos donde se recorrió pixel por pixel en dirección vertical (para calcular el límite horizontal) y en dirección horizontal (para el límite vertical) y, cuando se encontró un valor equivalente diferente al blanco, se guardó el valor de la columna -límites horizontales- o fila -límites verticales-, este proceso se repitió para cada imagen procesada.

- Las imágenes se concatenaron en una dirección acorde al valor cuadrado calculado, luego, se recortaron todas las imágenes con los límites determinados en el item anterior. Primero se concatenaron en forma horizontal y después en vertical, para presentar una matriz cuadrada con todos los fotogramas de ese vídeo.

Figura 18. Concatenación horizontal

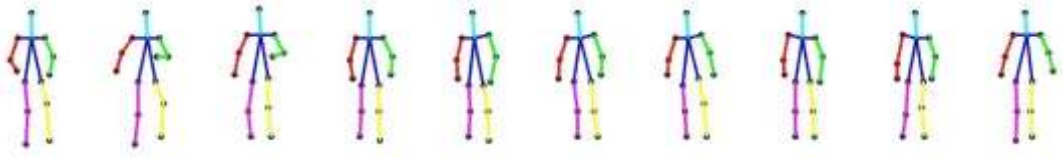
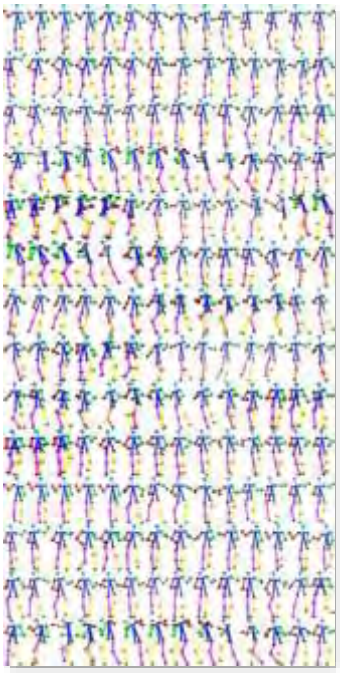


Figura 19. Ejemplo de entrada de red convolucional 2D



Con esto se generó una muestra para la red convolucional, que se repitió para cada vídeo y ritmo, consiguiendo así, el dataset de las CNN. En el anexo C se encuentra el código que contiene este proceso.

6.3 PROPUESTA DE LA MLP

En la propuesta de la MLP en Matlab, se tuvo como pretensión que la red estuviera estructurada de forma modular, para ajustarse a las necesidades de quién la use; en el anexo D se encuentra descrito el código implementado, el cual, se ve complementado por los anexos E, F, G, H, I e J. En cuanto a la MLP en Keras, se desarrolló conforme al caso específico y se encuentra en el anexo K.

6.3.1 Matlab

6.3.1.1 Lectura de datos y estructuración de la red

Los datos fueron extraídos del archivo .csv y exceptuando las últimas columnas –cuatro para éste caso-, que indican el tipo de baile -la combinación binaria anteriormente descrita-, es decir, la salida, el resto corresponden a la entrada.

Posteriormente, se adiciona una fila del tamaño total de las entradas, correspondiente al bias para la primera entrada, dicho valor se inicializa en 1 y es equivalente al parámetro wk_0. Luego, se generó un vector de números aleatorios con un rango proporcional al tamaño de la entrada, con el objetivo de crear una aleatoriedad en las entradas de la red, gracias a ello, se redistribuyeron en un nuevo orden todos los datos -tanto entradas como salidas- para el próximo entrenamiento.

6.3.1.2 Inicialización de la MLP

Se tomaron los parámetros referentes a la cantidad de datos a clasificar -número de neuronas de entrada y de salida- a partir de los arreglos obtenidos en la lectura de datos. Para la inicialización de la MLP, se generó una función que solicitaba la cantidad de neuronas de entrada, de salida, el número de capas -incluyendo la capa de salida y de entrada-, el número de neuronas por capa -exceptuando la entrada y la salida- y el tipo de función por cada capa de activación -en forma de texto-.

Dentro de la función se declaró la topología de la red -cantidad de capas más el número de neuronas de cada una-, la cantidad de capas de pesos sinápticos -equivalentes a interconexión entre las capas, es decir, el número de capas de la MLP más uno- y las celdas donde se guardan los pesos y las funciones de activación de la red.

Los valores que componen la topología de la MLP se calcularon a través de un ciclo for, que crea valores aleatorios de pesos sinápticos y asigna una función de activación determinada para cada capa de la red. En cuanto a la creación de las capas de pesos sinápticos, se utilizó la inicialización de pesos de Xavier, con un tamaño determinado: en filas, la cantidad de neuronas de la capa siguiente, y en columnas, la cantidad de neuronas de la capa actual y se adicionó una debido a los bias; respecto a la función de activación, se estableció que todas las capas ocultas tendrían como función Selu y la capa de salida tendría una función lineal.

Con los pesos sinápticos y las funciones de activación determinados, se ingresaron las dos celdas en una estructura que contiene la arquitectura de la red neural y se regresó al algoritmo central.

6.3.1.3 Funciones de activación

Para la definición de las funciones de activación, se creó una función con condicionales donde se evaluó en las estructuras a partir de la función de activación

solicitada y, de acuerdo a la seleccionada, se devolvió un arreglo de funciones anónimas, donde la primera parte del arreglo consiste en la función de activación y la segunda, en la derivada de la misma.

6.3.1.4 Hiper parámetros

Se determinan los hiper parámetros: 1000 iteraciones, un eta de 0.0005 y un Alpha de 0.0002.

6.3.1.5 Entrenamiento de la MLP

Se implementó una función que requiere de la estructura neural, los datos de entrenamiento -entrada y salida esperada-, hiper parámetros y una variable para ver el error acorde a una cantidad de épocas determinadas. En la función, se leyeron las variables de entrada y se construyó una matriz donde se guardaron los valores históricos del error; seguidamente, se creó un ciclo for que tiene como límite el número de iteraciones determinadas, en el cual, se procedió a efectuar el feedforward -desarrollado en una función que requiere las entradas y la estructura neural-, calcular el error en la capa de salida, implementar el backpropagation -en una función que requiere la salida por capa, la neta de las capas, el error de la capa de salida, la red neural y los parámetros de aprendizaje-, calcular el error cuadrático medio y promediar el error. Finalmente, de acuerdo a un parámetro elegido respecto a las épocas, se visualizó el error en la consola acorde a esta cantidad de iteraciones determinadas.

Dentro de la función feedforward, se crearon los arreglos de las salidas y las netas, antes de proceder a las operaciones, el límite de la iteraciones del ciclo for fue definido por la cantidad de capas de la red y la entrada de la red se determinó como la salida de la primer capa antes de iniciar con la estructura cíclica. Dentro del ciclo for implementado, se calculó la neta de cada capa y se procedió a evaluar la neta en cada función de activación, para las capas ocultas se adicionó un arreglo del tamaño equivalente a la entrada de la misma y el valor de salida de ésta se guardó en la posición siguiente, no obstante, en la última capa únicamente se tomaron los valores resultantes de la evaluación en la función de activación. A través de este ciclo se definen dos arreglos de celdas que contienen los salidas y las netas de todas las capas de la red neuronal.

En el código de training se calculó la diferencia entre la salida esperada y la obtenida con estos valores, buscando establecer el error de la última capa de la red para proceder con el backpropagation.

Dentro de la función backward, se especificó el valor de iteraciones respecto a la cantidad de capas y de neuronas de la capa de salida, al igual que se leyeron los parámetros de aprendizaje y se creó una celda de ceros con la variación de los deltas por capa; dentro de las iteraciones, se evaluó primero la última capa, donde se calculó el delta como el error de la capa de salida por la neta evaluada en la derivada de la función de activación. Ahora bien, para el caso de las otras capas, primero se halló el error con ayuda del delta de la capa siguiente -es importante recalcar que como va de adelante hacia atrás, el valor anterior sería el de la capa siguiente-, el cual, es multiplicado por los pesos sinápticos de la capa evaluada; una vez adquiridos los errores por neurona, se calculó el valor del delta de la capa con el mismo procedimiento realizado para la capa de salida. Posteriormente, se realizó la actualización de los deltas, multiplicando el eta por el delta de la capa por la salida más el alfa por la variación del delta -que puede ser positivo o negativo-; el valor originado se adiciona a los pesos de las capas. Finalmente, para terminar el ciclo, se actualizó el valor de las variaciones.

Con la actualización hecha, se calculó el error cuadrático medio de la red antes de avanzar a la siguiente iteración.

6.3.2 Keras

6.3.2.1 Importación de librerías y datos

Se importaron las librerías que permitieran el manejo numérico, visión computacional, creación de gráficos, herramienta de análisis y manipulación de datos, métricas estadísticas en Python además de las clases para modelos secuenciales que fueran de utilidad para la red MLP. A continuación, se cargó el archivo que contenía los datos -.csv- y se transformó a un formato tipo tabla.

6.3.2.2 Generación de etiquetas y división de datos

A partir de la combinación del OHE, se precisó una etiqueta de texto que indicara el ritmo y se concatenó con el arreglo que contenía todos los datos para el entrenamiento de la red.

Para la división de datos, primero se seccionó el arreglo generado entre datos de entrada y de salida, teniendo las salidas una etiqueta escrita referente al ritmo; seguidamente, se dividieron los datos nuevamente en entrenamiento y validación, teniendo un porcentaje de 90% y 10% respectivamente. Aún así, fue necesario

hacer una última división para entrenar, dado que la MLP trabaja con números, por lo que se separó la etiqueta escrita de los datos, pero conteniendo información respecto a la clasificación de ritmos.

6.3.2.3 Generación de capas, concatenación, declaración de hiper parámetros y entrenamiento

Se determinó un modelo secuencial, con funciones de activación Selu -seleccionada por tener el mejor desempeño- y un número de neuronas por capa equivalente a 150; después, se realizó una generación de capas de forma consecutiva con los mismos parámetros -exceptuando la primera y la última, cuyo tamaño correspondía de forma proporcional a los datos de entrada y salida respectivamente. En adición, la capa de salida tiene como la función softmax para su activación.

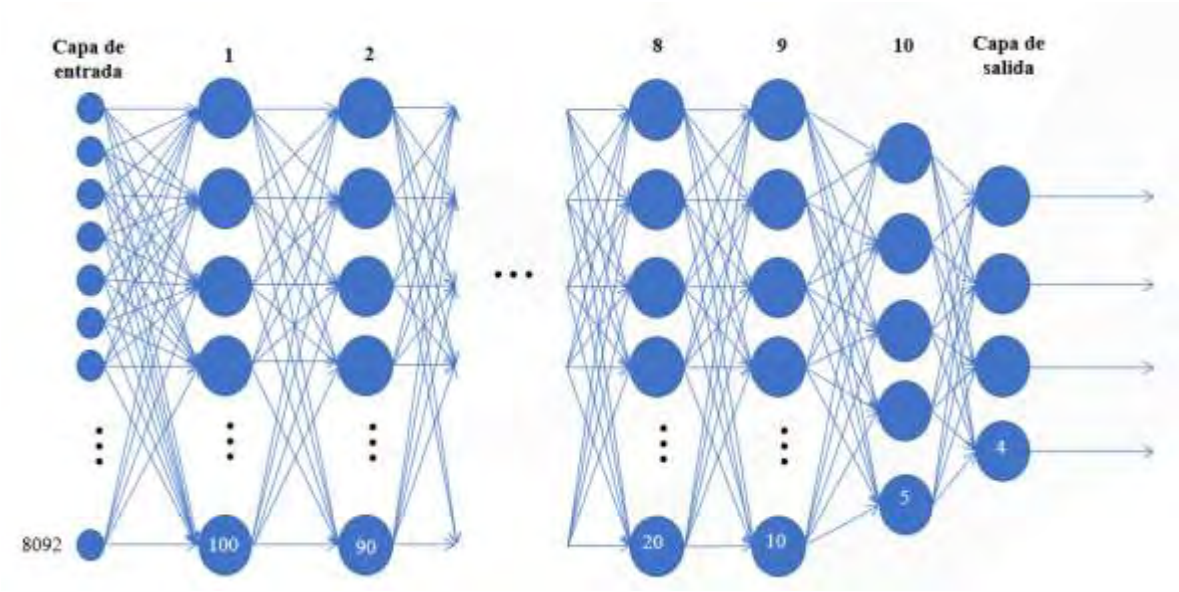
Sucesivamente, se establecieron el optimizador, el parámetro de aprendizaje, el eta, la función de pérdida y las métricas, simultáneamente a un guardado iterativo, con el objeto de guardar el mejor modelo neural generado, justo antes de poner a entrenar la estructura neuronal diseñada en 2000 épocas.

6.3.3 Estructuras neurales

Las diferentes propuestas elaboradas para la MLP se encuentran dispuestas a continuación; todas constan de 8092 neuronas en la capa de entrada -equivalente al tamaño del vector de entrada dispuesto por la cantidad de fotogramas- y de 4 neuronas en la capa de salida -correspondiente al OHE-. El arquetipo gráfico se encuentra inspirado en el modelo propuesto por Haykin en su libro titulado *“Neural Networks and Learning Machines”* en la página 124.

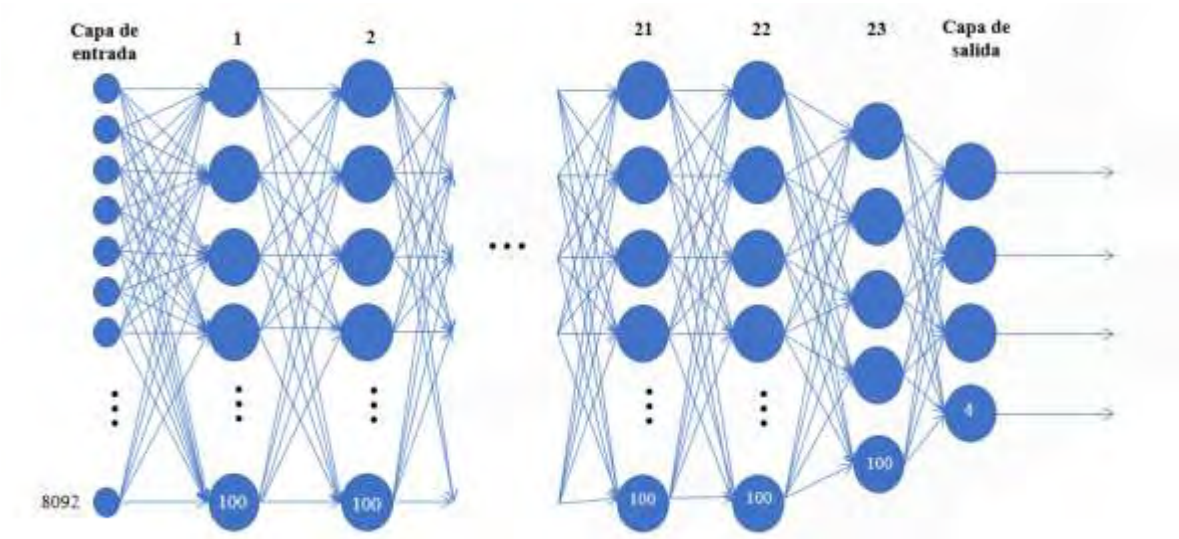
La primera estructura consta de 10 capas ocultas con función de activación tangente sigmoidal y en la capa de salida, una función lineal; con el número de neuronas decreciendo a partir de las primeras 100. Únicamente fue utilizada para Matlab.

Figura 20. Primera arquitectura neural de la MLP



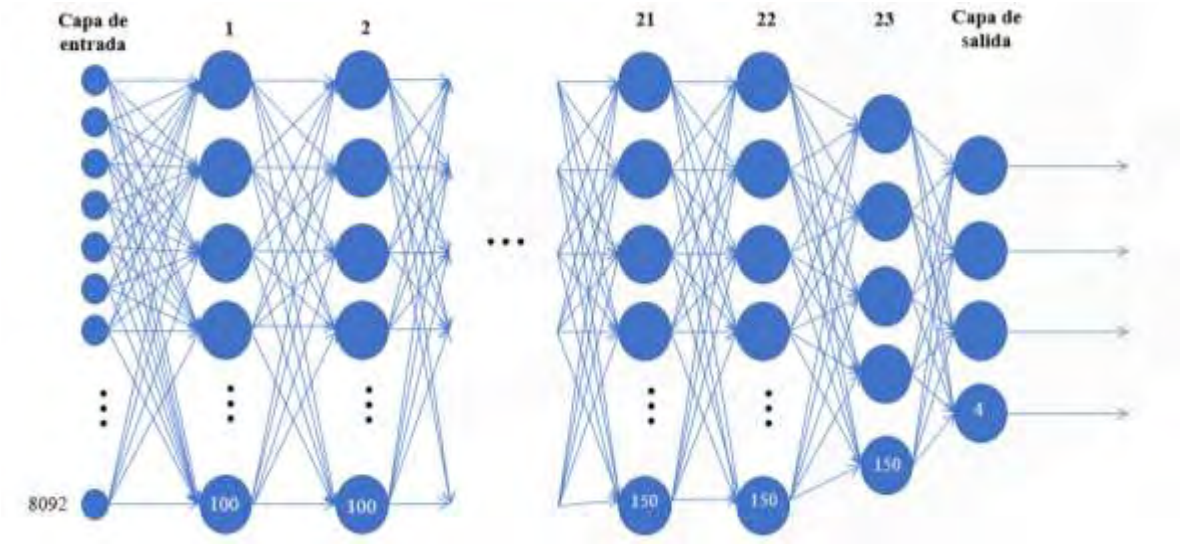
El segundo modelo se aplicó en ambos softwares, con una modificación en la cantidad de neuronas por capa y la cantidad de las mismas: la primera se vuelve constante y es de 100, en tanto que la segunda, pasa de 10 a 23. A pesar de que las funciones de activación en las capas ocultas no varían en ninguno de los dos casos, la diferencia radica para las de salida: Mientras en Matlab se utiliza lineal, para Keras se hizo uso de la función Softmax.

Figura 21. Segunda arquitectura neural de la MLP



Finalmente, en el último esquema se alteran únicamente la cantidad de neuronas de las capas ocultas -de 100 a 150-, puesto que se mantienen las funciones de activación anteriormente descritas.

Figura 22. Tercera arquitectura neural de la MLP



Para una comparación más consistente de las diferentes topologías generadas para la red MLP, se realizó el siguiente cuadro.

Cuadro 3. Análisis comparativo de topologías de MLP

	MLP Matlab 1	MLP Matlab 2	MLP Matlab 3	MLP keras 1	MLP keras 2
Número de capas	10	23	23	23	23
Neuronas por capa	[100, 90, 80, 70, 60, 50, 30, 20, 10, 5]	100	150	100	150
Función de activación en las capas ocultas	Tangente sigmoidal	SeLU	SeLU	SeLU	SeLU
Función de activación en la capa de salida	Lineal	Lineal	Lineal	Softmax	Softmax
Tiempo de entrenamiento	1 Horas	3 Horas	4 Horas	3.5 Horas	4.5 Horas
Parámetros entrenables	841.089	1.052.104	1.346.004	1.052.104	1.346.004

6.4 PROPUESTA DE LA CNN

6.4.1 Importación de librerías y lectura de datos

Se importaron las librerías que permitían el manejo numérico, visión computacional, creación de gráficos, herramienta de análisis y manipulación de datos, métricas estadísticas, así como las diferentes clases para modelos secuenciales que fueron de utilidad para la red MLP y las capas para generar la convolucional, el max pooling y el aplanamiento. Acto seguido, se accedió a las imágenes anteriormente cargadas a la plataforma de Drive.

Posteriormente, se cargó un fotograma con el fin de conocer las dimensiones de las imágenes y poder crear variables vacías de nombre I e Y para guardar tanto las imágenes como el etiquetado, respetando respectivamente, las dimensiones necesarias para el uso de las librerías.

6.4.2 Generación de etiquetas, aleatorización y división de datos

Al recorrer la carpeta donde las imágenes están contenidas, empezó a almacenarse cada imagen en la variable I, luego, la imagen fue relacionada a la etiqueta correspondiente de acuerdo a la ruta de acceso. Es decir, se estableció un comparativo con el nombre de la carpeta donde se encontraba guardada y el indicador de etiqueta para generar el esquema de OHE acorde al ritmo, para ser guardado en la variable Y.

Una vez terminado este proceso, se realizó una aleatorización de los datos para no tener datos consecutivos de la misma clase, es decir, hay una redistribución de las variables I e Y. A partir de allí, se generó la última división de datos, que constó de 4 variables, correspondientes a: los datos en entrenamiento (X_train, y_train) y los datos de validación (X_test y y_test), el porcentaje de valores correspondientes a cada segmento fue de 90% y 10% respectivamente, con el propósito de brindar la mayor cantidad de referencias a la red neuronal para unos mejores resultados.

6.4.3 Entrenamiento de la CNN

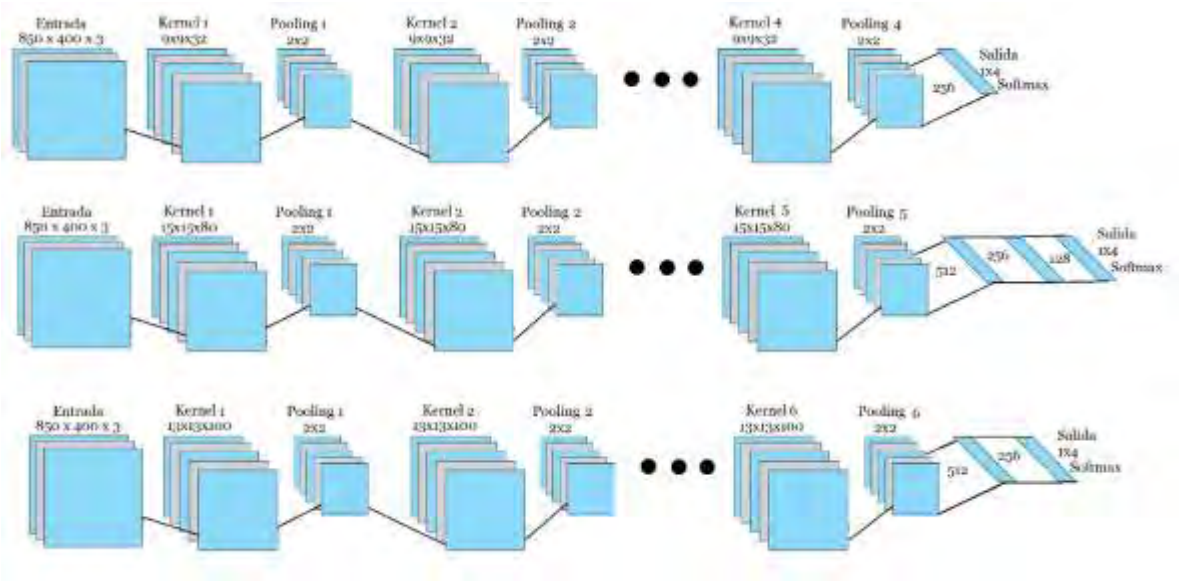
Para la elaboración de esta red, se utilizó la capa importada de la librería de Keras, Conv2D, que permitió la generación de seis capas convolucionales con 100 filtros, de tamaño 13x13, con una función Kaiming He para su inicialización y una función de activación selu.

En cuanto al max pooling, se estableció un valor para el pooling de 2x2, que se aplicó igualmente para todas las capas antes de ser aplanadas para servir de entrada a la MLP; dicha MLP, cuenta con tres capas: dos con función de activación selu y la capa de salida con una función softmax. Para finalizar la elaboración de la CNN, se estableció el optimizador adadelta para la red, en conjunto con el parámetro de aprendizaje, el eta, la función de pérdida y las métricas; se generó un guardado iterativo, para el mejor modelo neural obtenido, antes de poner a entrenar la estructura neuronal diseñada en 1000 épocas.

6.4.4 Estructuras neurales

Referente a la arquitectura de las CNN, sus modificaciones fueron presentadas en cuestiones de capas, filtros y composición de la MLP. Con relación a los filtros, se efectuó un ajuste sucesivo a las dimensiones en proporciones equitativas de ancho y alto de 9, 15 y 13; en tanto que las cifras de los mismos y las capas convolucionales tuvieron un aumento ininterrumpido: De 32, 80 y 100, para las primeras, y de 1 consecutivo para las segundas. En relación a la MLP, su alteración es de 1, 3 y 2, con el número de neuronas establecido descendiendo a la mitad de la capa anterior y empezando por 512. Cabe resaltar el arquetipo gráfico está inspirado en el modelo propuesto por Haykin en su libro titulado *"Neural Networks and Learning Machines"* en la página 202.

Figura 23.Estructuras neurales de las CNN



Para una mejor interpretación de las diferentes topologías generadas elaboradas para la red CNN, las características de cada esquema se disponen en el siguiente cuadro.

Cuadro 4. Análisis comparativo de topologías de CNN

	CNN Keras 1	CNN Keras 2	CNN Keras 3
Número de capas Convolucionales	4	5	6
Filtros por capa convolucional	32	80	100
Dimensiones de los filtros	9x9	15x15	13x13
Función de activación en las capas ocultas	SeLU	SeLU	SeLU
Número de capas MLP	1	3	2
Neuronas por capa MLP	512	[512, 256, 128]	[512, 256]
Función de activación en las capas ocultas MLP	SeLU	SeLU	SeLU
Función de activación en la capa de salida	Softmax	Softmax	Softmax
Parámetros entrenables	21.968.100	18.759.172	12.627.768

7. RESULTADOS

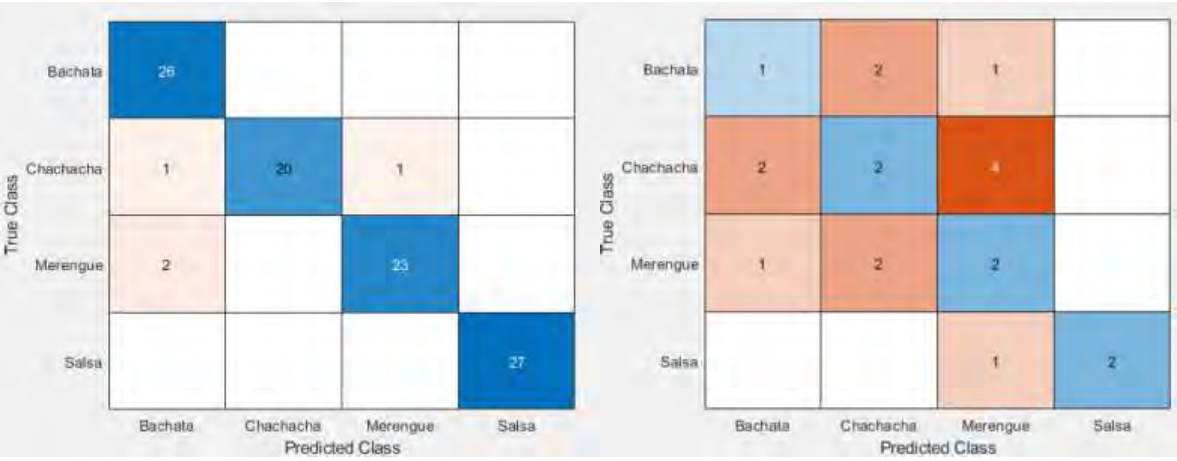
Para la evaluación de los resultados, se definió la matriz de confusión como herramienta para evaluar el rendimiento de la red. La razón de utilizar las métricas consecuentes de este mecanismo, se debe a que el ánalisis de la red se toma frente a los datasets elaborados, es decir, se evaluó la secuencialidad de los pasos en el tiempo, tanto en imágenes como en puntos, comparando los datos de entrenamiento contra los de validación. Dichos parámetros de medición, sólo podían ser obtenidos comparando la red en sí misma con los valores esperados, ya que esta apreciación no podía ser permeada por la valoración en la calidad de la estimación, pues fue suplida con la corrección realizada en el procesamiento de los datos conseguidos con el pose estimation.

7.1 MLP

7.1.1 Matlab

Para la visualización de la matriz de confusión, se generó un código donde, en un arreglo g, se guardó la posición donde se encontró el mayor valor a la salida, para determinar de forma más precisa el resultado del OHE en los casos donde los valores son confusos. Posteriormente, se generó un contador donde se acumuló la cantidad de vídeos por etiqueta; y después, se realizó una contraposición con la salida esperada para la visualización práctica del número de aciertos y errores, que presentaba cada red a través de una matriz de confusión.

Figura 24. Matrices de confusión para el primer entrenamiento MLP MATLAB

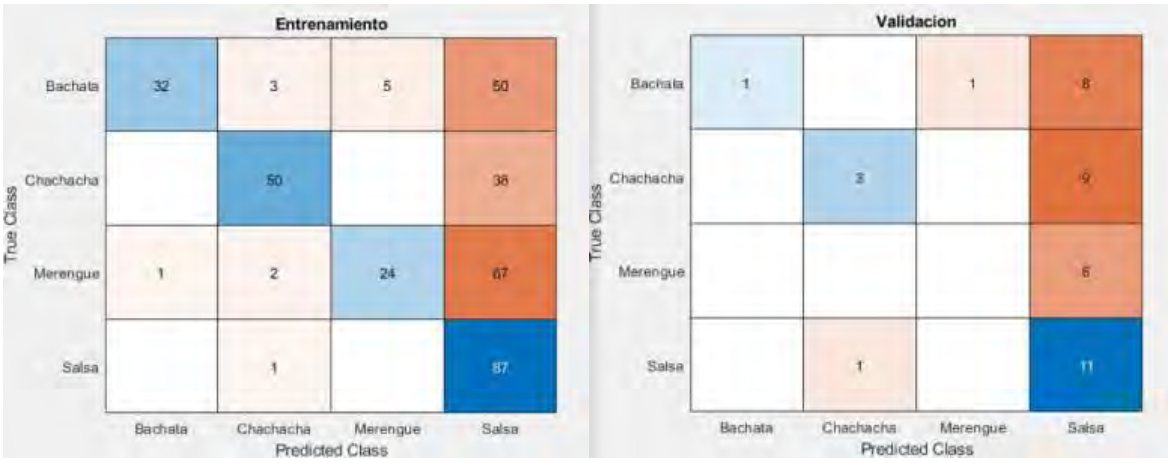


Cuadro 5. Tabla de métricas para el primer entrenamiento MLP MATLAB

	Entrenamiento		Validación	
Ritmo	Precisión	Sensibilidad	Precisión	Sensibilidad
Bachata	0.9	1	0.25	0.25
Chachachá	1	0.91	0.33	0.25
Merengue	0.96	0.92	0.25	0.4
Salsa	1	1	1	0.5
Exactitud = 0,96			Exactitud = 0,35	

Para el primer entrenamiento, se evidenciaron respuestas parciales debido a un dataset temporal de 120 videos, se puede ver la diferencia entre la exactitud del entrenamiento, correspondiente a un 96%, en comparación a la validación, de 35%. Una posible razón de este fenómeno puede ser un sobre entrenamiento de la red y la necesidad de una mayor cantidad de datos, para poder extraer suficiente información, con el fin de realizar una buena generalización de cada una de las clases.

Figura 25. Matrices de confusión para el segundo entrenamiento MLP MATLAB



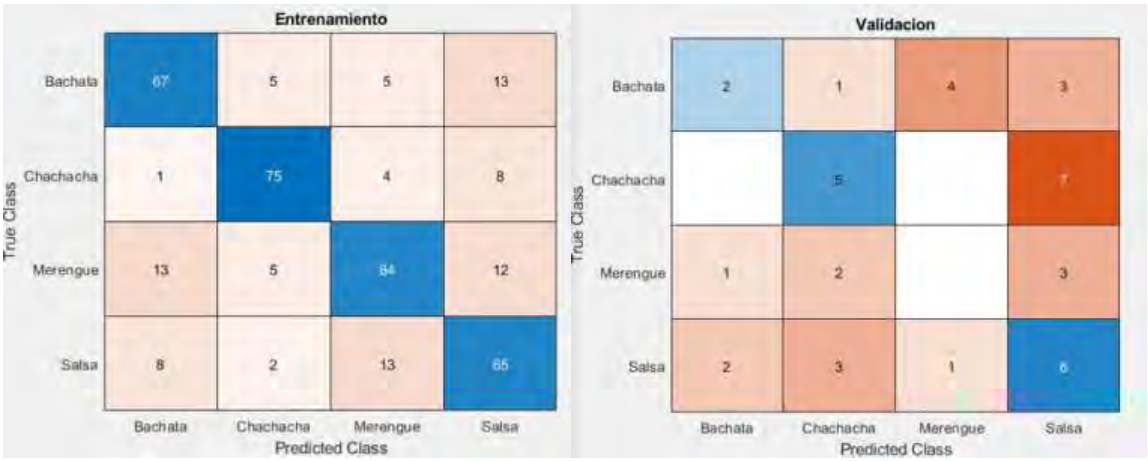
Cuadro 6. Tabla de métricas para el segundo entrenamiento MLP MATLAB

	Entrenamiento		Validación	
Ritmo	Precisión	Sensibilidad	Precisión	Sensibilidad
Bachata	0.97	0.36	1	0.1
Chachachá	0.89	0.57	0.75	0.25
Merengue	0.83	0.26	0.0	0.0
Salsa	0.36	0,99	0.32	0.92
Exactitud = 0,53			Exactitud = 0,38	

Así las cosas, elevando la meta total de vídeos de 120 a 400, se realizó un nuevo entrenamientode la red. En este caso se dividieron los datos en la misma proporción que en el caso anterior, es decir, 90% para entrenamiento y 10% para validación, siendo esta cifra equivalente a 360 vídeos y 40 vídeos, respectivamente.

Al realizar un análisis de las métricas, se apreció una mejora considerable en la exactitud de ambas partes, la exactitud en el entrenamiento disminuyó al 53% pero en la validacion aumentó a 38%. También se evidenció, que hubo una clase de baile que costó mucho más generalizar, dicho ritmo fue el conocido como “Merengue”.

Figura 26. Matrices de confusión para el tercer entrenamiento MLP MATLAB



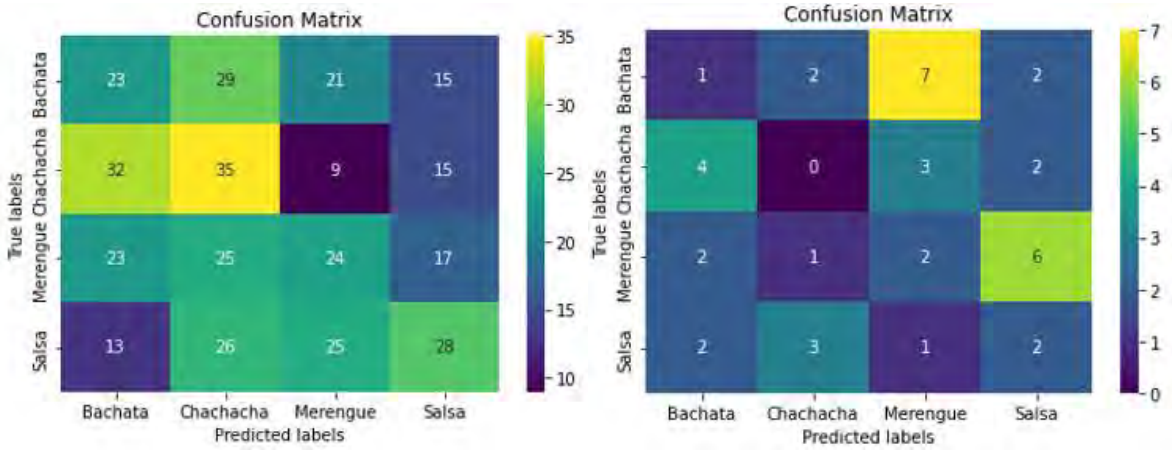
Cuadro 7. Tabla de métricas para el tercer entrenamiento MLP MATLAB

	Entrenamiento		Validación	
Ritmo	Precisión	Sensibilidad	Precisión	Sensibilidad
Bachata	0.75	0.74	0.4	0.2
Chachachá	0.86	0.85	0.45	0.42
Merengue	0.74	0.68	0.0	0.0
Salsa	0.66	0.74	0.32	0.5
Exactitud = 0,75			Exactitud = 0,32	

Con este entrenamiento se corroboró, que no necesariamente el aumento en la cantidad de neuronas brinda una mejora en la calidad de los resultados, dado que se presenta una desmejora en la tendencia de la generalizacion de la red, visualizada en la exactitud del 32% en validación. También se apreció que las métricas en el entrenamiento disminuyeron respecto a la precisión, pero mejoraron en la sensibilidad.

7.1.2 Keras

Figura 27. Matrices de confusión para el primer entrenamiento MLP Keras



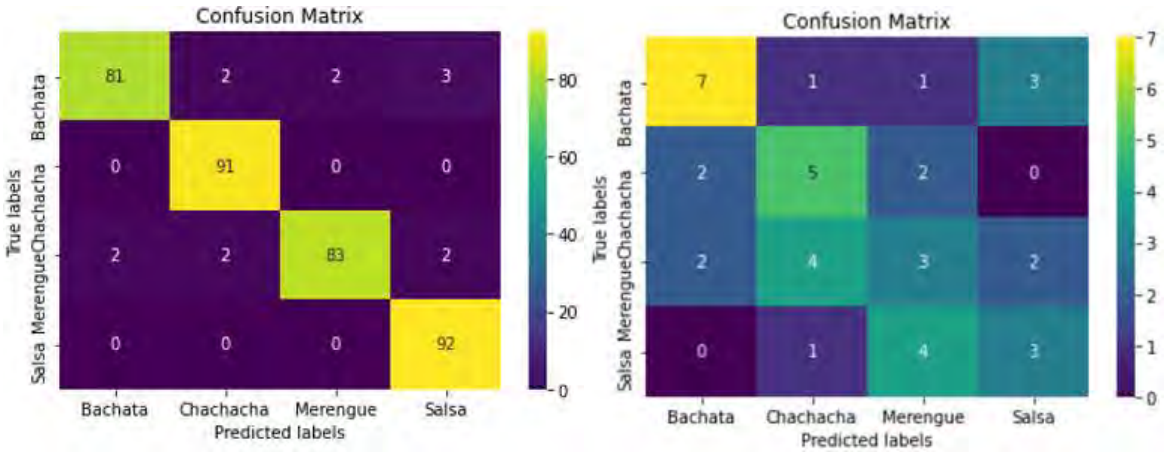
Cuadro 8. Tabla de métricas para el primer entrenamiento MLP Keras

	Entrenamiento		Validación	
Ritmo	Precisión	Sensibilidad	Precisión	Sensibilidad
Bachata	0.25	0.26	0.11	0.08
Chachachá	0.30	0.38	0.0	0.0
Merengue	0.30	0.27	0.15	0.18
Salsa	0.37	0.3	0.17	0.25
Exactitud = 0.31			Exactitud = 0.12	

Vemos como, para el primer entrenamiento, se tuvo referenciada la topología usada en MATLAB, por lo que se decidió entrenar la red con la misma estructura, a pesar de ello, no se obtuvo el mejor rendimiento.

Analizando los datos, se aprecia que la exactitud en el entrenamiento tuvo un desempeño del 31% y para validación un 12%, debido a estos índices se llegó a la conclusión de que se requerían unas alteraciones en la estructura de MLP.

Figura 28. Matrices de confusión para el segundo entrenamiento MLP Keras



Cuadro 9. Tabla de métricas para el segundo entrenamiento MLP Keras

	Entrenamiento		Validación	
Ritmo	Precisión	Sensibilidad	Precisión	Sensibilidad
Bachata	0.98	0.92	0.64	0.58
Chachachá	0.96	1	0.45	0.56
Merengue	0.98	0.93	0.30	0.27
Salsa	0.95	1	0.38	0.38
Exactitud = 0.96			Exactitud = 0.45	

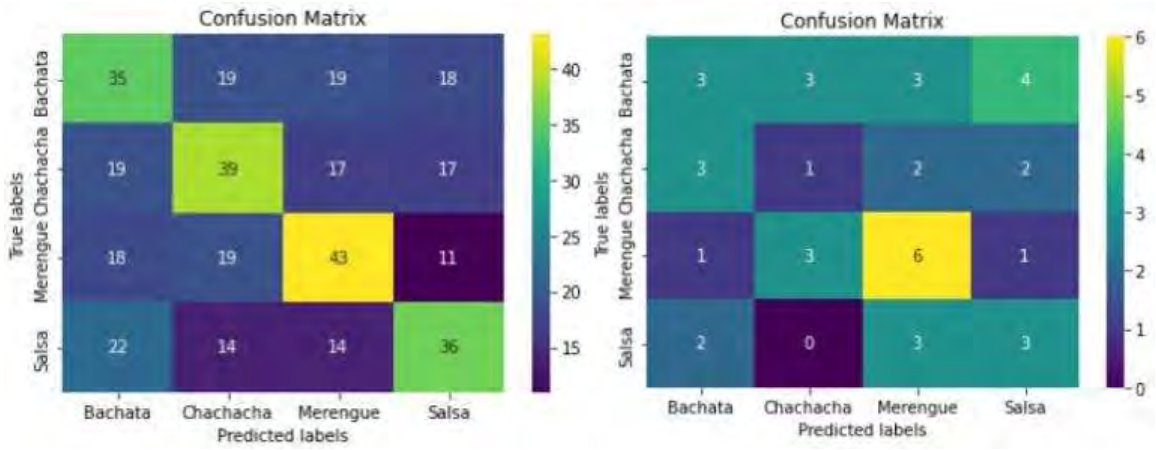
Se apreció una mejora muy considerable debido a la implementación de optimizadores con mejoras, por ejemplo el Adadelta -al que se accede con el uso de la librería de keras-, y la adición de una capa con función de activación softmax a la salida de la red; lo que permitió una mejora en los resultados, visible en la exactitud para el entrenamiento con 96% y para validación con 45%.

Al realizar un análisis, se puede apreciar la mejora de rendimiento de la MLP, sin embargo, no fue la mejora esperada; una posible razón, es que este tipo de redes son poco usadas y factibles en el momento de revisar secuencias temporales, debido a las posiciones de los datos en el vector de entrada, es muy poco robusto a cambios espaciales de las imágenes. También se alcanzaron estas conclusiones al momento de elaborar la red MLP sobre Matlab, ya que no se utilizaron herramientas, dicho de otra manera, una implementación desde cero, donde no se contó con gran facilidad para modificar el optimizador (SGD para este caso), en comparación a la flexibilidad brindada por las librerías en Keras, donde variar los parámetros y la estructura de la capa se hizo con mayor facilidad.

7.2 CNN

Para el caso de las redes convolucionales, se hizo uso de las etiquetas de texto generadas en el código para generar las matrices de confusión.

Figura 29. Matrices de confusión para el primer entrenamiento CNN

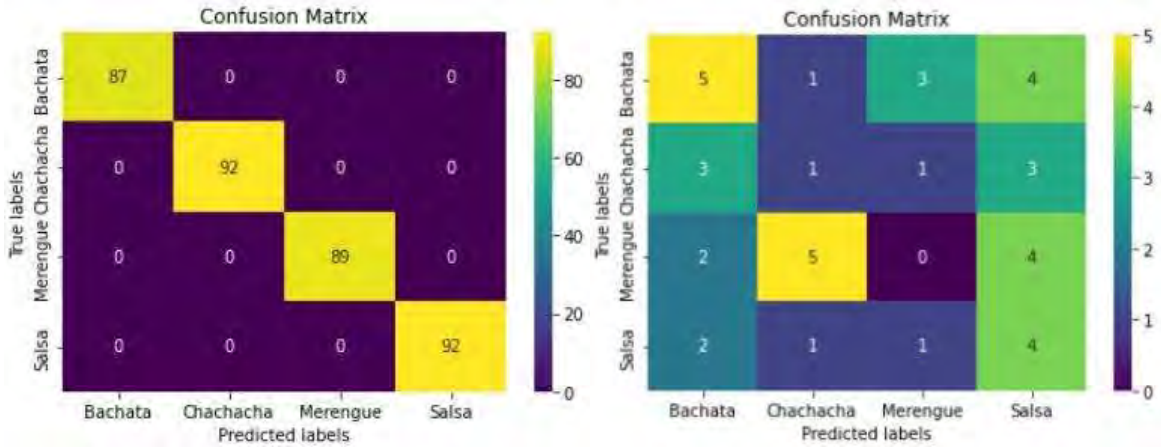


Cuadro 10. Tabla de métricas para el primer entrenamiento CNN

	Entrenamiento		Validación	
Ritmo	Precisión	Sensibilidad	Precisión	Sensibilidad
Bachata	0.37	0.38	0.33	0.23
Chachachá	0.43	0.42	0.14	0.12
Merengue	0.46	0.47	0.43	0.55
Salsa	0.44	0.42	0.30	0.38
Exactitud = 0.42			Exactitud = 0.33	

Para la primera estructura propuesta, se presentan resultados bastante divergentes, una de las posibles causas pudo ser que la topología de la CNN propuesta, no contara con la estructura necesaria para realizar la debida extracción de características, se ve una exactitud de 42% y 33%, para entrenamiento y validación respectivamente. Con estos valores se hace cambio en la estructura esperando mejora.

Figura 30. Matrices de confusión para el segundo entrenamiento CNN

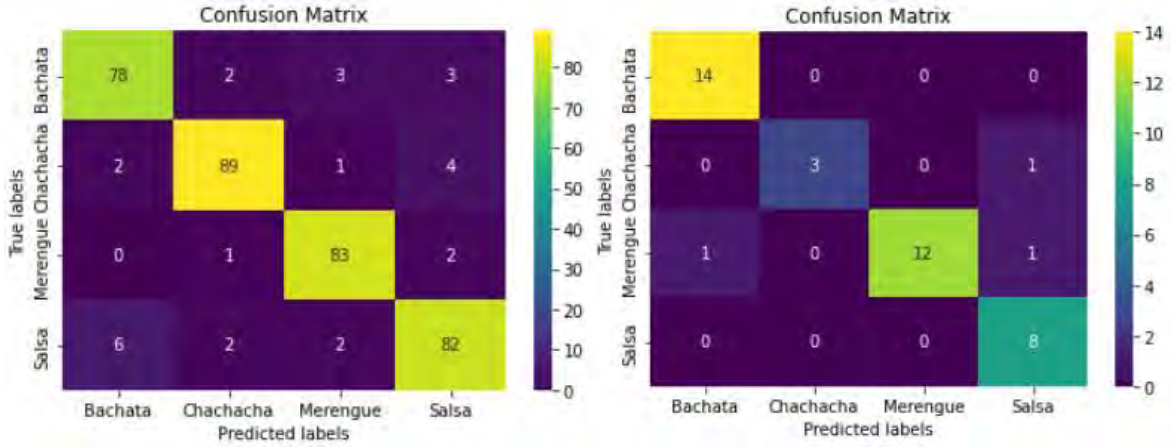


Cuadro 11. Tabla de métricas para el segundo entrenamiento CNN

	Entrenamiento		Validación	
Ritmo	Precisión	Sensibilidad	Precisión	Sensibilidad
Bachata	1	1	0.42	0.38
Chachachá	1	1	0.12	0.12
Merengue	1	1	0.0	0.0
Salsa	1	1	0.27	0.5
Exactitud =1			Exactitud = 0.25	

En este ejemplo, se pudo observar una mejora altamente considerable respecto a las métricas de entrenamiento, pese a todo, se presentaron resultados mixtos respecto a los datos de validación. Gracias a ello, se apreció el beneficio del cambio en la estructura de la CNN, a la vez que exigió una remodelación adicional para mejorar los datos de salida.

Figura 31. Matrices de confusión para el tercer entrenamiento CNN



Cuadro 12. Tabla de métricas para el tercer entrenamiento CNN

	Entrenamiento		Validación	
Ritmo	Precisión	Sensibilidad	Precisión	Sensibilidad
Bachata	0.91	0.91	0.93	1.0
Chachachá	0.95	0.93	1.0	0.75
Merengue	0.93	0.97	1.0	0.86
Salsa	0.9	0.89	0.8	1.0
Exactitud = 0.92			Exactitud = 0.93	

Teniendo en cuenta las fallas de la estructura anterior se realizaron las correcciones pertinentes a la estructura de la red, con un entrenamiento gradual donde se iba variando el parámetro de aprendizaje para épocas muy cortas, es decir, para 100 épocas se entrenaba con un valor, y para las siguientes con un valor más pequeño. Luego de 1000 épocas este modelo fue el de mejor rendimiento, con un 92% de exactitud en entrenamiento y 93% en validación.

La estructura de la CNN en la parte convolucional es bastante importante debido a las siguientes razones:

- Si se hace muy compleja, es decir, una estructura muy grande con diversos parámetros de entrenamiento para determinar, se debe contar con una alta

densidad de datos para entrenamiento, situación que el caso concreto, se dificultó conseguir, generando errores en la estructura desarrollada, debido a la falta de datos de entrenamiento requeridos por la profundidad de la red.

- En el caso contrario, se evidenció un sobre entrenamiento de la CNN (múltiples veces), debido a la variación de parámetros dependientes directamente de la cantidad de datos, caso que tampoco dio buenos resultados.

- Finalmente, se generaron diversas estructuras y de manera empírica, se obtuvo la mejor; visualizando el progreso paulatino de los modelos, se llegó a la estructura 3 donde la exactitud en validación del modelo fue del 92% en entrenamiento y 93% en validación.

8. CONCLUSIONES

A través de la realización de este trabajo, se evidenció que para el funcionamiento de un modelo neuronal en aplicaciones totalmente diferentes a las convencionales, se debe hacer uso de alternativas que permitan optimizar el rendimiento estándar de ellas. Por ejemplo, implementar librerías especializadas como Keras, para el análisis y desarrollo de las redes. También se puede hacer uso de procesos híbridos o alternos como el Transfer Learning, para reducir el tiempo de entrenamiento y disminuir la dificultad en la generalización del modelo.

A su vez, durante la realización de esta investigación se apreció que la parte más ardua de la labor es recaudar información para el dataset para un proyecto nuevo de Inteligencia artificial, en este caso, de clasificación; ya que es necesario que cuente con todas las características específicas para el modelo a entrenar, los datos deben ser lo más homogéneos posible -en cuestiones de toma de datos y resolución-. Además, se debe contar con una base de datos considerable para permitir tener generalidades bastante amplias en futuras predicciones del modelo, en este caso particular, la adquisición del dataset tuvo una alta complejidad y el progreso paulatino en su recolección se evidenció en los cambios obtenidos en las redes.

Respecto a las diferentes estructuras de las redes, se propone un énfasis referente a la importancia acerca de la topología adecuada para cada grupo de datos a entrenar, con el fin de tener el menor tiempo posible en entrenamiento para evitar consumos computacionales innecesarios; además, el tiempo de entrenamiento va acorde a las estructuras definidas. Hecho que se puede apreciar a través de las múltiples propuestas, unas con mejores resultados que otras y resaltando que la mejor respuesta se obtuvo de la CNN, pero el tiempo de entrenamiento de la misma es mucho mayor que la MLP. Esto permite concluir que, a pesar de tener como métrica importante el tiempo, la misma palidece en comparación a la asertividad, por lo que el factor referido para la selección de la CNN como red idónea en esta clasificación en particular, son las métricas de la matriz de confusión.

A nivel estructural de las redes MLP para este proyecto, se puede concluir que, en el software de Matlab, la topología que entrega mejores resultados tiene por composición: en las capas ocultas, 100 neuronas con función de activación SeLU y, en su capa de salida, una función de activación lineal. El número de neuronas y funciones de activación seleccionadas, se obtuvieron debido a una importante cifra de entrenamientos y comparación entre múltiples propuestas de topologías, en las que se presentaron los siguientes inconvenientes: Por un lado, el gran volumen de neuronas y capas excedió la capacidad de la MLP a causa de la baja la cantidad de datos que disponibles respecto a su densidad y, por otro lado, ante un tamaño

limitado de neuronas y capas ocurre un sobre entrenamiento, lo que impidió una buena generalización de la clasificación.

Con ayuda de la librería de Keras, se aplicó un cambio a las topologías anteriormente diseñadas en Matlab en la función de activación de la capa de salida, modificando de lineal a Softmax, buscando evaluar de forma más directa el avance de la persona. No obstante, la MLP de 150 neuronas -que tuvo el mejor rendimiento en Keras- no demostró el mejor desempeño y, a pesar de que la red elaborada en Matlab presentó una mayor generalidad, se concluyó que una MLP no es la mejor opción para la labor que se requería, procediendo al diseño estructural de la red CNN para comparar su funcionamiento.

A pesar de todo, dichas diferencias son justificadas debido a las funciones de activación de la capa final, donde divergen las proyecciones porque la función Softmax se usa para clasificación directamente, calculando las probabilidades dentro de la red; mientras, con la función lineal en la salida, se realiza un post procesamiento en los resultados, que puede provocar variaciones en la forma en que se presenta la respuesta de la red. Adicionalmente, los tiempos de entrenamiento de cada una varían, y no necesariamente corresponden a la obtención del mínimo error esperado, sino a la terminación de las iteraciones designadas, las cuales, también varían de un software a otro.

Para las CNN, las diferencias se centraron en la cantidad de capas -tanto convolucionales como de MLP-, filtros y dimensiones de los mismos; gracias a ello, el modelo que brindó la respuesta idónea para el objetivo del proyecto fue la estructura más densa en cantidad de capas convolucionales y, en el tema de la MLP, aquella cuya densidad era media en comparación a las otras. Este resultado puede justificarse con base en la dimensión de los filtros y su cantidad por capa, debido a que la segmentación de características realizada en esta estructura es más acorde a lo requerido, realidad que se refleja en las métricas de la matriz de confusión. Aunque las funciones de activación no variaron de una CNN a otra en ninguna de sus capas, la selección de las mismas se debió al comportamiento demostrado en la implementación experimental tanto de la red convolucional, como en la MLP.

Teniendo en cuenta el desempeño de las redes neuronales, se aprecia que la estructura neuronal con mejor rendimiento es la CNN en comparación con la MLP, claramente se debe a que la forma de procesar los datos de cada uno es totalmente diferente, es decir, la MLP no tiene una forma de visualizar los datos de forma secuencial, pues lo percibe de forma instantánea para todos los datos de entrada, en contraste con la CNN, que permite un control en el recorrido de los filtros por los datos de entrada y también tiene una alta permeabilidad a la traslación de los puntos

en imágenes. Adicionalmente, es importante tener en cuenta que la susceptibilidad de la MLP a la lejanía o cercanía del usuario en el vídeo puede llegar a afectar el rendimiento de la misma, ya que el dataset de la red está compuesto de puntos proporcionales a la imagen y las variaciones que estos puedan tener debido a la distancia de grabación afecta considerablemente el análisis de los datos en la red, debido a que pierden correlación, caso contrario a la CNN, ya que se puede "visualizar" el movimiento real.

Respecto a la mejora de los bailarines que se puedan beneficiar del proyecto, el cambio en su rendimiento se va a demostrar a través de una evaluación de forma progresiva, es decir, deben haber múltiples grabaciones en el tiempo para presentarse a las RNA. Como se observa, se tienen 4 neuronas de salida, cada una equivalente a un ritmo (Bachata, Chachachá, Merengue y Salsa), con ayuda de la función de activación Softmax a la salida, se evidencia cuál de las etiquetas presenta mayor magnitud, ya que la información que brinda la función de activación es la probabilidad de pertenecer a esa clase, y entre más alta sea, hay mayor probabilidad de que la secuencia de pasos ejecutados por el practicante sean de una determinada clase. En otras palabras, un practicante debe realizar varios videos a medida que realiza correcciones y presentarle cada uno para saber si existe o no un avance respecto al vídeo anterior.

Igualmente, en cuestiones de baile, existen movimientos específicos que son característicos de cada ritmo, significando que la clasificación errónea de la red del ritmo bailado con respecto al identificado, puede indicar, por ejemplo, la ejecución de un movimiento de forma indebida o un tempo mal ejecutado, lo que le permitiría al bailarín aprendiz mejorar su técnica en este tipo de ámbitos.

BIBLIOGRAFÍA

ACEVEDO, Natalia. Matriz de confusión en Machine Learning. Explicado paso a paso. [En línea] Nataliaacevedo.com. 23 de octubre de 2020. [Consultado: 12 de agosto de 2021] Disponible en: https://nataliaacevedo.com/matriz-de-confusion-en-machine-learning-explicado-paso-a-paso/

ALBERICH, Jordi; GÓMEZ FONTANILLS, David y FERRER FRANQUESA, Alba. Percepción Visual. [En línea] España: Universitat Oberta de Catalunya. 2013. 66 p. [Consultado el 16 de mayo de 2020]. Disponible en: https://www.exabyteinformatica.com/uoc/Disseny_grafic/Diseno_grafico/Diseno_grafico_(Modulo_1).pdf

ALEMI, Omid, FRANÇOISE, Jules y PASQUIER, Phipippe. GrooveNet: Real-Time Music Driven Dance Movement Generation using Artificial Neural Networks. En: *Machine Learning for Creativity.* [En línea] Canadá, agosto 2017. 6 p. [Consultado: 10 de marzo de 2021]. Disponible en: https://omid.al/docs/groovenet-ml4c-2017.pdf

AMAZON. Aprendizaje automático para aplicaciones multimedia. [En línea] USA. [Consultado: 9 de septiembre de 2021] Disponible en: https://aws.amazon.com/es/media/tech/machine-learning-for-media-applications/

Anónimo. El ritmo, un elemento esencial del baile. About Español. [2019]. [Consultado el 16 de mayo de 2020]. Disponible en: https://www.aboutespanol.com/el-ritmo-un-elemento-esencial-del-baile-297914

AYUNTAMIENTO DE GRANADA. Información General: Análisis espectral. [En línea] Granada, España. (20 de mayo de 2014) [Consultado 20 de agosto de 2021] Disponible en: https://www.granada.org/inet/sonidos.nsf/d483b298c3f6a1b9c1257cdd00384c53/3fdfc36a7489b607c1257cde0024bb34!OpenDocument

ÁZUCAR COMPAÑÍA ARTÍSTICA & ESCUELA DE BAILE. [vídeo] Santiago de Cali. Facebook. Azúcar compañía artística & escuela de baile. (29 de marzo de 2020). 0:11 minutos. [Consultado: 17 de mayo de 2020]. Disponible en: https://www.facebook.com/azucarescueladebaile/videos/2533679163403777/

BARRIOS ARCE, Juan I. La matriz de confusión y sus métricas. [En línea] Health Big Data. 2019. [Consultado: 12 de agosto de 2021] Disponible en: https://www.juanbarrios.com/la-matriz-de-confusion-y-sus-metricas/

BERZAL, Fernando. Entrenamiento de redes neuronales. [diapositivas] Universidad de Granada. Granada, España. 37 diapositivas. [Consultado 20 de agosto de 2021]. Disponible en: https://elvex.ugr.es/decsai/deep-learning/slides/NN4%20Training.pdf

BUDUMA, Nikhil. Fundamentals of Deep Learning. 1ra Edición. USA: Sebastopol. O'Reilly. 2017. ISBN: 9781491925614

CAO, ZHE, et al. Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. En: CVPR. [En línea] USA. 2017. [Consultado: 25 de octubre de 2021]. Disponible en: https://arxiv.org/pdf/1611.08050.pdf

CHAN, Caroline, *et al*. Everybody Dance Now. 1ra USA. ICCV. 2019. [Consultado: 10 de marzo de 2021]. Disponible en: https://arxiv.org/pdf/1808.07371.pdf

COOPERATIVE INSTITUTE FOR METEOLOGICAL SATELLITE STUDIES. What is Matlab? [En línea] University of Wisconsin – Madison. CIMSS. [Consultado: 2 de julio de 2021] Disponible en: https://cimss.ssec.wisc.edu/wxwise/class/aos340/spr00/whatismatlab.htm

CUESTAS, Eduardo. Variables. En: *Revista de la Facultad de Ciencias Médicas.* [En línea] Argentina: Universidad Nacional de Córdoba, agosto-septiembre 2009. vol. 66, nro. 3. p. 118 – 122. [Consultado: 15 de agosto de 2021] Disponible en: http://www.revista2.fcm.unc.edu.ar/Rev.2009.3/Variables_Cuesta.pdf

CRNKOVIC-FRISS, Luka y CRNKOVIC-FRISS, Louise. Generative Choreography using Deep Learning . USA. 7th International Conference on Computacional Creativity. 2016. [Consultado: 10 de marzo de 2021]. Disponible en: https://arxiv.org/ftp/arxiv/papers/1605/1605.06921.pdf

DONAHUE, Chris, LIPTON, Zachary y MCAULEY, Julian. Dance Dance Convolution. [En línea] Cornell University . USA. ICML. Conference paper. 2017. [Consultado: 10 de marzo de 2021]. Disponible en: https://arxiv.org/pdf/1703.06891.pdf

EREÑO Álvarez, C. El proceso de enseñanza-aprendizaje en la danza tradicional. Alternativas metodológicas. Reflexiones desde la propia práctica [En línea] Lecturas: Educación Física y Deportes. No. 46. Buenos Aires. 2002. [Consultado el 16 de mayo de 2020]. Disponible en: https://www.efdeportes.com/efd46/baile.htm

FOMINAYA, Carlota. Velocidad de procesamiento ¿Qué es y cómo influye en tu hijo? [En línea] En: ABC Educación. España. 2019. [Consultado el 16 de mayo de 2020]. Disponible en: https://www.abc.es/familia/educacion/abci-velocidad-procesamiento-y-como-influye-inteligencia-hijo-201903250148_noticia.html

FRANCO GARCÍA, Angel. Interpolación. [En línea] Universidad del País Vasco. País Vasco. [Consultado: 13 de mayo de 2021] Disponible en: http://www.sc.ehu.es/sbweb/fisica3/datos/regresion/interpolacion.html

GAMBOA DE BUEN, Berta. El uso de la palabra promedio y su significado matemático. [En línea] Ciencia y Desarrollo. México [Consultado 10 de agosto de 2021] Disponible en: https://www.cyd.conacyt.gob.mx/?p=articulo&id=127

GARCIA FERNANDEZ, Luis Alberto. Usos y aplicaciones de la inteligencia artificial. En: La Ciencia y el Hombre. México. 2004. [Consultado el 15 de mayo de 2020]. Disponible en: https://www.uv.mx/cienciahombre/revistae/vol17num3/articulos/inteligencia/

GARDNER, Howard. La danza. En: *Revista Kinesis* Vol 2 Nº. 6. Bogotá. D. E. 1991. Res. Min. Gobierno 2113/89.

GONZALEZ LOPEZ, Francisco Javier. Desarrollo de un sistema de interacción humano-humanoide mediante el reconocimiento y aprendizaje del lenguaje corporal [en línea]. Trabajo de grado Ingeniero Mecatrónico. Santiago de Cali. Universidad Autónoma de Occidente. Facultad de Ingeniería. Departamento de Ingeniería. 2018. 54 - 64 p. [Consultado: 20 de febrero de 2020]. Disponible en: Biblioteca - Recursos Electrónicos. http://hdl.handle.net/10614/10162

GONZÁLEZ, Mónica. Bailar, ¿una profesión o una afición?. [blog]. Blogs EMagister. [Consultado 17 de mayo de 2020] Disponible en_ https://www.emagister.com/blog/bailar-una-profesion-o-una-aficion/

HAYKIN, Simon. Neural Networks and Learning Machines. 2da Edición. New Jersey: Pearson Education. 1999. ISBN: 9780131471399

HE, Kaiming, et al. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. [En línea] USA. (6 de febrero de 2015). [Consultado: 15 de julio de 2021]. Disponible en: https://arxiv.org/pdf/1502.01852v1.pdf

HSU, Hwei P. Signals and Systems. 1ra ed. USA: McGraw-Hill. 1995. 470 p. ISBN: 0070306419.

JARAMILLO ECHEVERRY, Luis Guillermo y MURCIA PEÑA, Napoleón. La danza y el baile. [En línea] Lecturas: Educación Física y Deportes. No. 46. Buenos Aires. 2002. [Consultado el 14 de mayo de 2020]. Disponible en: https://www.efdeportes.com/efd46/baile.htm

KHAN ACADEMY. Standard Deviation: Calculating Step by step. [En línea] Khanacademy.org [Consultado: 13 de mayo de 2021] Disponible en: https://www.khanacademy.org/math/statistics-probability/summarizing-quantitative-data/variance-standard-deviation-population/a/calculating-standard-deviation-step-by-step#:~:text=Step%201%3A%20Find%20the%20mean,the%20number%20of%20data%20points.

KERAS. About Keras. [En línea] Keras.io. [Consultado: 28 de octubre de 2021]. Disponible en: https://keras.io/about/

LEXICO BY OXFORD. Coreografía. [en línea] Oxford Languages. [Consultado: 20 de septiembre de 2021] Disponible en: https://www.lexico.com/es/definicion/coreografia

--------, Dato. [en línea] Oxford Languages. [Consultado: 20 de septiembre de 2021]. Disponible en: https://www.lexico.com/es/definicion/dato

MARTIN, Eugenia. ¡Conoce qué es un fotograma y cómo están hechas las películas! [en línea] Crehana. (9 de marzo de 2021) [Consultado: 20 de agosto de 2021] Disponible en: https://www.crehana.com/co/blog/video/que-es-un-fotograma/

MATLAB. Anonymous Functions. [en línea] Mathworks. [Consultado 2 de julio de 2021] Disponible en: https://www.mathworks.com/help/matlab/matlab_prog/anonymous-functions.html

--------. Creating, Concatening and Expanding Matrices. [En línea] Mathworks [Consultado 2 de julio de 2021] Disponible en: https://la.mathworks.com/help/matlab/math/creating-and-concatenating-matrices.html?lang=en

--------. El lenguaje del cálculo técnico. [En línea]. Mathworks. [Consultado: 28 de octubre de 2021] Disponible en: https://es.mathworks.com/products/matlab.html

--------. MATLAB Product Description. [En línea]. Mathworks. [Consultado: 28 de octubre de 2021] Disponible en: https://es.mathworks.com/help/matlab/learn_matlab/product-description.html?lang=en

MATPLOTLIB DEVELOPMENT TEAM. Matplotlib: Python plotting. [En línea]. Matplotlib.org. 13 de agosto de 2021. [Consultado: 28 de octubre de 2021]. Disponible en: https://matplotlib.org/

MICROSOFT. Crear o editar archivos .csv para importarlos a Outlook. [En línea] Support Microsoft. [Consultado: 10 de agosto de 2021] Disponible en: https://support.microsoft.com/es-es/office/crear-o-editar-archivos-csv-para-importarlos-a-outlook-4518d70d-8fe9-46ad-94fa-1494247193c7

MGM RESORTS INTERNATIONAL. The Truth About Entertainment [infografía]. Disponible en: https://www.mgmresorts.com/content/dam/MGM/corporate/corporate-initiatives/welcome-to-the-show/mgm-resorts-truth-about-entertainment-infographic.pdf

MISHRA, Abhishek. Data Exploration and Preprocessing. Machine Learning for iOS Developers. [en línea] John Wiley & Sons, Ltd. 2020.[Consultado 30 de agosto de 2021] Capítulo 3. Disponible en https://onlinelibrary.wiley.com/doi/epub/10.1002/9781119602927

NUMPY. What is NumPy?. [En línea] The NumPy community. 22 de junio de 2021. [Consultado: 28 de octubre de 2021] Disponible en: https://numpy.org/doc/stable/user/whatisnumpy.html

OPENCV. OpenCV: Introduction. [En línea]. OpenCV. 9 de octubre de 2021. [Consultado: 28 de octubre de 2021]. Disponible en: https://docs.opencv.org/4.5.4/d1/dfb/intro.html

PATEL, Meher Krishna. 1. Pandas Basic. [En línea] pandasguide.readthedocs.io [Consultado: 28 de octubre de 2021]. Disponible en: https://pandasguide.readthedocs.io/en/latest/Pandas/basic.html

PYTHON. What is Python?: Executive Summary. [En línea] Python.org. [Consultado: 10 de agosto de 2021] Disponible en: https://www.python.org/doc/essays/blurb/

PYTHON SOFTWARE FOUNDATION. Os – Miscellaneous operating system interfaces. [En línea] Python.org. [Consultado: 28 de octubre de 2021]. Disponible en: https://docs.python.org/3/library/os.html

--------. .Graphical User Interfaces with Tk. [En línea] Python.org. [Consultado: 28 de octubre de 2021]. Disponible en: https://docs.python.org/3/library/tk.html

RAMIREZ MORENO, David F. y HURTADO LOPEZ, Julián. Modelamiento y simulación de circuitos sinápticos sensoriomotores: Introducción a la neurobiología computacional. 1ra Edición. Colombia: Universidad Autónoma de Occidente. 2014. 152 p. ISBN: 9789588713571

ROVAI, Marcelo. Realtime Multiple Person 2D Pose Estimation using TensorFlow 2x. [En línea] Towards data science. 3 de Agosto de 2020. [Consultado: 7 de septiembre de 2020] Disponible en: https://towardsdatascience.com/realtime-multiple-person-2d-pose-estimation-using-tensorflow2-x-93e4c156d45f

SCIKIT-LEARN DEVELOPERS. 3.3 Metrics and scoring: quantifying the quality of predictions. [En línea]. Scikit-learn.org. [Consultado: 28 de octubre de 2021]. Disponible en: https://scikit-learn.org/stable/modules/model_evaluation.html

SOLAI, Pavithra. Convolutions and Backpropagations. [En línea] Medium.com. 19 de marzo de 2018. [Consultado: 5 de mayo de 2021]. Disponible en: https://medium.com/@pavisj/convolutions-and-backpropagations-46026a8f5d2c

STANFORD UNIVERSITY. Unsupervised Feature Learning and Deep Learning Tutorial. [En línea] Deep Learning, Computer Science Department. [Consultado: 26 de julio de 2021] Disponible en: http://ufldl.stanford.edu/tutorial/supervised/ConvolutionalNeuralNetwork/

TENSORFLOW. Pose estimation. [En línea] Tensorflow.org. [Consultado: 28 de octubre de 2021]. Disponible en: https://www.tensorflow.org/lite/examples/pose_estimation/overview

VARIZANI, JYOTI. 10 Real World Examples of Deep Learning Models & AI. [En línea] Futran Solutions. [Consultado: 8 de septiembre de 2021]. Disponible en: https://www.futransolutions.com/10-real-world-examples-of-deep-learning-models-ai/

YADAV, Deepak. Posture Detection using tf-pose-estimation. [En línea] Medium.com. 22 de octubre de 2019. [Consultado: 6 de julio de 2020] Disponible en: https://medium.com/@deepakec1031/posture-detection-using-tf-pose-estimation-fdad6197303b

YALTA, Nelson, *et al*. Weakly – Supervised Deep Recurrent Neural Networks for Basic Dance Step Generation. [En línea] IJCNN. 2019. [Consultado: 11 de mayo de 2021] Disponible en: https://arxiv.org/pdf/1807.01126.pdf

WASKOM, Michael. An introduction to seaborn. [En línea]. Seaborn.pydata.org. [Consultado: 28 de octubre de 2021]. Disponible en: https://seaborn.pydata.org/introduction.html

ZHOU, Victor. CNNs, Part 1: An Introduction to Convolutional Neural Network. [En línea] Victorzhou.com. 22 de mayo de 2019. [Consultado: 4 de mayo de 2021] Disponible en: https://victorzhou.com/blog/intro-to-cnns-part-1/

--------, --------. CNNs, Part 2: Training a Convolutional Neural Network. [En línea] Victorzhou.com. 29 de mayo de 2019. [Consultado: 4 de mayo de 2021] Disponible en: https://victorzhou.com/blog/intro-to-cnns-part-2/

ANEXOS

Anexo A. Código procesamiento de imágenes (ver en archivos adjunto).

Anexo B. Código función de interpolación y desviación estándar (ver en archivos adjunto).

Anexo C. Código generación dataset de convolucional (Ver en archivos adjunto).

Anexo D. Código MLP Matlab (ver en archivos adjunto).

Anexo E. Código División de datos MLP Matlab (ver en archivos adjunto).

Anexo F. Código Inicialización MLP Matlab (ver en archivos adjunto).

Anexo G. Código Funciones de activación Matlab (ver en archivos adjunto).

Anexo H. Código entrenamiento MLP Matlab (ver en archivos adjunto).

Anexo I. Código Feedforward MLP Matlab (ver en archivos adjunto).

Anexo J. Código Backward MLP Matlab (ver en archivos adjunto).

Anexo K. Código MLP Keras (ver en archivos adjunto).

Anexo L. Código CNN Keras (ver en archivos adjunto).

Anexo M. Links de vídeos extraídos de internet (ver en archivos adjunto).