Avatar virtual con inteligencia artificial y algoritmos de aprendizaje automático en aplicaciones deportivas

Luis Andres Aguilar Luhrs

2022

Universidad Autónoma de Querétaro

Facultad de Ingenieria

Avatar virtual con inteligencia artificial y algoritmos de aprendizaje automático en aplicaciones deportivas

Tesis

Que como parte de los requisitos para obtener el Grado de

Maestro en Ciencias en Inteligencia Artificial

Presenta

Luis Andres Aguilar Luhrs

Dirigido por:

Dr. András Takács

Querétaro, Qro. a 27/05/2022

Universidad Autónoma de Querétaro

Facultad de ingeniería

Maestría en ciencias en inteligencia artificial

Título del tema de tesis registrado

Avatar virtual con inteligencia artificial y algoritmos de aprendizaje automático en aplicaciones deportivas

Que como parte de los requisitos para obtener el grado de Maestro en ciencias en inteligencia artificial

Presenta

Luis Andres Aguilar Luhrs

Dirigido por:

Dr. András Takács

Dr. Saúl Tovar Arriaga
Secretario
Dr. Jesús Carlos Pedraza Ortega
Vocal
Dr. Juan Manuel Ramos Arreguín
Sinodal
Dr. Edgar Alejandro Rivas Araiza
Sinodal

Centro Universitario, Querétaro, Qro.
27 de Mayo de 2022
México

A mis padres, Mi hermano, Paola y a Pepe.

Agradecimientos

Agradezco a la Universidad Autónoma de Querétaro, a mis profesores, asesores y a mis compañeros por todo el apoyo brindado, Así como al CONACYT por el apoyo económico brindado mediante del programa de becas nacionales.

Índice

Índice de Figuras

Resumen

En el presente trabajo se propone un modelo que permita la generación de secuencias de poses humanas, representadas por puntos en el espacio con base en algoritmos de inteligencia artificial, por esta razón, se emplearán principalmente, redes neuronales. Este modelo está enfocado en la emulación del cuerpo humano, mientras este realiza actividades físicas. las limitaciones de este proyecto están en la cantidad de secuencias que se puedan obtener, el poder computacional y la precisión con la que el equipo de captura pueda obtener una representación adecuada de los movimientos de los individuos grabados.

Palabras clave: Síntesis de movimiento, Red generativa adversarial, Aprendizaje profundo.

Abstract

This work proposes a model which allows for the generation of human pose sequences, represented by points in the 3D space based on artificial intelligence algorithms, for which the vast implementation will be made on neural networks. This model is focused on the emulation of the human body while it is performing physical activities. The limitations of this project rely on the number of sequences that can be obtained, computational power and the motion capture device accuracy.

Keywords: Motion synthesis, Generative adversarial network, Deep Learning.

I. Introducción

Aproximadamente en el año 502 D.C., Bodhidharma viajó de Persia a China y llegó a Guangzhou [1] según se menciona, en los escritos “Tao-hsiian's Continued Biographies of Eminent Monks”, dónde enseñó a un puñado de estudiantes lo que serían los fundamentos modernos de las artes marciales orientales.

Aunque las artes marciales nacieron, como su nombre lo indica, como técnicas para la guerra, hay que hacer la distinción entre actividades y técnicas, se sabe incluso que no las practicaban para la guerra sino como actividades benéficas para la salud de cuerpo y mente. Los monjes Shaolin, por ejemplo, perfeccionaron primero sus habilidades con el bastón, hasta ser considerados los mejores de China, y no fue hasta el siglo XVI que perfeccionaron sus artes marciales de mano vacía, donde empleaban hojas con los puntos vitales del cuerpo, y explicaciones de las posiciones de las manos para ejecutarlas correctamente [2].

Durante el reinado del emperador K'anghsi de la dinastía Ching (1662-1722), el Kung Fu se hizo muy fuerte en el monasterio Siu Lam (Shaolin) [3], donde aparecieron 116 figuras de piedra que fueron condensadas en una sola [4], estos muñecos se emplearon para practicar las formas del arte marcial conocido como wing chung, disminuyendo en parte la necesidad de compañeros humanos. La forma condensada del muñeco de wing chung ayudó a acelerar el aprendizaje y perfeccionar los movimientos de los practicantes de esta arte marcial, estos consisten en presentar trayectorias suaves para reflectar y contratacar oponentes más fuertes y grandes a través de trabajo de pies, esto es dar pasos de formas específicas, así como empleando la economía del movimiento para gastar poca energía, este método requiere una cantidad inmensa de práctica para poder comprenderlo.

En 1949, Ip man, exponente moderno del wing chung, migró a Hong Kong desde China y al tener que adaptarse a las necesidades de espacio, fue que cambió el material del muñeco a uno de madera haciéndolo más portátil y compacto, facilitando su transporte

y su uso para practicar. Los cambios hechos a la forma de practicar debido al uso del muñeco de madera dieron lugar a la forma “Mook Yan Jong”, la cual sigue siendo vigente hasta nuestros días.

De la misma manera, la mayor parte de los deportes de contacto emplean figuras para facilitar su entrenamiento a lo largo de su formación marcial o deportiva. Un ejemplo moderno de este tipo de aparatos es el BotBoxer [5], un aparato mecánico que simula los puntos más comunes de defensa y ataque de un peleador real con la intención de que los boxeadores mejoren sus técnicas de golpeo. Aunque usar estos elementos mecánicos facilitan la práctica con un oponente, éste no puede reproducir el movimiento de un ser humano real, además en el caso del entrenamiento en el boxeo, el 70% de las lesiones ocurren durante el entrenamiento y un 69% debido a la gran cantidad de fuerza en emplean para golpear de forma repetitiva [6].

Recientemente han aparecido múltiples métodos que les permiten a los deportistas mejorar su rendimiento sin comprometerse físicamente de la misma forma que lo hacían antes, como es el caso del rugby, el hándbol mostrado por Cathy Craig [7], usando asistencia virtual para mejorar sus capacidades, empleando el Cybermind Visette Pro HMD, un sistema de realidad virtual, así como el empleo de realidad virtual para simular entornos de combate automatizados [8], ambas ofrecen un compañero de práctica virtual realista el cual puede interactuar con el usuario a través de captura de movimiento por video y el apoyo de un coach.

Está demostrado que los sistemas de realidad virtual descritos en el párrafo anterior son capaces de mejorar [9] incluso el tiempo de reacción de los practicantes de karate según el Vienna test system de 109.75 a 91.62 ms (hasta en 18.49 ms de mejora) sin ponerlos en riesgo de sufrir lesiones como en los métodos tradicionales.

El inconveniente de todos estos sistemas es que las interacciones con el usuario son limitadas a aquellas que puedan ser programadas, ya que la manera en la que se implementan requiere de un riguroso proceso de selección e implementación que implica

grabar nuevas secuencias de poses y evitar manualmente que interfieran con las anteriores , esto produce movimientos rígidos y lineales, además, tienen problemas al generar dos avatares que interactúen entre ellos simultáneamente de una forma natural, ya que sus mallas pueden interceptarse o alejarse demasiado unas con otras, esto según lo menciona Bill Tomlinson [10].

I.I. Planteamiento de problema

Existen algunos sistemas de entrenamiento asistido para deportes individuales y de equipo, tanto de realidad virtual [8], [9], [11] como dispositivos electrónicos y mecánicos de entrenamiento [5], y aunque estos ofrecen una alternativa al entrenamiento que requiere de un asesor o compañero, no lo sustituyen en la mayor parte de los aspectos, a la vez que exponen al cuerpo a grandes cantidades de daño por impactos y desgaste físico.

Los sistemas de realidad virtual usualmente carecen de un método para extenderlos y dependen de su implementación original para funcionar, esto impide que puedan ser modificados sin pasar por un arduo proceso de retrabajo.

A su vez, ninguno de estos ofrecen movimientos realistas o comparables a los que se pueden obtener usando aprendizaje de máquina, en lugar de métodos tradicionales de captura y reproducción de acciones para responder a los movimientos del usuario de forma realista [12].

De la misma manera que ocurre con las maquinas descritas en el primer párrafo de esta sección, al momento de probar técnicas con otros practicantes del mismo deporte o de otros similares, todo tipo de lesiones pueden ocurrir, no solo debido a un brusco contacto físico, como es el caso de las artes marciales o deportes de contacto como el boxeo, sino por la naturaleza misma de las actividades físicas en grupo, altamente propensas a la ocurrencia de accidentes y errores en la ejecución de varias rutinas o técnicas en equipo.

El mejor ejemplo de esto es un combate entre dos deportistas de un deporte de contacto, el puro hecho de que estos van a combatir implica que ambos sufrirán daños en varias partes de su cuerpo, estos daños pueden ser masivos en algunos casos, como ocurre con los deportes que involucran patadas y golpes de rodilla, dos de las técnicas más letales existentes, ya que pueden dejar fuera del entrenamiento, incluso al deportista más

experimentado. Por ejemplo, la patada de un practicante de tae kwon do [1] puede aplicar hasta 5475 ± 1293 N de fuerza, más que suficiente para dejar a un hombre adulto inconsciente, especialmente en la mandíbula [2].

Además del esfuerzo físico que se ejerce al ejecutar, existe un esfuerzo psicológico [3] generador por la presión grupal o el compromiso con un profesor.

La creación de sistemas que permitan la interacción de un practicante humano y otro virtual que emplee algoritmos de aprendizaje automático suponen el siguiente paso para evitar del todo el contacto físico reduciendo las lesiones, permite crear sistemas cuyo repertorio de técnicas y movimientos pueda ser extendido con un alto grado de realismo agregando directamente movimientos nuevos, al tiempo que ofrecen el nivel de realismo requerido para tener un entrenamiento adecuado, sin una fuente presión psicológica que pueda mermar el ritmo de aprendizaje y la motivación del usuario.

I.II. Justificación

El aprendizaje de cualquier deporte siempre ha estado limitado por la existencia de compañeros para mejorar las capacidades motoras y técnicas que lo componen, así mismo de un entrenador que esté durante su práctica para poder corregir detalles o imperfecciones en su ejecución.

Aunque, usualmente no existe la posibilidad de revisar detalladamente el propio desarrollo, impidiendo que sea el mismo estudiante el que eleve su habilidad que poder percibir su propio cuerpo, también disminuye su capacidad de entender sus limitaciones, incluso haciendo el uso de espejos o grabación, muchas veces la información no le proporciona datos de forma precisa, como ángulos entre miembros o niveles de tensión en los músculos.

La perseveración de malas prácticas en la realización de algunos movimientos es difícil de percibir, incluso para los instructores, permitiendo que aparezcan malformaciones en cartílagos, contracturas y otros tipos de lesiones propias de un deportista [6], [13].

La repetición de técnicas y movimientos no es la única tarea del deportista, también es necesario que éste practique su habilidad con otras personas para poder medir su grado de eficiencia y al mismo tiempo aprender de los movimientos de otros para mejorar los propios [14].

El entrenamiento físico debe ser llevado a cabo con la menor cantidad de lesiones posibles, la seguridad debe ser la prioridad de cualquier deportista [15], el desarrollo de un sistema de entrenamiento que emplee tecnologías de compañeros virtuales y visión artificial, representa una de las opciones para que el deportista alcance este objetivo al tiempo que mejora su ritmo de aprendizaje, una técnica correcta reduce la posibilidad de sufrir daños [11] y eleva el crecimiento del rendimiento físico al reducir el tiempo que se está lesionado.

Las limitaciones de la práctica del deporte en ausencia de algún método que evalúe la actividad más básica de los deportes, que es el movimiento, convierte el aprendizaje en un proceso largo y difícil, más aún para adquirir una técnica menos propensa a producir lesiones a largo plazo. Los hechos mencionados anteriormente se acentúan de la necesidad de tener un compañero para poder ejecutar algunas técnicas o prácticas que son comunes sin más guía que la retro alimentación visual de un maestro un compañero, un espejo y tal vez un video con recomendaciones.

I.III. Hipótesis

Mediante la creación un algoritmo que, empleando aprendizaje de máquina, al recibir una secuencia de poses correspondientes al movimiento capturado de un ser humano en un contexto deportivo específico, se puede generar una respuesta apropiada y semejante a la de una persona que respondiera dicho movimiento con uno propio en forma de secuencia de poses.

I.IV. Objetivos

I.IV.1. Objetivo general

Crear un algoritmo basado en aprendizaje automático que reciba una secuencia de poses correspondientes al movimiento capturado de un ser humano en un contexto deportivo especifico y permita generar una respuesta en forma de poses, semejante al de una persona que actúe como su compañero deportivo.

I.IV.2. Objetivos específicos

- Proponer e implementar un método de captura y almacenamiento para secuencias de poses.
- Proponer un método para la limpieza de secuencias de poses para que estas se adapten a las necesidades del proyecto.
- Plantear e implementar un algoritmo que pueda ser entrenado para identificar y generar secuencias de poses con los datos disponibles.
- Plantear e implementar un método que permita renderizar las poses obtenidas por el generador de secuencias para poder visualizarlas.
- Establecer un método para mostrar, evaluar y cuantificar los resultados entregados por el algoritmo generador.

I.V. Alcances

Generar un sistema que permita a un esqueleto humanoide virtual responder satisfactoriamente a los movimientos efectuados por un sujeto grabado con un dispositivo de captura de movimiento, empleando aprendizaje automático que pueda ser extendido posteriormente.

Las posibles aplicaciones van desde un asistente de practica virtual, hasta la simulación de múltiples individuos en prácticas deportivas, hasta la posibilidad de modelar interacciones cotidianas siempre y cuando las poses capturadas contengan interacciones con suficiente variedad.

I.VI. Organización de la tesis

La tesis se compone de las secciones enlistadas a continuación con su respectiva descripción:

- El estado del arte donde se describen los trabajaos más recientes, su relación y sus diferencias con el método propuesto en esta tesis.
- El marco teórico donde se describen todos los conocimientos que dan forma a la propuesta de esta tesis.
- La metodología, donde se describen los pasos a seguir para cumplir con los objetivos propuestos.
- La sección de resultados donde se pueden observar el producto de los experimentos que se llevaron a cabo para el desarrollo de la tesis para determinar si la hipótesis propuesta es verdadera o falsa.
- La sección de discusión donde se argumenta sobre los resultados obtenidos durante el desarrollo de la tesis.

II. Estado del Arte

Existen varios trabajos de síntesis del movimiento humano. Algunos de ellos utilizan los niveles de actividad a lo largo del tiempo para emular los movimientos de un ser humano mientras está en una conversación [16], el habla humana para la generación de gestos de la parte superior del cuerpo [17]. Otros métodos calculan el ritmo obtenido de la música para crear secuencias de baile [18], texto y audio para crear movimientos faciales [19]. La mayoría de estos enfoques usan un modelos articulados 2D como el del esqueleto del trabajo OpenPose [20], [21] para la captura y generación de movimiento y luego se limitan a una proyección 2D de sus poses resultantes.

Existen técnicas que si emplean esqueletos 3D, y que llevan a cabo la generación de secuencia de pose basada en la transferencia de estilo usando mezclas de modelos autorregresivos (MAR) [22]. También es posible crear variación de secuencias de movimiento humano utilizando Dynamic Bayesian Networks [23]. Modelos jerárquicos ocultos de Márkov para generación de poses y modelado de movimiento [24], también existe un trabajo que toma una trayectoria para generar una secuencia factible de movimientos a medida que se mueve a través de un entorno dado [25].

Un par de trabajos guardan bastante similitud con el presentado en esta investigación, el primero usa un enfoque adversarial bayesiano para la síntesis del movimiento humano [26], el otro trabajo que se centró en el uso de pseudo imágenes de los valores de los ángulos en las articulaciones del esqueleto para generar movimientos a partir de un espacio latente, utilizando una GAN convolucional clásica [27], también es similar y emplea una base de datos propia que no fue compartida.

También los trabajos que tienen como objetivo servir como herramientas de entrenamiento virtual tienen objetivos similares a los de esta tesis, uno de los más destacables es el llamado KaraKter [8], este permite crear un compañero virtual para entrenar karate y está pesando para usarse con realidad virtual, la efectividad de estos sistemas ya se ha comprobado anteriormente [9] y es una buena alternativa al trabajo

con compañeros reales. Existen algunos otros sistemas de entrenamiento [11] aunque la mayor parte de estos se dedican a entrenar los reflejos humanos y la destreza visual más que la técnica.

Un modelo que ofrece estas prestaciones empleando aprendizaje de máquina llamado Neural State Machine, emplea una red neuronal experta que permite generar transiciones automáticas entre dos poses dado un objetivo concreto cambiando entre redes suavemente dependiendo de la situación del ambiente y el avatar para después hacer estimación de poses, como sentarse o pasar por un agujero, con un error posicional inferior a los 3 cm, un esquema que le permite agregar más acciones sin la necesidad de incrementar enormemente los datos requeridos [28], empleando un control bidireccional que usa sensores virtuales de forma programática en el avatar y los elementos del ambiente.

Aunque la correcta interacción con diversas superficies depende de la disponibilidad de una acción similar grabada como secuencia de poses en la base de datos, que actúe con una similaridad razonable, además de que un cambio brusco de objetivos puede traer movimientos muy abruptos, de forma homologa K. Kania [25] propuso un modelo con el mismo objetivo, pero en este caso empleando un auto-encoder variacional que le permitiera generar una secuencia de poses coherentes en tiempo para un Avatar virtual pero este solo interactuaría con un espacio plano y sin obstáculos u objetos con los que pudiera interactuar en general.

Existe también métodos que emplean otro tipo de datos de entrada con la intención de crear poses en diferentes contextos, entre los más destacables se encuentra el trabajo de Nishimura [16] que emplea la intensidad de la interacción como entrada de una red neuronal convolucional como generador para una red GAN, consiguiendo que esta genere los movimientos de la parte superior de un avatar virtual con la intención de emular una conversación real sin dialogo.

Otro algoritmo que trabaja sobre el área de la emulación de la conversación humana es el trabajo de Bowen Wu, este consiste en tomar muestras de audio y los gestos que se ejecutan durante las conversaciones y asignarlos mutuamente empleando la técnica GAN, de esta forma consiguiendo que el generador pueda producir movimientos a partir de frases o diálogos que ocurran durante una conversación, Propuesto por Lee et al. en conjunto con Nvidia en la publicación “Dancing to music” [18], de forma similar al trabajo de Bowen se mapean sonidos a los movimientos de una sube de puntos que representa a una persona, sin embargo esta vez se emplea el ritmo de los sonidos para hacer que los movimientos producidos emulen el baile de una persona, este método también emplea la técnica de GAN.

También hay técnicas que emplean varios tipos de entrada con la intención de hacer más rica la cantidad de posibles movimientos generados por los modelos empleados, un ejemplo es el trabajo de Lingyun Yu [19], donde se emplearon redes neuronales convolucionales profundas para producir los movimientos de las articulaciones de un avatar virtual mediante el uso de texto y audio, consiguiendo que los resultados sean coherentes en el tiempo y más precisos que sus predecesores alcanzando un error cuadrático medio de 0.563mm.

En un método propuesto por Wang Xi [29] se empleó nuevamente el método GAN para entrenar una red generadora para producir secuencias de movimientos en forma de pseudo imágenes, esto es imágenes que representan el ángulo de las articulaciones del avatar virtual en el tiempo, donde las filas representan pasos en el tiempo y las columnas cada ángulo para cada articulación, de esta manera consigue construir un espacio latente que codifica da una de las secuencias de poses existentes en la base de datos sobre la que se ejecutó el algoritmo de entrenamiento, de forma similar pero empleando redes bayesianas adversariales, R. Zhao [26] para modelar la distribución de los movimientos humanos usando la base de datos CMU Motion Capture [30].

De los métodos como el Neural State Machine pueden beneficiarse sistemas que usan avatares virtuales en sistemas de realidad virtual como guías para ejecutar la apropiada

locomoción de movimientos en esqueletos virtuales para ser usados como compañero de entrenamiento, incluso cuando estos sistemas ya proporcionan una mejora sustancial en la precisión espacio temporal de los usuarios, hasta de un precisión en el espacio de 20.250 deg·s−1 según Laurentius A. Meerhof et al. [31],empleando una comparación en el tiempo de la pose adecuada que debe tener la persona practicando una actividad física y la pose que está usando en la realidad [32] para después mostrarla en tiempo real o al finalizar el ejercicio, esto demostró ser efectivo en practicantes de gimnasia que obtuvieron una mejora promedio de 2.3% en el posicionamiento de sus miembros al realizar molinos en el caballo con aros justo después de usar retro alimentación auditiva, donde la mejora obtenida fue retenida durante dos semanas [33].

Aunque una de las principales debilidades de las redes neuronales es el costo computacional, existen ahora métodos que proveen de una precisión del 91.1% para la estimación de poses con video monocular con un costo computacional del 14.3% que requeriría el método más preciso según Feng Zhang [34], un sistema de este tipo puede hacer la estimación de pose usando únicamente una cámara común y corriente, en lugar de equipos de captura especializado.

En el futuro estos avances en conjunto con los del internet de las cosas permitirán que, a través de una base de datos colocada en un servidor nube dedicado, se recolecten y almacenen datos empleando equipo deportivo inteligente, así como sistemas de captura de movimiento para procesarlos y crear sugerencias personalizadas, rutinas e incluso avatares adaptados a las necesidades de cada deportista usando aprendizaje de máquinas como lo menciona [35].

De todo lo antes mencionado, aparece la necesidad de proveer un método que permita generar y representar movimientos más naturales, que tengan una mayor facilidad para interactuar con otros avatares dentro del mismo escenario sin tener que modificar el código y la base de datos de animaciones disponibles de forma significativa, a menos que el contexto de las acciones sea distinto. Las redes neuronales tienen una ventaja importante sobre los sistemas basados únicamente en poses, que consiste en él una red

una vez entrenada puede ser extendida, esto es, que puede aprender nuevos movimientos incrementando la base de datos de movimientos y entrenando la red como en el modelo propuesto por Wang [32].

No hay mucha información sobre los métodos que pueden usar poses como entrada para generar una pose nueva y no vista como respuesta, reduciendo la posibilidad de usar el factor de interacción humana basado en el lenguaje corporal que se ofrece en este trabajo. Además, los sistemas actuales que emplean redes neuronales se usan en su mayoría para crear video o imágenes a partir de poses capturadas por medio de algún sistema.

Los sistemas especializados en entrenamiento deportivo al no emplear redes neuronales pierden la posibilidad de entregar movimientos suaves y naturales, así como es difícil para estos responder a secuencias de poses no previstas en el diseño de estos, no es posible comparar directamente la mayoría de los métodos descritos ya que dependen de diferentes tipos de entradas, conjuntos de datos y, objetivos. Incluso si todos son modelos de generación de movimiento humano, estos tienen diferentes propósitos; Lo mismo sucede con este trabajo, ya que no hay otro trabajo que modele una interacción de dos sujetos utilizando datos de poses basados en puntos como entrada.

III. Marco teórico

El aprendizaje de máquina emplea redes neuronales que emulan el funcionamiento de redes neuronales biológicas, haciendo pasar la información a través de un grupo de nodos que funcionan como neuronas sintéticas, es posible procesar información para prealimentar la red y obtener una salida que es la solución a un problema de clasificación o regresión, a su vez, estas neuronas emulan la función de aprendizaje a través de la retro propagación, corrigiendo el error en cada una de ellas de acuerdo a la diferencia entre el resultado esperado y el obtenido.

Existen muchas arquitecturas y técnicas de aprendizaje, en este caso se emplearán redes adversariales para generar secuencias de puntos.

A través del uso de aprendizaje de máquina se llevará a cabo el entrenamiento de una red neuronal adversarial compuesta por una red de clasificación y una variación de una máquina de estados neuronales para la recolección de datos.

III.I. Redes neuronales adversariales

De acuerdo con Creswell et al. [36], se trata de un algoritmo de aprendizaje profundo donde se emplean dos modelos, un modelo generativo G y otro discriminativo D, donde D ayuda al mejor entrenamiento de la red G ya que D ayuda a examinar los datos generados por G mejorando su capacidad de distribuir el aprendizaje a través de toda la red. Esto significa que una red se entrena para generar datos mientras la otra se entrena para verificar que estos sean correctos, de tal manera que tengan la menor cantidad de error posible.

Aunque se menciona que ambas deben ser un par de redes neuronales simples en la práctica modela el principio de cualquier par de generador - discriminador.

Observemos el ejemplo más básico donde ambas partes D y G son redes neuronales simples.

Sabemos que D evalúa la calidad de la entrada x, ésta métrica es representada por la ecuación (1).

$$J(D,G) = \mathbb{E}_{x\sim Pdata}[logD(x)] + \mathbb{E}_{z\sim Px}\left[log\left(1 - D\big(G(z)\big)\right)\right] \quad (1)$$

El primer término de la ecuación 1 representa el valor esperado de la probabilidad de que x sea un dato real, el segundo de la ecuación 1 que el dato sea falso o sea trata de ruido o estímulos no deseados. Como el objetivo de D es el de maximizar a J y el de G el de minimizarlo, es este conflicto de objetivos el que resulta en la interacción de mínimos y máximos de la red neuronal dada por la ecuación 2.

$$\min_{G}\max_{D} J(D,G) \quad (2)$$

Para entregar una Generative Adversarial Network (abreviado GAN por sus siglas en inglés), es necesario introducir ruido a la red generativa y una imagen real a la discriminativa, de tal manera que al comprarse se estime un gradiente que sirva para actualizar los valores de los pesos en el discriminador y al mismo tiempo que los parámetros de la red discriminativa como se puede ver en la Fig. 1.

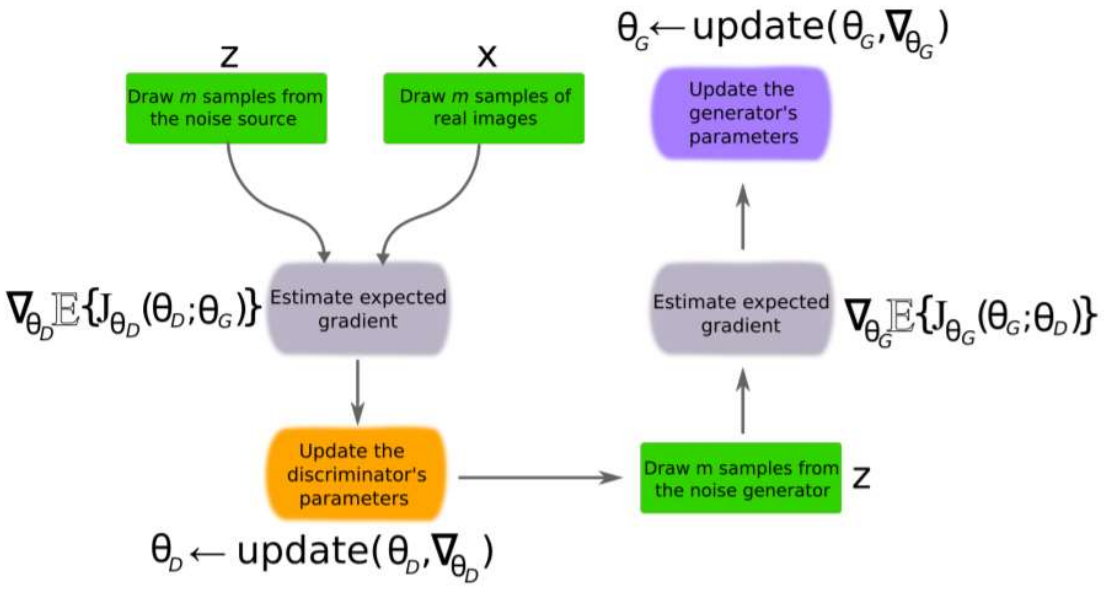

Fig. 1 Proceso de entrenamiento de una GAN [36]

Así la red discriminante intenta distinguir entre las imágenes reales y las generadas, mientras que la red generadora intenta generar imágenes lo más parecida que sea posible a las reales, de tal manera que pueda engañar a la red discriminante, sin embargo, la red discriminante sigue aprendiendo a distinguir entre las imágenes falsas y las verdaderas haciendo difícil, si no es que imposible, que este tipo de red neuronal converja en el error en un momento dado.

Otro conflicto que ocurre al usar una GAN es que únicamente permite generar patrones similares a los de la base de datos a partir del ruido entregado, esto impide que se tenga algún tipo de control sobre la salida deseada de la red generativa, así que ahora introducimos una variación de la red GAN.

III.II. Redes neuronales adversariales condicionales

Las redes neuronales adversariales condicionales o Conditional Generative Adversarial Nets (abreviado CGAN por sus siglas en inglés), permiten agregar una entrada adicional para poder obtener una entrada dada una etiqueta, esto significa extender el vector de entrada para la red discriminadora y la generadora, quedado la función objetivo como se muestra en la ecuación 3.

$$\min_{G}\max_{D} V(D,G) = \mathbb{E}_{\sim p_{data}(x)}\left[\log D(x|y) + \mathbb{E}_{z\sim p_z(z)}\left[\log\left(1 - D\big(G(z|y)\big)\right)\right]\right] \qquad (3)$$

Como se puede apreciar ahora se tienen por entradas combinadas del ruido y la etiqueta para la red generativa y la de la de la imagen real combinada con la etiqueta, de tal forma que ahora el diagrama de la red neuronal se presenta en la Fig. 2.

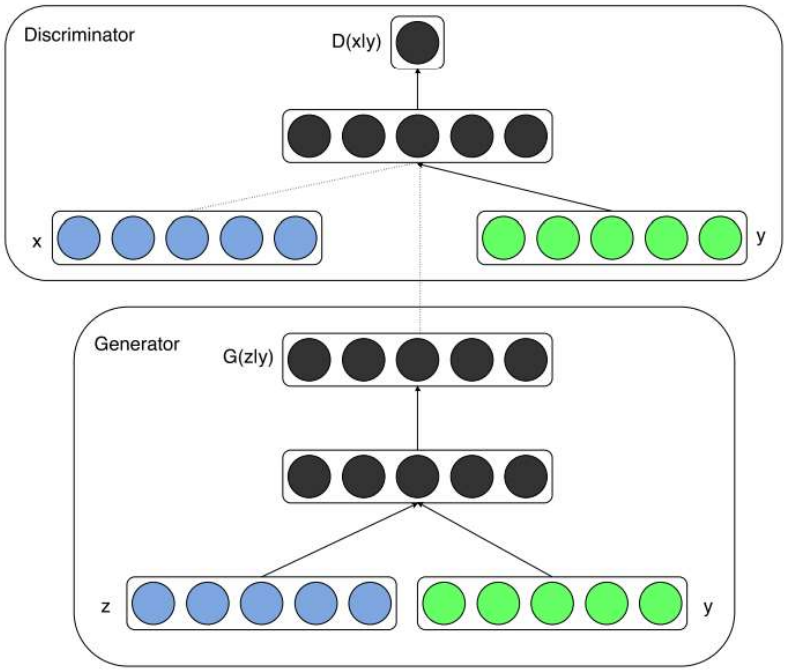

Fig. 2 Diagrama de la red CGAN con la entrada auxiliar [27]

Así al entrenar una red neuronal dado un grupo de imágenes con sus respectivas etiquetas, por ejemplo, de letras escritas a mano etiquetadas con sus respectivos valores

a través del proceso de su entrenamiento permiten generan nuevamente imágenes similares a las empleadas para entrenar la red cuando dicha etiqueta fue usada.

También es posible variar la semilla aleatoria para obtener resultados con pequeñas diferencias, como se puede ver en la Fig. 3.

Fig. 3 Imágenes generadas a partir de entrenar la red con imágenes de letras manuscritas [27].

Es natural pensar entonces que es posible usar imágenes como entradas para así emplear la red como un filtro directamente, consideremos la Fig. 4.

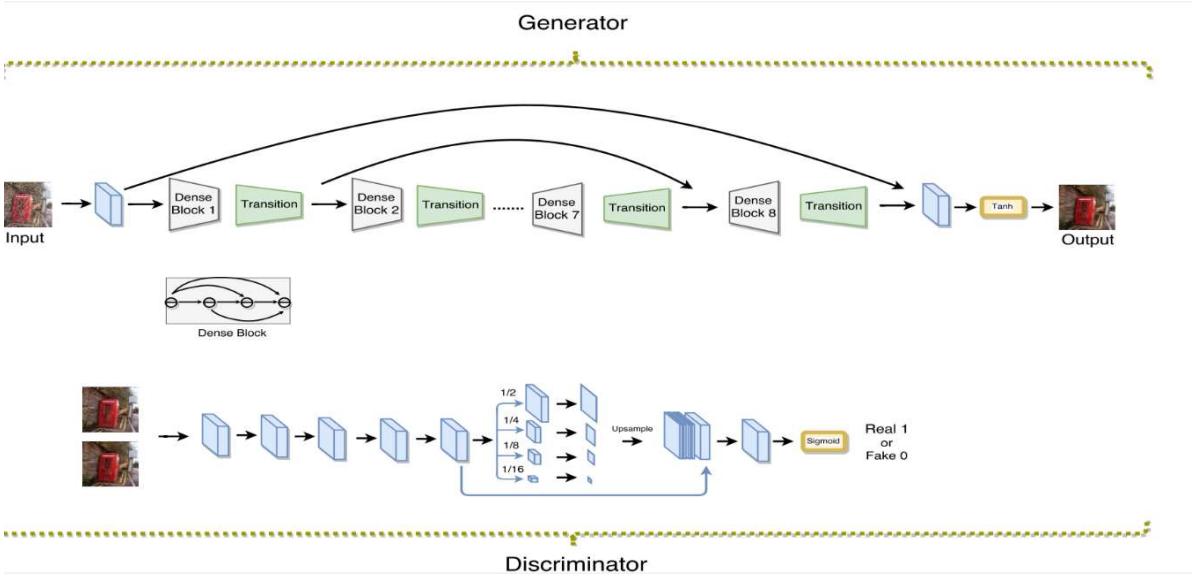

Fig. 4 Diagrama de filtro de imagen empleando CGAN [37]

Dada una imagen de entrada x, la red GAN puede aprender una función no lineal para generar una imagen de salida con el efecto deseado, como el caso en la red generada por Zhang et al [37] podemos remover la lluvia de una imagen empleado este método.

Para una red CGAN la función de error se define como la ecuación 4, donde y es la información que condiciona la salida y x la imagen de entrada.

$$\min_{G}\max_{D} V(D,G) = \mathbb{E}_{X\sim p_{data}(x)}\left[\log D(x,y) + \mathbb{E}_{x\sim p_{data}(x,y)}\left[\log\left(1 - D\left(x, G(x)\right)\right)\right]\right] \quad (4)$$

Dado que en este trabajo se pretende usar una CGAN para calcular la reacción de un esqueleto virtual haciendo uso de dos secuencias de poses que servirán como entrada x y etiqueta y, observemos las consideraciones tomadas del siguiente estudio, que emplea un modelo de CGAN para generar video de movimiento:

Empleando la misma arquitectura CGAN como la que se menciona en la ecuación 4, pero con sus componentes renombrados y empleado 3 redes diferentes para obtener el resultado creado por el método GOHAG [38]:

a) Pose generation GAN (Abreviado PGAN por sus siglas en inglés)

Esta red GAN se entrena para generar acciones en forma de poses, para entrenar a la red se introduce ruido en el generador, junto con una etiqueta de clase, tanto para la generativa como para la discriminante, la arquitectura es la que se muestra en la Fig. 5.

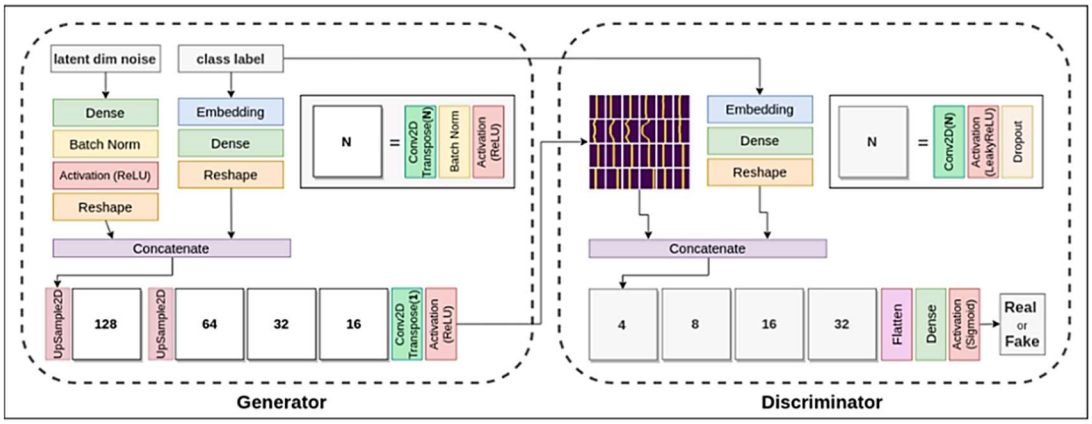

Fig. 5 Arquitectura PGAN [38].

b) Poses Optimization GAN (abreviado POGAN por sus siglas en inglés)

Esta red tiene la función de llevar a cabo la optimización de las poses entregadas por la red PGAN, permitiendo generar una salida estable. La red POGAN es una red CycleGAN condicionada, esto es una red que se acopla a otra, en este caso la PGAN e intenta limpiar las salidas empleando los puntos reales, consiguiendo así una mejor calidad en la señal de salida como se puede apreciar en la Fig. 6.

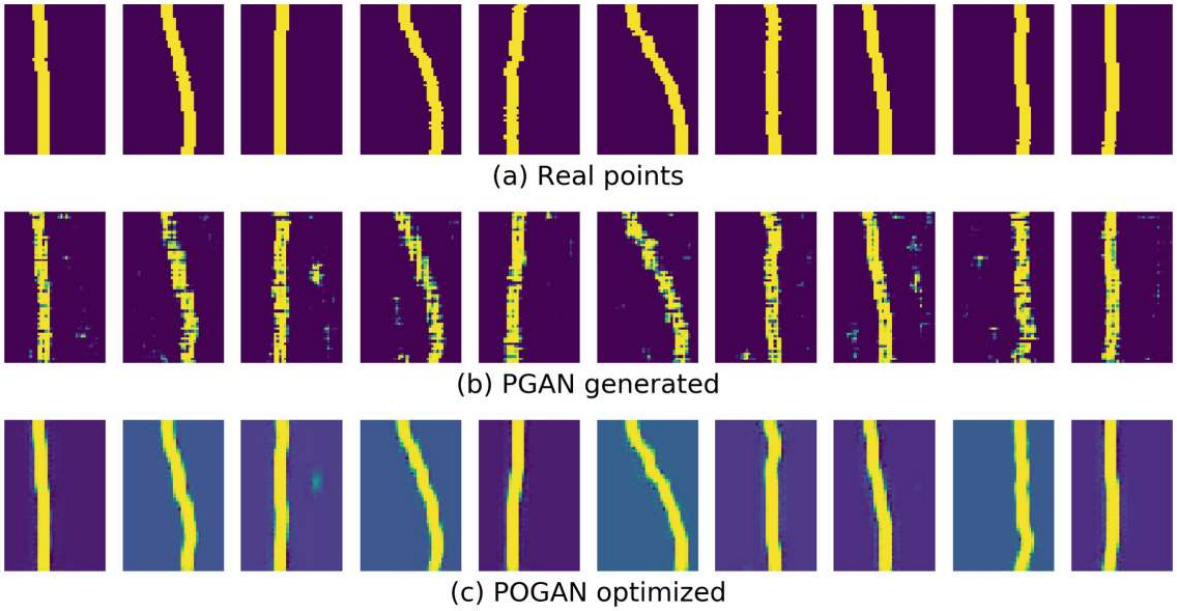

Fig. 6 a) Información original b) Información generada por la red PGAN c) Información limpiada por la POGAN en base a la información generada contra la original [38] .

c) Frame generation GAN (abreviado FGAN por sus siglas en ingles)
Esta red tiene como función generar los fotogramas correspondientes para cada pose, esto lo consigue usando las poses producidas por la red PGAN y POGAN al tiempo que las compara con las imágenes originales para así poder generar múltiples movimientos como animaciones en secuencias de fotogramas.

Juntas estas redes conforman la red GANs Orchestration for Human Actions Generation (abreviada GOHAG, por sus siglas en inglés), cuya arquitectura se muestra en la Fig. 7.

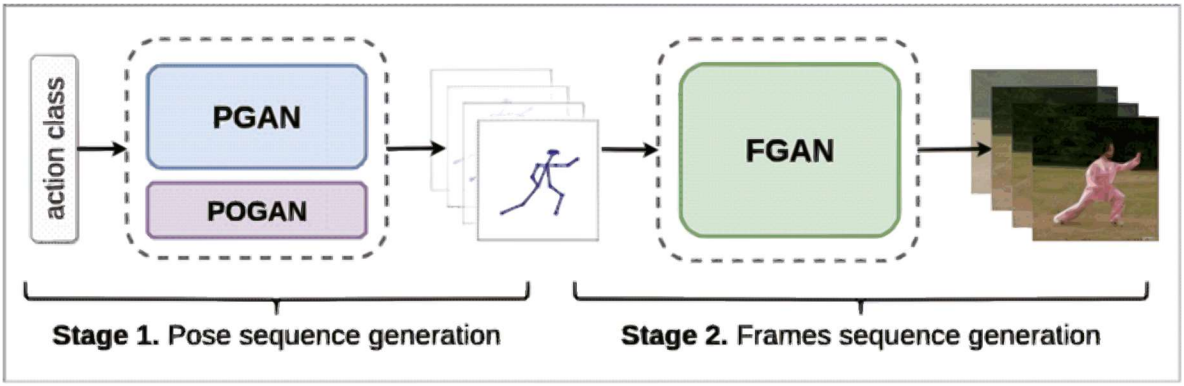

Fig. 7 Diagrama de la red GOHAG [38].

La red que se va a emplear será similar a la etapa uno del modelo GOHAG, ya que se no se requiere de la generación de poseas sea en forma de imágenes si no de esqueletos humanoides.

III.III. Recolección de poses

Evaluar y entrenar redes neuronales a través del uso de bases de datos es una tarea ardua para cualquier persona que desee hacerlo, especialmente cuando el tipo de acciones que se desean entrenar es muy específico.

Es muy importante considerar el grado de error que existe entre la información real y la obtenida por un sistema o algoritmo de captura, al mismo tiempo hay que tomar en cuenta el ruido que pueda ser generado al capturar las poses deseadas, es por eso por lo que será necesario establecer un filtrado de los puntos obtenidos.

Los datos obtenidos deben ser almacenados siguiendo un procedimiento, un método que puede ser empleado es el propuesto por altexsoft en su sitio web de ingeniería en software [39], proveé de los pasos para identificar y establecer mecanismos de recolección, crear un formato consistente y reducir la cantidad de datos de ser redundantes, este se describe a continuación:

1. Recolectar datos si estos no se poseen: es preferible emplear bases de datos abiertas en un comienzo, si esto no es posible se pueden recolectar manualmente los datos intentando que tengan un bajo nivel de complejidad a la vez que sean los menos posibles, ya que un dataset grande puede necesitar de equipo altamente especializado y potente para guardarlo o procesarlo.

2. Comprender el problema que se desea resolver: para poder recolectar datos y emplearlos adecuadamente es necesario entender que es lo que se va a hacer con ellos, para esto se debe distinguir entre las diferentes operaciones que se puede hacer con aprendizaje automático que son:

    - Clasificación
        - Algoritmos que emplean etiquetas para entregar una respuesta binaria dado un tipo de dato.

- Agrupamiento
    - Algoritmos que averiguan los grupos en los que se encuentran distribuidos un grupo de datos y los principios que los caracterizan.
- Regresión
    - Algoritmos que entregan un valor numérico dependiendo de cual fuese la entrada, pueden ser usados para estimar valores en el futuro o las acciones de un sistema.
- Ranking
    - Algoritmos que ordenan objetos de acuerdo con un número de características, muy usados a la hora de recomendar películas o mostrar productos que al usuario pudieran interesarle.

Se recomienda evitar problemas muy complejos, al tiempo que se empleen herramientas existentes de ser posible.

3. Establecer mecanismos de recolección de datos: el primer problema para tratar es la fragmentación de la información, si ésta se encuentra dispersa en diferentes sectores del área que se quiere estudiar, será necesario fragmentar la obtención de datos, esto es segmentar los esfuerzos por recopilar la información, y debe entenderse que no siempre se pueden cubrir todas las fuentes simultáneamente si existen muchas fuentes de datos, es por eso importante retener la información y hacerla digerible para el sistema que se emplee.

   La recolección de datos es un proceso tedioso y largo, entonces, es importante crear herramientas que faciliten esta tarea para evitar una sobrecarga de operaciones manuales que pudieran afectar la obtención de información.

   Por esto es útil emplear sistemas sencillos para recopilar la información, como sistemas de archivos o bases de datos relacionales como Aito [40].

4. Hacer los datos consistentes: escoger un formato para guardar los datos que se va a emplear es importante ya que permite homogeneizar los datos adquiridos,

por ejemplo, si se tienen 3 mediciones diferentes como 4, $4 y ‘cuatro’ será necesario escoger una y transformar la base de datos al tipo correspondiente.

5. Reducir los datos: puede ser tentador incluir tanta información como sea posible, pero, esta idea puede ser contraproducente. Saber qué tipo de información se quiere obtener y descartar el resto de los datos es importante, este método se llama muestreo de atributos, si se deseara predecir la siguiente compra de un usuario, su compra anterior, edad, sexo y ubicación pueden ser más útiles que su número de tarjeta de crédito, aunque debemos recordar qué valores diferentes pueden ayudarnos a descubrir dependencias ocultas, se debe ser observador en este aspecto.

   Otro método es llamado registro de muestreo, este método borra información que resultara errónea, interpolándola, reemplazándola o simplemente descartándola, también es posible usar métodos de reducción dimensional.

   Finalmente, el método de agregación consiste en crear grupos de datos que pueden estar relacionados en el tiempo, por tipo de evento o en general cualquier otro dominio en el que se encuentren, el agrupamiento puede ser útil en este caso.

6. Descomponer la información: después de reducir y clasificar los datos puede ser útil separarlos en varias partes y agruparlos de acuerdo con alguna de sus características, por ejemplo, si las ventas de un local dependen claramente del día de la semana, es posible que una mejor correlación pueda ser encontrada si se separan los días por cada día de la semana independientemente de sus fechas.

7. Normalizar la información: Normalizar los valores obtenidos es una etapa importante del procesamiento de la información, escogen un mínimo y máximo para el rango de los valores, moviendo el punto decimal, o discretizándola información redondeándola son técnicas que pueden ser usadas para este propósito.

Otro punto importante que hay que considerar es el sobre ajuste, esto sucede cuando una red neuronal es entrenada sobre un grupo de datos muy específicos, elevando el error para casos más generales, por ejemplo, entrenar una red para detectar una persona que camina sobre el pavimiento, pero no puede distinguir la acción cuando la persona camina en una superficie de otro color. Para detectar el sobre ajuste hay que evaluar el algoritmo con datos con los que no fueron entrenados, así el data set debe separarse en dos o tres partes si se va a usar un grupo de datos de validación para el entrenamiento como se muestra en la Fig. 8.

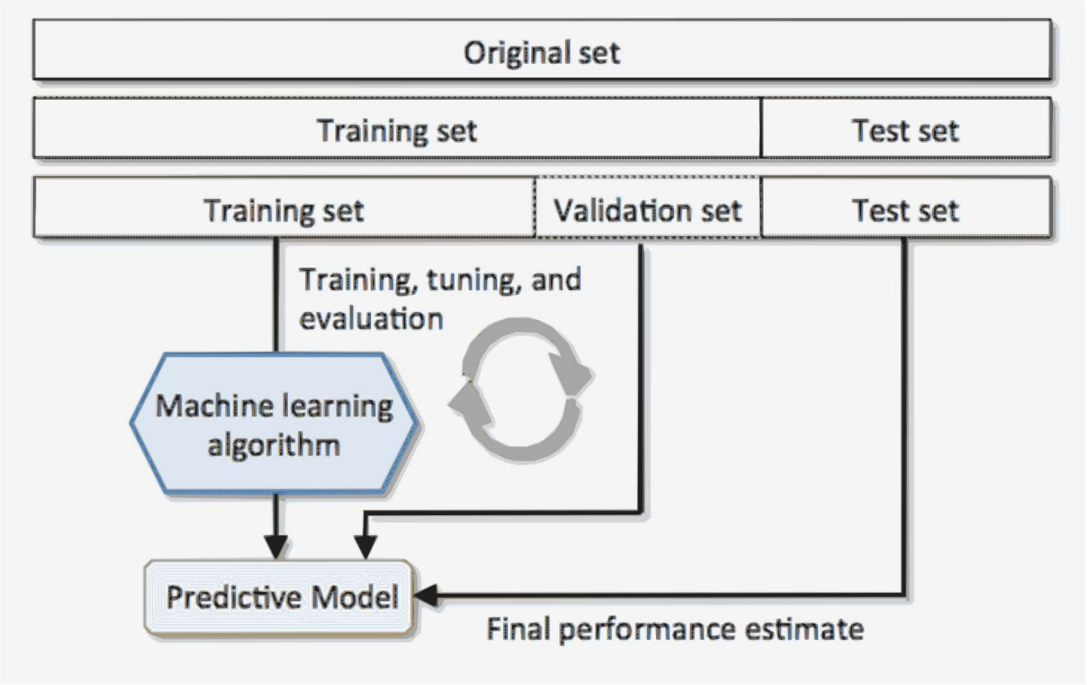

Fig. 8 Diagrama que ilustra el proceso de segmentación de datos para el entrenamiento de un modelo [41].

Por último, mencionemos las bases de datos públicas existentes, estas pueden ser útiles para entrenaran el algoritmo en aspectos más generales, mientras que los data sets privados usualmente contienen información muy específica. Se pueden encontrar data sets de excelentes prestaciones en GitHub, ya que, aunque existan otros data sets

comerciales, estos son usualmente completamente gratuitos y solo requieren mención en el trabajo en el que se van a usar.

III.III.1. Sistemas de recolección de poses

La recolección de poses es una actividad que consiste en obtener la posición o ángulo de las articulaciones en forma de puntos o miembros de un individuo en forma de segmentos, el objetivo de esta tarea es obtener una representación de una pose o secuencia de poses ejecutadas por dicho individuo, para poder apreciar o estudiarlas con más detalle posteriormente. Lo más usual es que se capturen secuencias de poses de tal forma que se obtengan secuencias animadas de movimientos, esto se puede llevar a cabo mediante varios métodos que van desde el uso de cámara digitales, radares y otros métodos que nos permitan convertir los datos recolectados en información que represente los movimientos de un sujeto en 3D, algunos métodos destacables se describen a continuación.

III.III.1.1. Sistemas ópticos

Los métodos de captura de movimiento óptico pueden llegar a ser bastante precisos cuando se emplean métodos en el estado del arte. Usualmente no son métodos que funcionen en tiempo real, aunque hay algunas excepciones [42]–[44], esto no es lo más usual por lo que no es posible tener retro alimentación mientras se recolectan las poses deseadas a menos que los movimientos sean muy simple y no existan muchos subjetos en la grabación. Estos métodos por lo general requieren de un post procesamiento extenso para poder usar la información recolectada, así que el costo computacional puede ser alto.

Un método para llevar a cabo la recolección de poses empleando sistemas ópticos es el método de captura multi cámara con marcadores, este consiste en vestir a los sujetos que estas siendo grabados con ropas que tengan marcadores de colores brillantes que después puedan ser localizados automáticamente o manualmente al revisar las

grabaciones de cada una de las cámaras, para finalmente triangular sus posición basándose en la ubicación de las cámaras una con respecto de la otra, en la Fig. 9 se puede apreciar como los marcadores están distribuidos sobre el sujeto que va a ser grabado empleando un sistema de captura, en este caso la recuperación de las posiciones de los puntos se hace de forma automática empleando el software ViconIQ [45], este fue el método que se empleó para capturar todas las pose de las base de datos de la CMU [30].

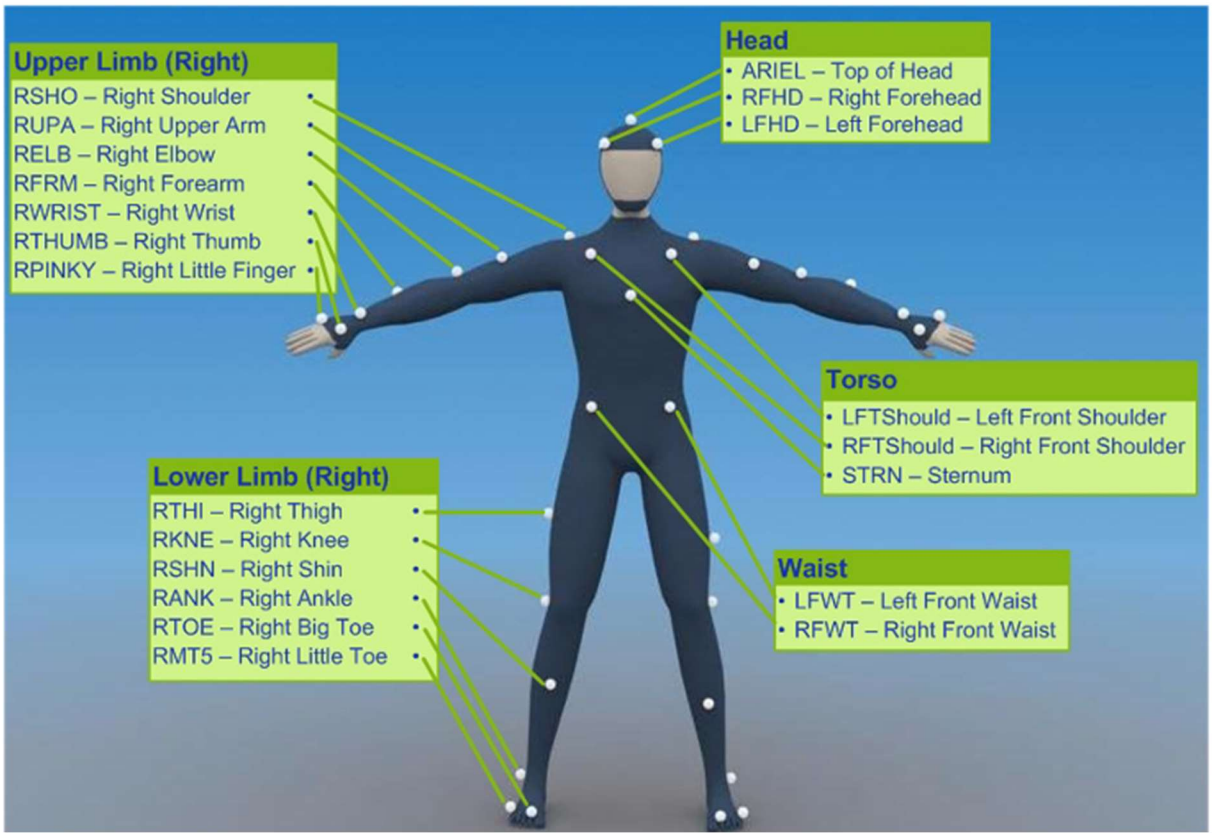

Fig. 9 Diagrama que muestra la colocación de los marcadores en un sujeto de prueba.

Otro método de captura que emplea métodos ópticos el propuesto por Microsoft durante el proyecto Natal, también denominado por su nombre comercial, y como se le denominara a partir de ahora en este trabajo, Kinect. El dispositivo de captura Kinect captura imágenes de profundidad que después son utilizadas por un bosque aleatorio para clasificar, punto por punto, la articulación que corresponde a cada una de las secciones del cuerpo del sujeto que está siendo grabado empleando bosques aleatorios [46], en la Fig. 10 el proceso a través del cual se obtienen las poses se puede apreciar

más a detalle, en el primer paso se puede ver que se obtienen la imagen de profundidad con los sensores del dispositivo Kinect, en el segundo paso se obtiene la probabilidad de que cada punto del mapa de profundidad pertenezca a una articulación en específico y finalmente en el último paso se obtienen las coordenadas tridimensionales de cada uno de los puntos a partir de la probabilidad de cada punto.

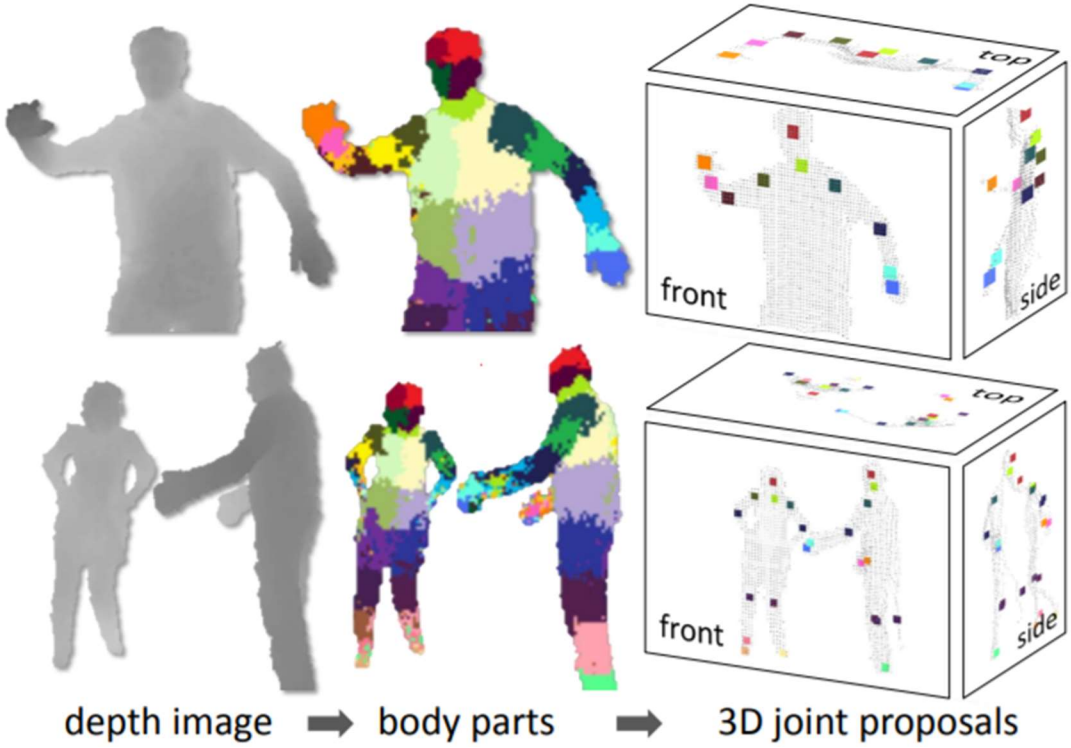

Fig. 10 Se ilustra el proceso de obtención de poses del dispositivo Kinect.

III.III.1.2. Sistemas de radiofrecuencia

Los sistemas de radiofrecuencias como su nombre lo indica, emplean equipos de radio para llevar a cabo la triangulación de un dispositivo que envía y recibe señales desde el equipo de captura. Los métodos más usuales de captura mediante radiofrecuencias se listan a continuación:

- Global Positioning System (GPS)
- Real-Time Location Systems (RTLS)
- Local positioning systems (LPS)

Estos sistemas cuentan con la ventaja de no requerir la re-identificación de sujetos después de la captura, funcionan en cualquier condición de iluminación, y pueden capturar grandes cantidades de información en tiempo real, sin embargo son sensibles a la interferencia electromagnética y no funcionan cuando hay objetos metálicos entre los sujeto que están siendo grabados y el dispositivo recepción de radiofrecuencia.

III.III.1.3. Sistemas electromagnéticos

Los sistemas de captura basados en métodos electromagnéticos consisten en arreglos de receptores que miden su distancia en relación con la intensidad del campo del sistema de medición, estos receptores están conectados a un sistema de control electrónico mediante cables individuales que les proporcionan energía directa o alterna, dependiendo del funcionamiento de cada equipo. Usualmente están compuestos de 11 a 18 receptores que puedes generar hasta 144 mediciones por segundo, las cuales son obtenidas filtrando y amplificando las mediciones del campo de cada una y calculando los ángulos entre ellas al obtener el cabeceo, inclinación y vuelco. Las mediciones de estos dispositivos pueden verse severamente afectadas por la presencia de cualquier tipo de objeto metálico en la zona de grabación, así que este es un aspecto que debe tenerse en cuenta si se emplea cualquiera sistema de este tipo.

III.III.1.4. Sistemas electromecánicos

Estos sistemas emplean una combinación de mecanismos articulados y dispositivos electrónicos como potenciómetros o giroscopios, para llevar a cabo la captura de los

ángulos de las articulaciones más cercanas a cada uno de los sensores, algunas versiones más modernas emplean sensores inerciales para estimar la posición de los miembros del individuo sobre el cual se efectúa la captura de movimiento, a este tipo de equipos se denomina MEMS [47]. Aunque este tipo de equipos tiene la gran ventaja de poder una gran variedad de movimientos en un amplio rango de acción, es difícil o imposible para estos grabar movimientos globales, por lo que sus capacidades están limitadas a movimientos centrados en el origen del traje, que puede ser la cadera o torso.

III.IV. Medición del error y métricas

Existen múltiples funciones de error, esto se hace con la intención de encontrar condiciones suficientes para la minimización del riesgo empírico, de hecho, el rol que juegan las funciones de error es bastante marginal, y la elección de qué tipo de perdida se va a emplear depende del problema computacional que se va a resolver [48], y algunas veces los resultados son difíciles de interpretar.

En general se dividen en 2 tipos dependiendo de su uso, clasificación y regresión, Algunas funciones de error usadas comúnmente en clasificación son:

- Entropía cruzada binaria [49]
- Divergencia de Kullback-Leibler [50]
- Probabilidad logarítmica negativa [50]

Este tipo de funciones de error pueden caer en otros 2 tipos, clasificación, donde se busca una clase con la mayor probabilidad y estimación de probabilidad de clase, de tal forma que se aproxima la probabilidad condicional de la salida de un modelo, la común de estas en aprendizaje de maquina es la clasificación [50].

Las funciones de error para regresión tienen la función de estimar el valor de una función desconocida, mediante la distancia entre el valor real y valor estimado, algunas funciones de costo comúnmente usadas para este tipo de problema se listan a continuación [51]:

- Error cuadrático medio
- Error absoluto medio
- Error absoluto medio porcentual
- Similaridad de coseno

Estas funciones de error representan distancias entre 2 vectores la cual se desea minimizar.

En este caso se emplearán 3 funciones de costo:

- Error cuadrático medio
- entropía binaria cruzada
- entropía binaria categórica

III.IV.1. Error cuadrático medio

El error cuadrático medio es una métrica que sirve para evaluar modelos, usada usualmente en modelo de regresión. El significado del error cuadrático medio de un modelo con respecto al conjunto de prueba, esto es las muestras que se pretende imitar, es la media de los cuadrados de la diferencia entre el valor obtenido y el valor del conjunto de prueba para todas las instancias correspondientes, como se muestra en la ecuación 9.

$$\mathrm{ECM} = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y_i} - Y_i)^2 \tag{9}$$

En este caso, el conjunto de prueba está representado por las coordenadas de las articulaciones obtenidas mediante el uso de la red neuronal y las de la base de datos de prueba.

III.IV.2. Entropía binaria cruzada

La entropía binaria cruzada es un caso particular de la métrica de entropía, la cual es una medida de incertidumbre para una variable discreta con 2 estados, esta forma especial está definida como lo muestra la ecuación 10 [49]:

$$H(x) \equiv x\log\frac{1}{x} + (1-x)\log\frac{1}{(1-x)} \tag{10}$$

Donde H es la función de incertidumbre para una variable aleatoria x.

III.IV.3. Entropía categórica cruzada

La entropía categórica cruzada es un caso más general de la métrica de entropía, la cual es una medida de incertidumbre para una variable discreta con K estados, esta forma especial está definida como lo muestra la ecuación 11 [52] :

$$H(x) \equiv -\sum_{k=1}^{K} p(x=k)\log_2 p(x=k) \tag{11}$$

Donde H es la función de incertidumbre para una variable aleatoria x, k es el número de posibles estados para la variable x y p es la probabilidad para un estado dado.

III.IV.4. Distancia euclidiana

La $\boldsymbol{norm-t^2}$ o distancia euclidiana $|\boldsymbol{x}|$ es una norma vectorial que se define como el vector complejo mostrado en la ecuación 12:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \tag{12}$$

para

$$|\mathbf{x}| = \sqrt{\sum_{k=1}^{n} |x_k|^2} \tag{13}$$

Donde k es la dimensión en la cual se mide el espacio entre puntos.

III.IV.5. 3D PCK

La distancia euclidiana tiene un problema fundamental al momento de ser empleado como una medida del error, y es que, puede ser perceptualmente engañoso en el sentido de que un error pequeño puede verse afectado por el operador cuadrático y pasar desapercibido, al mismo tiempo que un error grande en uno de los de elementos se puede amplificar y ofrecer una mala retroalimentación de los resultados reales.

Para reducir el impacto de algunas de estas observaciones, se propuso la métrica PCK [53], la cual se muestra en la ecuación 14, que emplea una tolerancia perceptual dependiendo el número de dimensiones presentes en el espacio en el que se está trabajando, se recomienda que para espacios donde se emplean tres dimensiones, se empleé un margen de tolerancia de $t = 150\,mm$.

$$CPK = 100\frac{1}{p_n}\sum_{i=1}^{p_n}\left[\begin{cases}1\ if\ D_e(P_{x_{1_{ij}}}, P_{x_{2_{ij}}}, P_{y_{1_{ij}}}, P_{y_{2_{ij}}}, P_{z_{1_{ij}}}, P_{z_{2_{ij}}}) \geq 150\ mm \\ 0\ if\ D_e(P_{x_{1_{ij}}}, P_{x_{2_{ij}}}, P_{y_{1_{ij}}}, P_{y_{2_{ij}}}, P_{z_{1_{ij}}}, P_{z_{2_{ij}}}) < 150\ mm\end{cases}\right] \tag{14}$$

Donde p_n es el número de puntos y P es algún punto de las nubes de puntos que se están comparando.

La ventaja de este método, es que los errores debajo de esta métrica son perceptualmente indistinguibles, y no afectan de forma significativa el resultado final, esto se puede apreciar mejor en la Fig. 11, donde mostramos como los puntos que se encuentran dentro del umbral de distancia del punto esperado se consideran como puntos válidos y se les asigna el valor de 1, a los puntos que no cumplen con este requisito se consideran como inválidos y se les asigna el valor de 0.

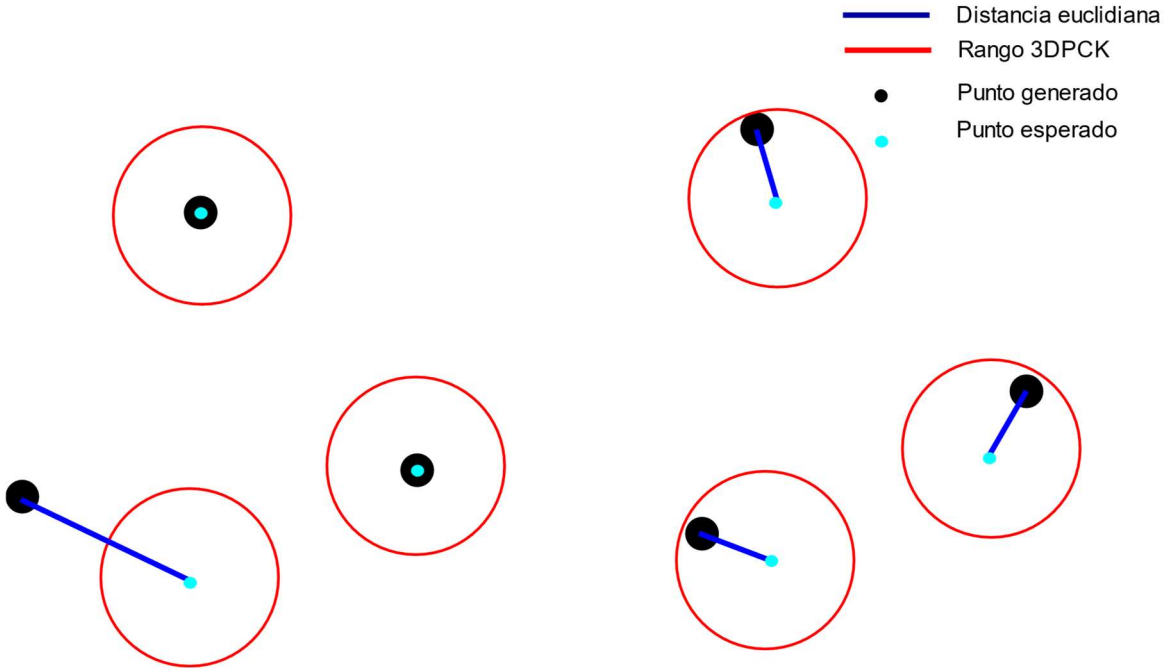

Fig. 11 Comparación gráfica entre la métrica 3DPCK y la distancia euclidiana.

III.IV.6. Prueba de Kolmogorv-Smirnov

La prueba de Kolmogorv-Smirnov o simplemente K-S [54] es un método de prueba no paramétrico para determinar la correlación que existe entre dos distribuciones de probabilidad, mediante su forma empírica acumulativa.

La ecuación para calcular la distancia que representa la métrica de K-S para dos muestras n, m de una variable aleatoria x está dada por la ecuación 15:

$$D_{n,m} = \sup_{x} \left| F_{1,n}(x) - F_{2,m}(x) \right| \tag{15}$$

Esto representa la distancia más grande entre la diferencia de probabilidad acumulativa entre 2 distribuciones. Una de las principales ventajas de la prueba de K-S, es que no requiere que las distribuciones que se están evaluando seas normales, pueden ser de hecho cualquier tipo de distribución.

III.IV.7. Inception score

El puntaje incepción [55] (IS por sus siglas en inglés), se basa en el principio de que las muestras que contienen objetos significativos deben tener una distribución de etiquetas condicional p (y | x) con baja entropía, esto es que para cada una de las imágenes exista una alta probabilidad de pertenecer a una clase en particular, además, se espera que el modelo genere imágenes variadas.

Cabe destacar que para esta métrica se emplean las predicciones obtenidas del modelo Inception V3 [56]. La métrica se define como se muestra en las ecuación 16, donde y representa los datos esperados y x los generados.

$$\exp\left(\mathbb{E}_{\boldsymbol{x}} \mathrm{KL}\left(p(y \mid \boldsymbol{x}) \parallel p(y)\right)\right) \tag{16}$$

donde $\int p(y \mid \boldsymbol{x} = G(z))dz$ debería tener una alta entropía.

III.IV.8. Fréchet inception distance

La distancia Inception de Fréchet [57] (FID por sus siglas en inglés) es una versión mejorada del IS. El problema con el IS es que no puede comparar las estadísticas de la muestra real con las muestras generadas o sintéticas. Por otro lado, el FID emplea la distancia de Wasserstein-2 para medir los dos primeros momentos o polinomios, la covarianza y el promedio, para las expectativas de la igualdad $p(.) = p\_w\ (.)$, Que se mantienen si y solo si $\int p(.)f(x)dx = \int p\_w\ (.)f(x)$, donde $f\ (x)$ son los polinomios para los datos dados, entonces x se sustituye por una de las capas de codificación del modelo Inception V3 [56] para obtener atributos relevantes. La métrica FID está representada por la ecuación 17, donde m es la media y $\boldsymbol{C}$ es la covarianza de la capa de codificación de la red Inception v3 para cada entrada x.

$$d^2\big((\boldsymbol{m}, \boldsymbol{C}), (\boldsymbol{m}_w, \boldsymbol{C}_w)\big) = \|\boldsymbol{m} - \boldsymbol{m}_w\|_2^2 + \mathrm{Tr}\left(\boldsymbol{C} + \boldsymbol{C}_w - 2(\boldsymbol{C}\boldsymbol{C}_w)^{1/2}\right) \quad (17)$$

IV. Metodología

En esta sección se muestran los pasos que se llevaron a cabo para poder conseguir cada uno de ellos puntos expuestos en los objetivos de la tesis, para esto se llevaron a cabo los siguientes pasos:

1. Se generó una base de datos empleando un software de captura y un sensor de captura Kinect, también se obtuvo una base de datos de interacciones entre sujetos en tareas más comunes para probar la efectividad de los modelos propuestos en actividades más generales.
2. Se entrenaron varios modelos para comprobar cuál era el más efectivo en cada tarea y seleccionar aquel que sea más adecuado para la generación de poses.
3. Se propuso e implemento un sistema de visualización de poses para poder mostrar las poses obtenidas durante la captura, preprocesamiento y generación de poses.
4. Se propusieron un conjunto de métricas para evaluar el desempeño de los modelos propuestos para ambas bases de datos.

Para poder mostrar los objetivos de esta tesis se siguió el proceso que se muestra en la Fig.12 en él se muestra la etapa de captura de poses mediante el uso del dispositivo Kinect v2, el almacenamiento se las poses en el disco duro de la computadora en formato CSV [58], el preprocesamiento de los datos que se llevó a cabo usando Python, el entrenamiento e inferencia de las poses en la computadora y la visualización así como la evaluación de las poses generadas.

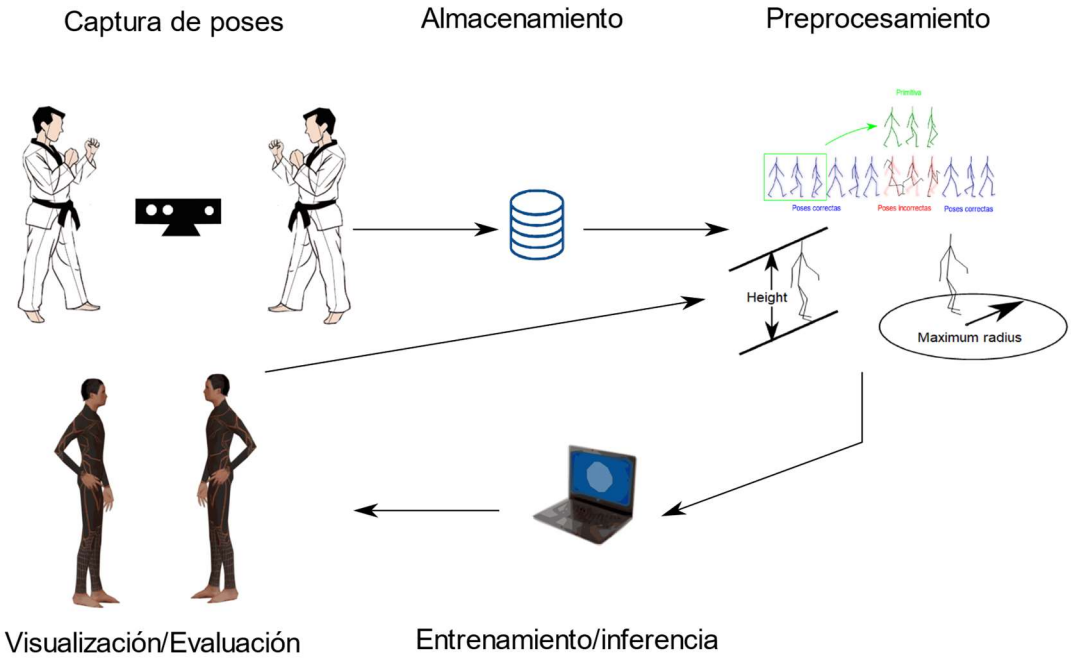

Fig.12 Diagrama del proceso.

Se empleó un sensor Kinect 2.0, Conectado a una computadora con un CPU Intel(R) Core(TM) i7-8750H CPU @ 2.20 GHz (12 CPUs), ~2.2 GHz y un GPU NVIDIA GeForce GTX 1060 with Max-Q Design, que también se empleó para el entrenamiento de la red GAN.

El software usado para la implementación del algoritmo de la red GAN fue el editor de lenguaje Python Spyder 4.0 [59], Python 3.6 [60], Numpy 1.20.1 [61], Matplotlib [62] y la librería Tensor Flow [63] mediante su implementación 2.4 [64].

IV.I. Generación de base de datos

IV.I.1. Software de captura

El software que se empleó para la captura de poses fue el paquete de C# de Visual Studio 2019 de Microsoft [65] y el SDK de Kinect v2 [66], una captura de pantalla del software empleado se muestra en la Fig. 13 en está podemos ver el botón de inicio de captura con la etiqueta ‘Start’ y la caja de texto con la etiqueta ‘File name’ empleada para escribir el nombre de la secuencia que deseamos guardar, este software está basado en el código del paquete de ejemplos del SDK de Kinect v2 con el título ‘Body Basic-WDF’.

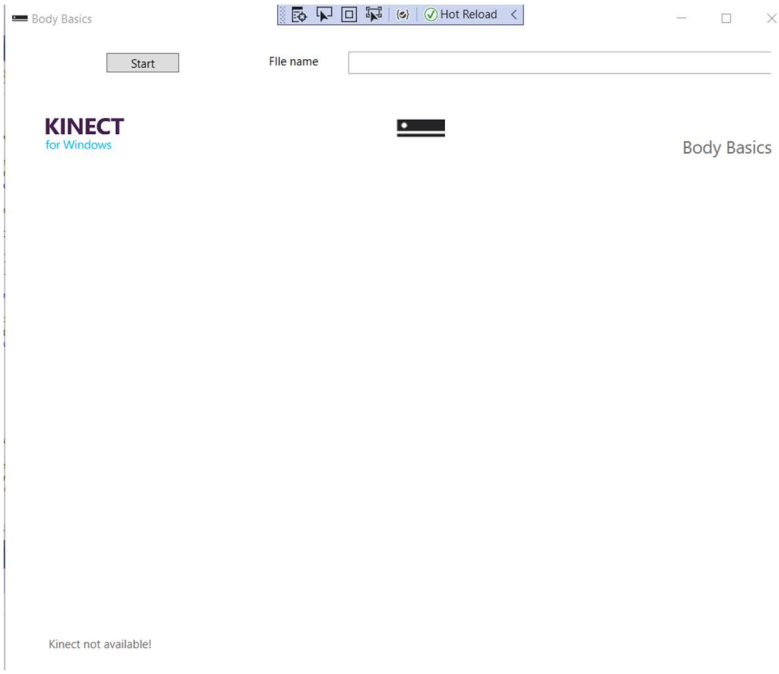

Fig. 13 Captura de pantalla del software de captura usado.

IV.I.2. Captura de poses

La captura de poses para la base de datos Kinect, forma en la que llamaremos de aquí en adelante a la base de datos que se obtuvo para diferenciarla de la base de datos de la CMU, se llevó a cabo con el sensor Kinect mencionado en la sección anterior empleando el software de captura de la Fig. 13, el cual será colocó a 1.5 metros de los sujetos que fueron grabados, así mismo los sujetos no se alejaron más de 3.5 metros de la cámara, como se muestra en la Fig. 14, para facilitar este proceso se agregaron un

par de cintas paralelas en el piso donde los voluntarios llevaron a cabo sus actividades, las cuales fueron almacenadas y usadas para entrenar la red neuronal contenida en el sistema de simulación.

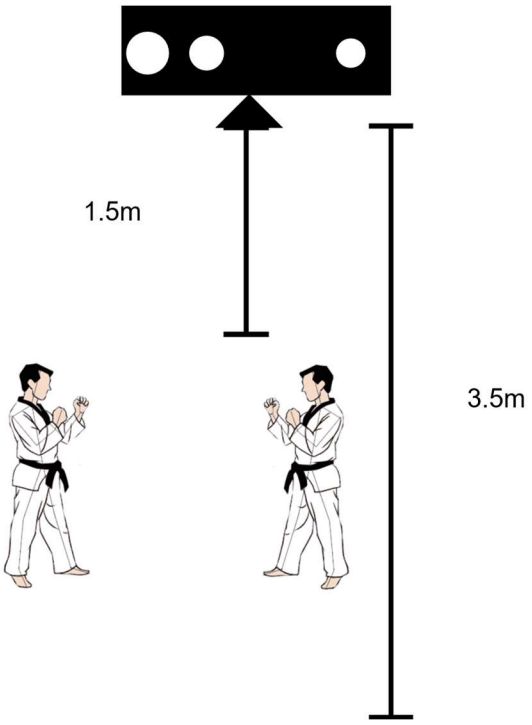

Fig. 14 Distancia de grabación de los practicantes

IV.I.3. Almacenamiento de poses

Después de concluir la captura de las poses empleando el software de captura y el Kinect, se almacenaron las poses capturadas en un archivo CSV, el cual sigue el formato de la Tabla 1, Podemos ver las primer columna almacena el tiempo en el que se grabó, la segunda el ID de la pose. Las siguientes 75 filas contienen los ejes de cada uno de los puntos que se grabaron, empezado con el eje X y terminando en el Y, las últimas 25 consisten en el tipo de dato que se capturó, dependiendo de si este se realizó correctamente o fue inferido, cabe destacar que no siempre se capturan correctamente los puntos que Kinect considera como seguidos exitosamente, y más adelante proponemos una solución sencilla este problema.

Tabla 1 Formato de la tabla donde se almacenan las secuencias de poses.

Time	ID	Joint count	SpineBase_X	SpineBase_Y	…	SpineBase	SpineMid	…	Neck
28:45.0	7.2058E+16	25	0.67931867	-0.29640555	…	Tracked	Tracked	…	Tracked
28:45.0	7.2058E+16	25	0.68391931	-0.29869863	…	Tracked	Tracked	…	Tracked
28:45.0	7.2058E+16	25	0.68959624	-0.29994413	…	Tracked	Tracked	…	Tracked
⋮	⋮	⋮	⋮	⋮	…	⋮	⋮	…	⋮
⋮	⋮	⋮	⋮	⋮		⋮	⋮		⋮
28:45.1	7.2058E+16	25	0.72679293	-0.30321574	…	Tracked	Tracked	…	Tracked

En total se capturaron 57 secuencias de poses entre los 2 sujetos que se grabaron.

IV.I.4. Preprocesamiento de la base de datos

En este trabajo se emplean 2 bases de datos, la base de datos de Cambridge Mellon University (CMU) [30] de la sección “Two subjects”, y una base de datos que se recolectó usando un dispositivo Kinect V2, el preprocesamiento de ambas bases de datos es ligeramente distinto, ya que la base de datos CMU posee datos de muy alta calidad y poses perfectamente capturadas, esto es sin errores de tracking.

IV.I.4.1. Preprocesamiento de la base de datos CMU

Primero, fue necesario definir cuantos puntos podía poseer cada sujeto, ya que algunas grabaciones de la base datos CMU no tienen la misma cantidad de puntos o tienen puntos con nombres distintos en posiciones distintas. En total se definió que los primeros 41 puntos que se encontraran serían los que se tomarían en cuenta para llevar a cabo el procesamiento, el resto fue eliminado directamente.

Después de definir de la cantidad de puntos por secuencia, es necesario establecer el rango de los datos en la base de datos que vamos a emplear, para lo cual únicamente se procede a tomar la altura en z de la cabeza en cada secuencia, y a dividir la secuencia de poses entre el valor obtenido, obteniendo de esta forma secuencias donde los sujetos representados por las nubes de puntos tienen la misma altura.

Ahora, ambos individuos tienen la misma altura es necesario normalizar los puntos en las articulaciones de las secuencias de poses a un valor entre -1 y 1, esto con la intención de facilitar el proceso de aprendizaje de las redes neuronales que van a emplear esta

base de datos, esto se ilustra mejor en la Fig. 15, donde podemos apreciar cómo se toma la altura total del sujeto y la distancia máxima del centro en la cual los puntos de cada pose pueden ser representados.

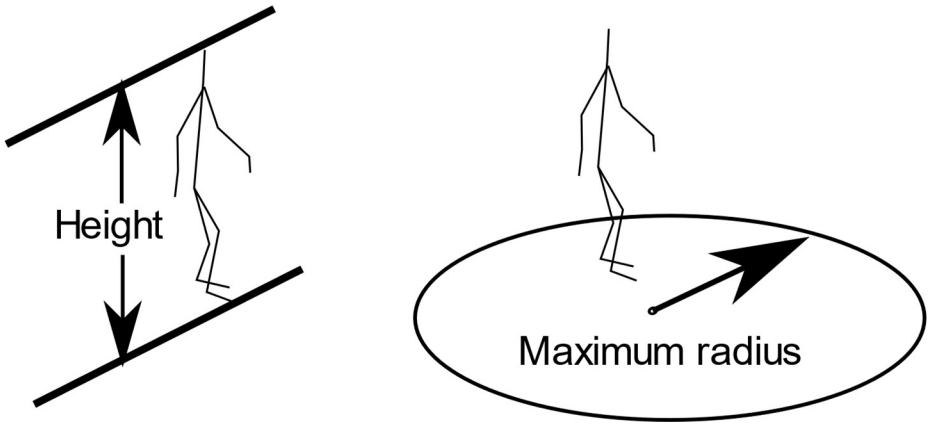

Fig. 15 Representación gráfica del proceso de estandarización de poses.

Ahora, los valores se encuentran en el rango (-1,1) y hace falta segmentar la secuencia en subsecuencias más pequeñas, quedando así, subsecuencias de 3 poses y 43 puntos para esta base de datos, que se guardaron en formato pickle [43], y, al tratarse de dos individuos, se almacenaron las poses del sujeto 1 como “fs”m y del sujeto 2 como “ss”m, donde m es el número de pose en la secuencia de poses correspondiente.

IV.I.4.2. Preprocesamiento de la base de datos Kinect

La cantidad de puntos que el sensor Kinect v2 pudo grabar es fija, 25 puntos por fotograma, por esta razón no es necesario eliminar puntos de las grabaciones tomadas, lo que se hizo con las secuencias fue determinar cuáles de ellas forman una pareja de secuencias, esto significa poses que pertenecen a los sujetos A y B, y que comparten el mismo tiempo de inicio y final, aquellas secuencias que no cumplen este criterio fueron descartadas, dejando un total de 38 secuencias.

El proceso de estandarización que se siguió fue el mismo que el explicado en la sección anterior e ilustrado en la Fig. 15.

Debido a que Kinect capturó algunas poses incorrectamente, incluso cuando reporta que el estado de éstas es el correcto, es necesario reparar algunas de las poses, en este caso, se emplea un algoritmo muy sencillo, cuyos pasos se enlistan a continuación:

a) Se selecciona un grupo de secuencias de poses correctamente capturadas que llamaremos primitivas.
b) Se calcula la distancia euclidiana entre cada subsección sucesiva de poses que pertenezca a la secuencia de donde fue tomada la primitiva.
c) Las subsecuencias con los máximos locales a una distancia de por lo menos dos veces la cantidad de fotogramas de las primitivas se seleccionan como pivotes.
d) Las subsecuencias ubicadas a los pivotes seleccionados se reemplazan con las primitivas.

El proceso descrito en la lista se puede apreciar mejor en la Fig. 16 donde podemos observar cómo se selecciona la primitiva de una secuencias de poses correctamente tomadas y se usa para reemplazar una secuencia de poses incorrectamente capturadas.

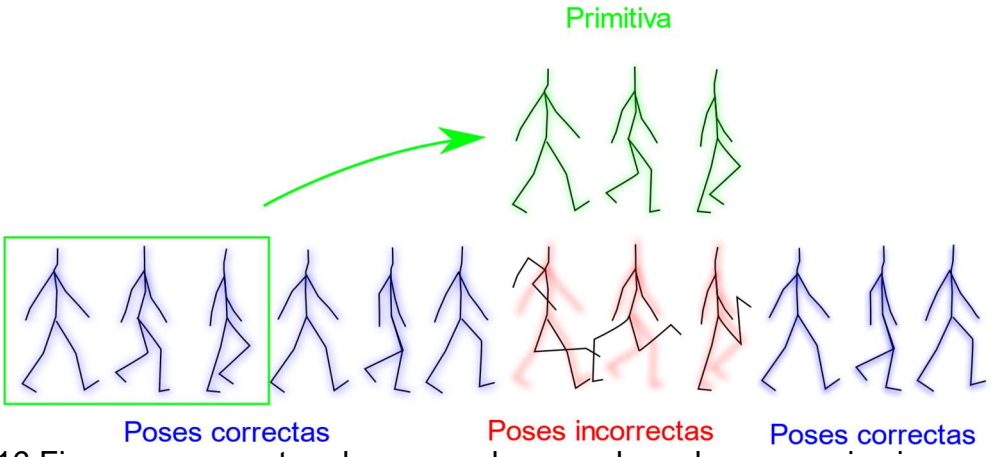

Fig. 16 Figura que muestra el proceso de reemplazo de secuencias incorrectas.

IV.I.5. Definición de la estructura de las poses

Para ambas bases de datos, tanto como la base de datos CMU como la base de datos Kinect, se usó la misma estructura para representar las poses que se usaron como entrada para la red neuronal. En este caso, se asignaron coordenadas a una nube de puntos representada por un vector P_0 el cual contiene sus componentes espaciales, es decir, X, Y, Z, para cada c como si se tratara de colores en un mapa de bits de 1XN, donde N es el número de articulaciones del método de captura que se esté empleando, en este caso 43, como se puede ver en la Fig. 17.

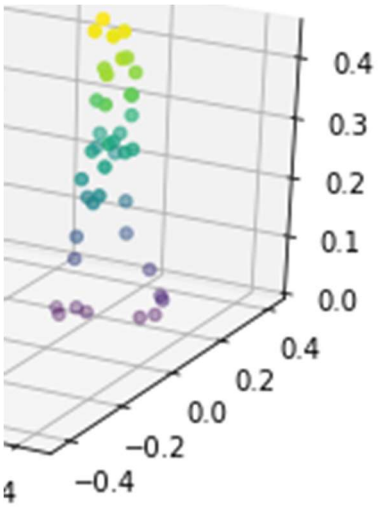

Fig. 17 Pose representada por una nube de puntos

Posteriormente, hace falta aplanar la matriz de tal manera que las posiciones X, Y y Z quedaran distribuidas en el lugar de cada uno de los puntos en una matriz bidimensional que represente las coordenadas de cada articulación, de tal manera que quedó un vector de 1 dimensión, como se puede apreciar en la ecuación 17.

$$\begin{bmatrix} q_{x11} & q_{y11} & q_{z11} \\ q_{x1} & q_{y12} & q_{z1} \\ q_{x1} & q_{y13} & q_{z13} \\ & \vdots & \\ q_{x1m} & q_{y1m} & q_{z1m} \end{bmatrix} \downarrow \quad (17)$$

$$[q_{x11}, q_{y11}, q_{z11}, q_{x12}, q_{y12}, q_{z12}, q_{x13}, q_{y13}, q_{z13}, \cdots, q_{x1m}, q_{y1m}, q_{z1m}]$$

Donde m es el número de puntos en cada pose.

Esta reducción de dimensionalidad es necesaria para poder procesar correctamente cada una de las poses, pero sobre todo, para permitir que cada elemento de las coordenadas pueda influenciar al otro mediante operaciones matemáticas, veremos este detalle más adelante, cuando exploremos el entrenamiento de las redes neuronales.

IV.I.6. Estructura de secuencia de poses

Para crear la estructura de las secuencias de poses que se dan como entrada a la red neuronal, se tomó la definición de la estructura de la sección anterior y se llevó a una dimensionalidad superior, al apilar varios vectores en una matriz que representara un grupo de poses en orden, como se ve en la ecuación 18.

$$\begin{bmatrix} q_{x11} & q_{y11} & q_{z11} & q_{x12} & q_{y12} & q_{z12} & q_{x13} & q_{y1} & q_{z13} & \cdots & q_{z1m} \\ q_{x21} & q_{y21} & q_{z21} & q_{x2} & q_{y22} & q_{z22} & q_{x23} & q_{y23} & q_{z2} & \cdots & q_{z1m} \\ q_{x3} & q_{y31} & q_{z31} & q_{x3} & q_{y32} & q_{z32} & q_{x3} & q_{y313} & q_{z33} & & q_{z13} \\ & & & & \vdots & & & & & \ddots & \\ q_{xn1} & q_{yn1} & q_{zn1} & q_{xn2} & q_{yn2} & q_{zn2} & q_{xn3} & q_{yn3} & q_{zn3} & & q_{znm} \end{bmatrix} \quad (18)$$

Donde n es el número de poses y m el número de puntos por pose, en cada secuencia. Un ejemplo de lo mencionado en el párrafo anterior, pero con tres poses en secuencia como se muestra en la Fig. 18, aunque pareciera que son más bits, esto se debe a la interpolación por default que ocurre al graficar la matriz.

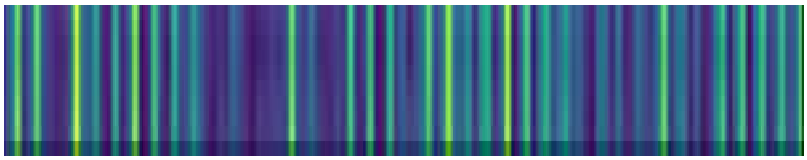

Fig. 18 matriz de poses representada como imagen RGB

IV.II. Diseño de los modelos

Para poder generar poses a partir de otra pose de entrada, es necesario tener un modelo que haga la inferencia de la pose, en este caso, se emplearon redes neuronales para ejecutar esta tarea, los modelos que se usaron fueron los siguientes:

- Red neuronal densa.
- Red neuronal convolucional.
- AC-CGAN semi supervisada convolucional con clasificador auxiliar basado en etiquetas no supervisadas.
- AC-CGAN semi supervisada convolucional con clasificador auxiliar basado en etiquetas no supervisadas y un paso de entrenamiento supervisado adicional.
- AC-CGAN con generador recurrente y 2 discriminadores auxiliares, uno basado en etiquetas no supervisadas y el otro en la métrica 3DPCK.

Los mismos modelos fueron empleados tanto para la base de datos CMU como para la base de datos Kinect.

IV.II.1. Diseño de red neuronal densa

Primero, se hicieron pruebas con una red neuronal densa para determinar si era suficiente para resolver el problema de la generación de poses que se trata en este problema, en este caso, se escogió una red neuronal densa multi capa con función de activación Leaky ReLU por las ventajas que suponen al ser usadas respecto a otras funciones de activación [68] gracias a su no linealidad.

Se emplearon 10 capas densas de neuronas con una cantidad de neuronas igual al producto de las dimensiones de la imagen de entrada, entre la cada de entrada de la red neuronal y las intermedias se aplanará la matriz de la imagen y la capa de salida de la red, y las capas intermedias se volverá a regresar a su forma matricial empleando una operación de reshape, que llevó a cabo la operación inversa a la operación de aplanado, todo esto se ilustra en la Fig. 19.

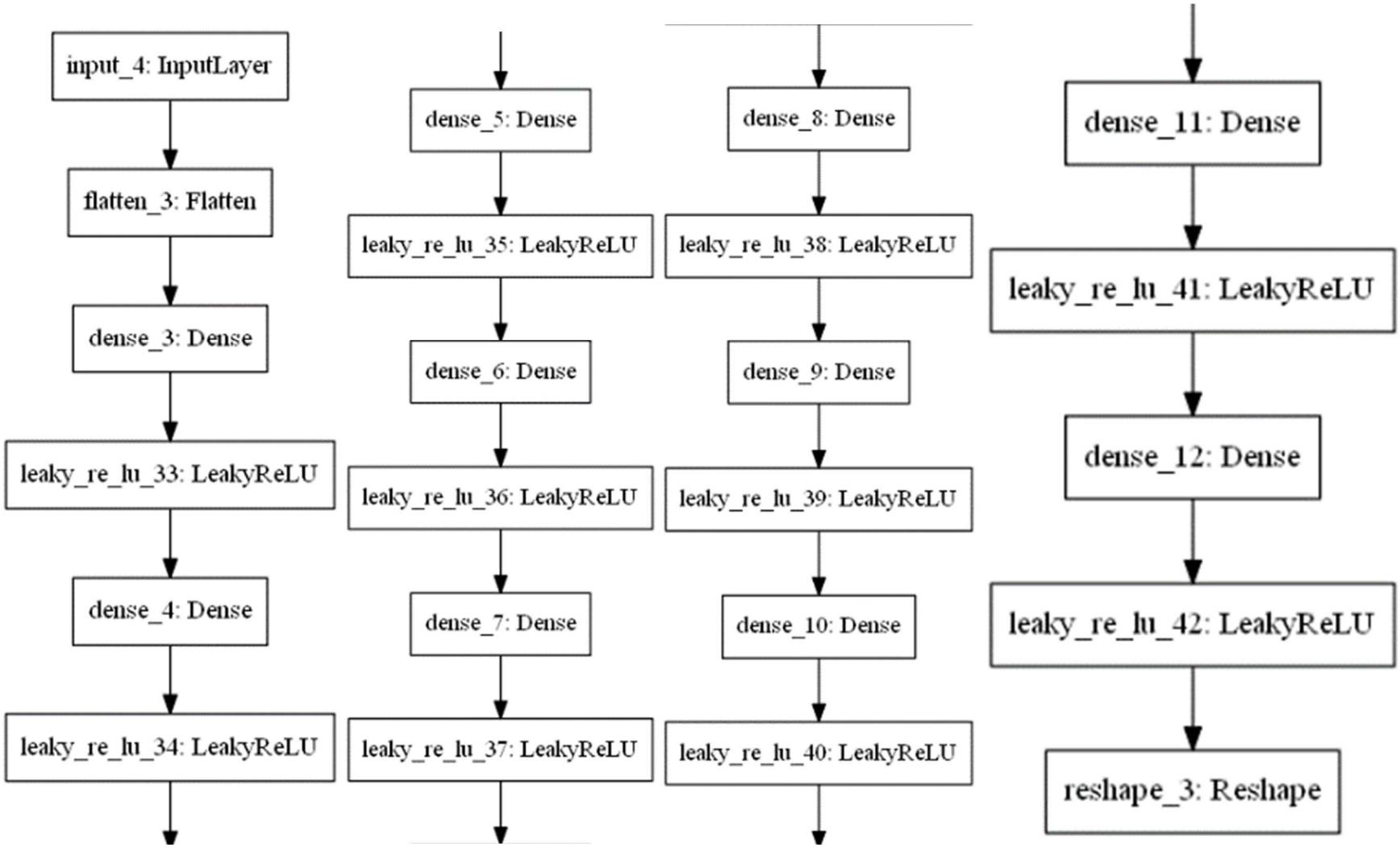

Fig. 19 Modelo de la red neuronal densa

Se empleó el optimizador Ada delta para evitar los problemas que presentan otros optimizadores estocásticos, que tienen mayor dificultad para salir de mínimos locales y su reducida sensibilidad a los cambios en sus híper parámetros [45].

Los detalles de los parámetros para el modelo mostrado en la Fig. 19 se muestran en el Anexo 1, aquí se pueden apreciar con mucho más detalle cuantas neuronas tiene cada capa, el rango de valores entre los cuales se inicializaron los pesos, entre otros.

IV.II.2. Diseño de red neuronal convolucional

En este caso, se escogió una red neuronal convolucional transpuesta, ya que este tipo de red eleva la complejidad de la entrada [70] y se hizo la suposición que esto ayudaría a elevar la variedad de soluciones, usando la función de activación Leaky RELU por las ventajas que ya se mencionaron.

Se emplearon 11 capas convolucionales con 32 filtros de 4x4, entre la última capa convolucional transpuesta de la red neuronal y las intermedias se aplanó la matriz de la imagen para permitir la inserción de una capa densa y la capa densa de la red y la capa de salida se volvió a regresar a fu forma matricial empleando una operación de reshape, que llevó a cabo la operación inversa a la operación de aplanado, todo esto se ilustra en la Fig. 20.

Se empleó el optimizador Ada delta por las mismas razones que en la red densa. Los detalles de los parámetros para el modelo mostrado en la Fig. 20 se muestran en el Anexo 2, aquí se pueden apreciar con mucho más detalle cuantas neuronas tiene cada capa, el rango . entre los cuales se inicializaron los pesos, entre otros

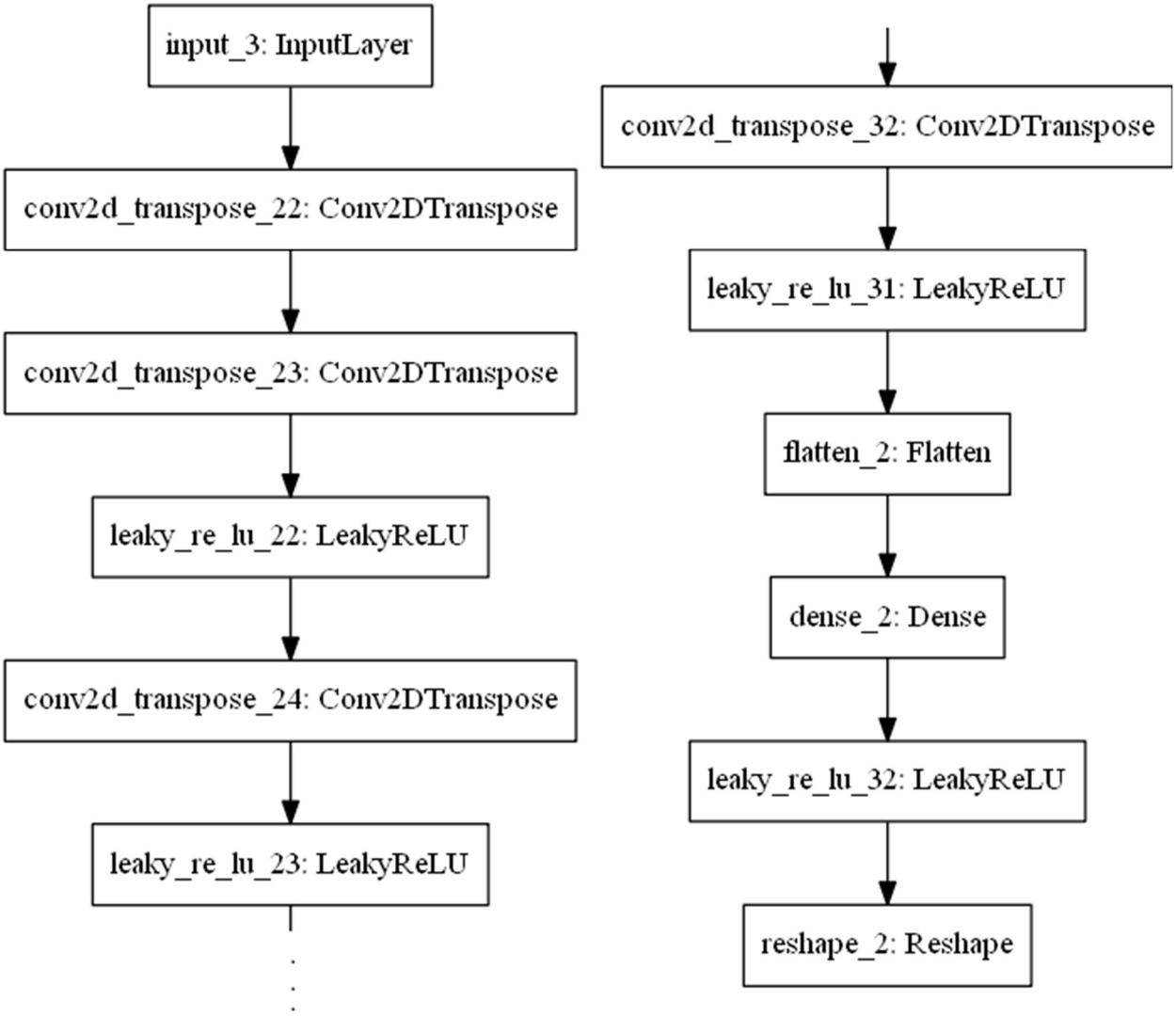

Fig. 20 modelo de la red convolucional

IV.II.3. Diseño de red neuronal AC-CGAN

Para este modelo se agregaron capas de dropout y batch normalization, ya que estas se recomiendan para este tipo de red [71] y sin estas capas el modo colapsaba inmediatamente incrementando drásticamente el error. Ahora se implementó un discriminador que cumpliría la función de distinguir las muestras generada y las reales, así como de clasificar en tipos cada uno de los movimientos ya que esto reduce la probabilidad de que ocurra colapso de modo entre las salidas mientras que mejora la variedad y la calidad de la solución respecto a otros métodos adversariales [48].

Se emplearon 11 capas convolucionales con 32 filtros de 4x4, entre la última capa convolucional transpuesta de la red neuronal y las intermedias se aplanó la matriz de la imagen para permitir la inserción de una capa densa y entre la capa densa de la red y la capa de salida se regresa a su forma matricial empleando una operación de reshape, que llevó a cabo la operación inversa a la operación de aplanado, todo esto se ilustra en la Fig. 21.

Los detalles de los parámetros para el modelo mostrado en la Fig. 21 se muestran en el Anexo 3, aquí se pueden apreciar con mucho más detalle cuantas neuronas tiene cada capa, el rango de valores entre los cuales se inicializaron los pesos, entre otros.

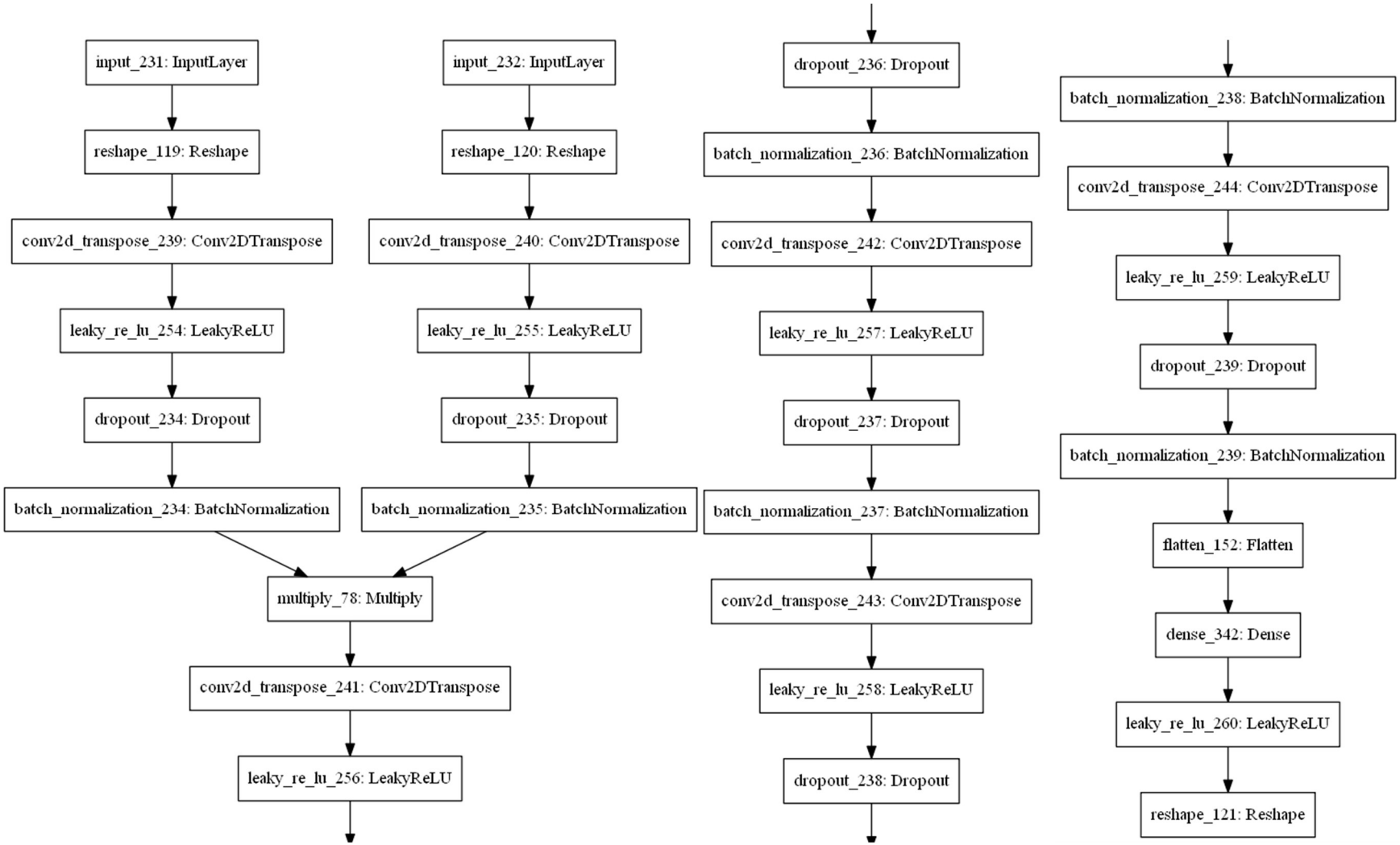

Fig. 21 modelo de la red SC-GAN

IV.II.4. Diseño de red neuronal AC-CGAN con MSE

En este caso se tomó el mismo generador que en la prueba con una AC-CGAN, sin embargo, se agregó una tercera salida auxiliar a la red discriminadora, se entrenó en el error cuadrático medio igual que la red convolucional, esto se debe a las siguientes razones:

a) Evita el sobre entrenamiento al reducir el error mientras se aproxima a 0, consiguiendo así que las salidas tengan más variedad
b) Se evita que caiga rápidamente a vacíos locales.
c) Se obtuvieron buenos resultados en la red convolucional cuando se empleó.
d) Se espera que evite que los puntos tengan coordenadas muy lejanas del valor esperado.

Se emplearon 3 capas convolucionales con 16 filtros de 9x4, entre la última capa convolucional transpuesta de la red neuronal y las intermedias se aplanará la matriz de la imagen para permitir la inserción de una capa densa y entre la capa densa de la red y la capa de salida se volverá a regresar a fu forma matricial empleando una operación de reshape, que llevará a cabo la operación inversa a la operación de aplanado, todo esto se ilustra en la Fig. 22, se empleó también el optimizador Ada delta para entrenar el modelo.

Los detalles de los parámetros para el modelo mostrado en la Fig. 22 se muestras en el

Anexo 3 donde se pueden apreciar con mucho más detalle cuantas neuronas tiene cada capa, el rango de valores entre los cuales se inicializaron los pesos, entre otros.

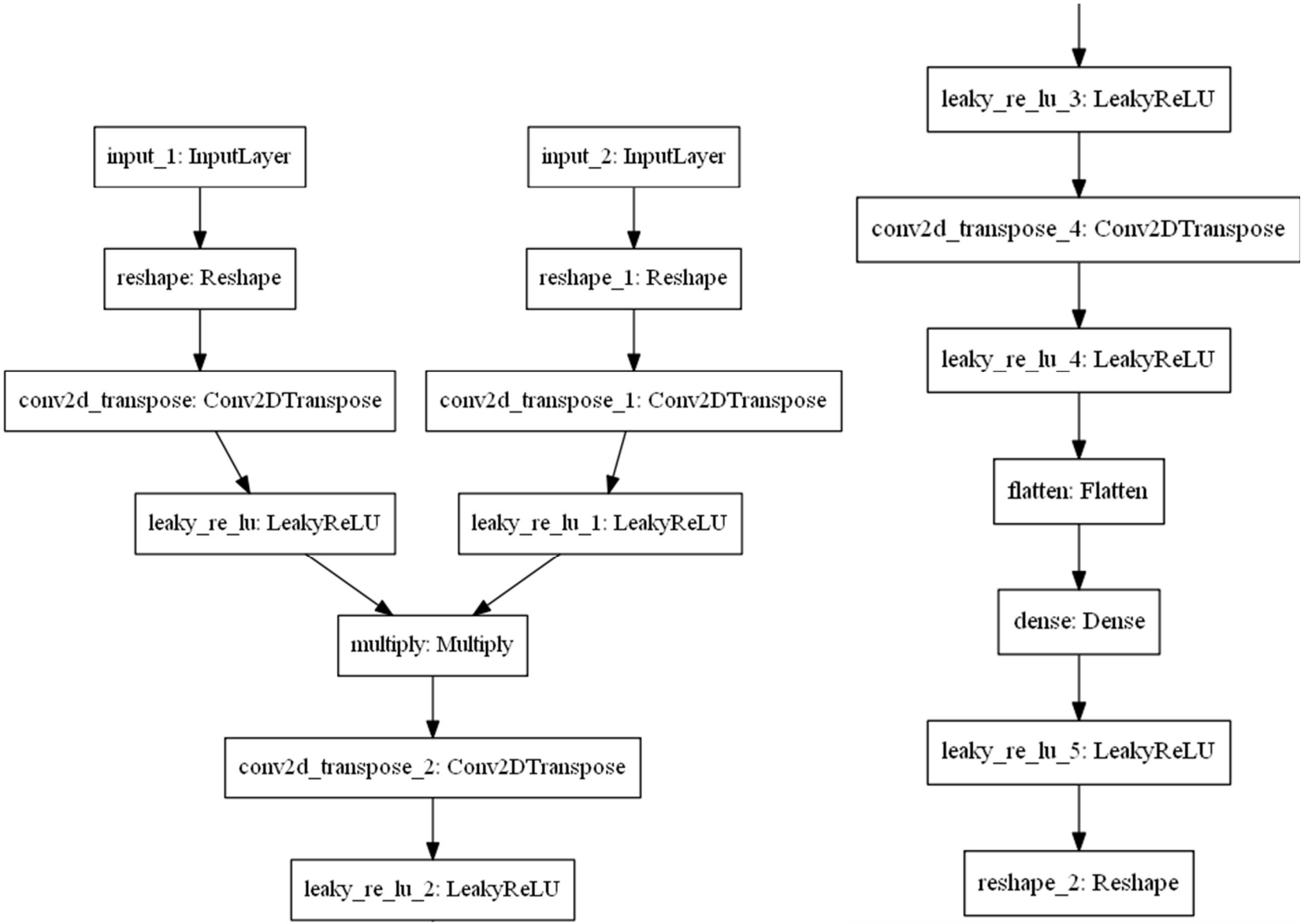

Fig. 22 modelo de la red propuesta

IV.II.5. Diseño de red neuronal recurrente AC-CGAN GRU

En la quinta prueba se tomó el mismo discriminador que en la prueba 2, sin embargo se cambió la arquitectura del generador y se adoptó una configuración basada en capas de neuronas recurrentes y se agregó un nuevo discriminador con una configuración similar a la del discriminador auxiliar pero con valores para los hiperparámetros diferentes, específicamente para el numero de filtros, así como una cantidad de capas diferente, los cambios pueden verse en la Fig. 23 y la Fig. 24.

El objetivo de emplear capas de neuronas recurrentes es hacer uso de las propiedades que estas poseen, particularmente su capacidad de almacenar información en forma de memoria en sus celdas, lo que permite hacer un uso más provechoso de datos en el tiempo, sobre todo si estos se componen de patrones complejos [73] como lo son el movimiento humano.

Los detalles de los parámetros para el modelo mostrado en la Fig. 23 y Fig. 24 se muestran en el Anexo 4 donde se pueden apreciar con mucho más detalle cuantas neuronas tiene cada capa, el rango de valores entre los cuales se inicializaron los pesos, entre otros.

En este experimento fue donde las métricas cambiaron para favorecer la naturaleza del problema que se desea resolver, las métricas que fueron escogidas son el 3DPCK y la distancia euclidiana también conocida para estimación de poses como MPJPE (Mean Per Joint Position Error) [53].

Con la intención de mejorar estos resultados, es posible emplear múltiples modelos con parámetros e hiperparámetros inicializados con diferentes valores, diferentes métodos y modelos, entrenando cada modelo con datos diferentes, etc. [51], además se sabe que es posible alcanzar mejores resultados combinando varios métodos que empleando un único modelo [52]. En este caso en particular, combinando redes neuronales con pesos

iniciales diferentes entre ellas, y los mismo hiperparámetros, esto se puede apreciar mejor en la
Fig. 25, donde se muestran las entradas que parametrizan la salida de cada red neuronal en amarillo, ensambladas de tal forma que la predicción de cada una de ellas se promedia dando lugar a una nueva pose generada con más calidad que si se hiciera la inferencia con una red.

Para crear el ensamble de modelos se entrenaron 5 modelos con los mismo datos, durante 13000 iteraciones cada uno, ya que se determinó del experimento anterior mediante una condición de paro de paciencia, donde este era el momento último en el cual la red dejaba de aprender después de 100 iteraciones.

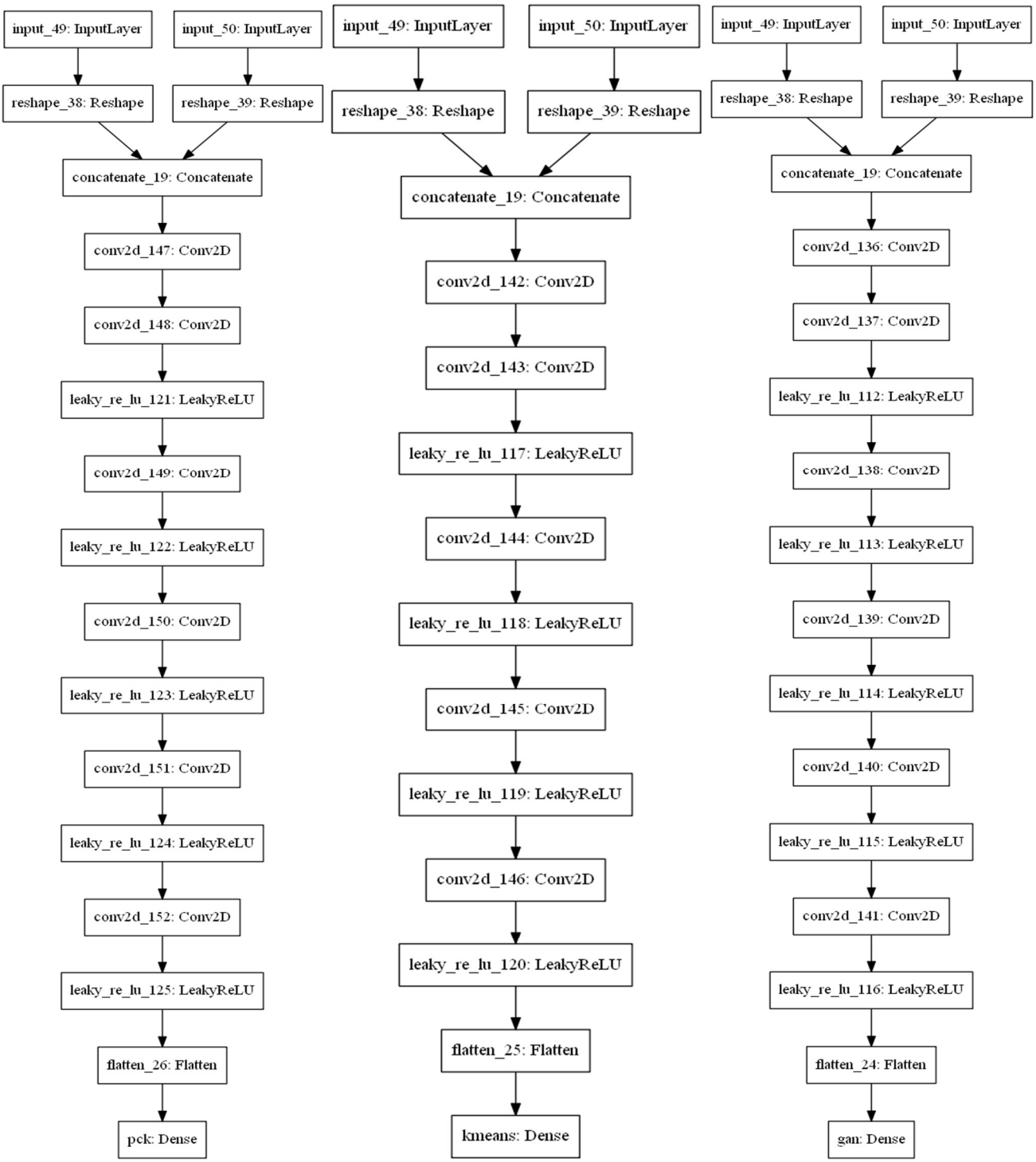

Fig. 23 De izquierda a derecha: Discriminador auxiliar con métrica 3DPCK, Discriminador auxiliar etiquetas K-means, discriminador principal.

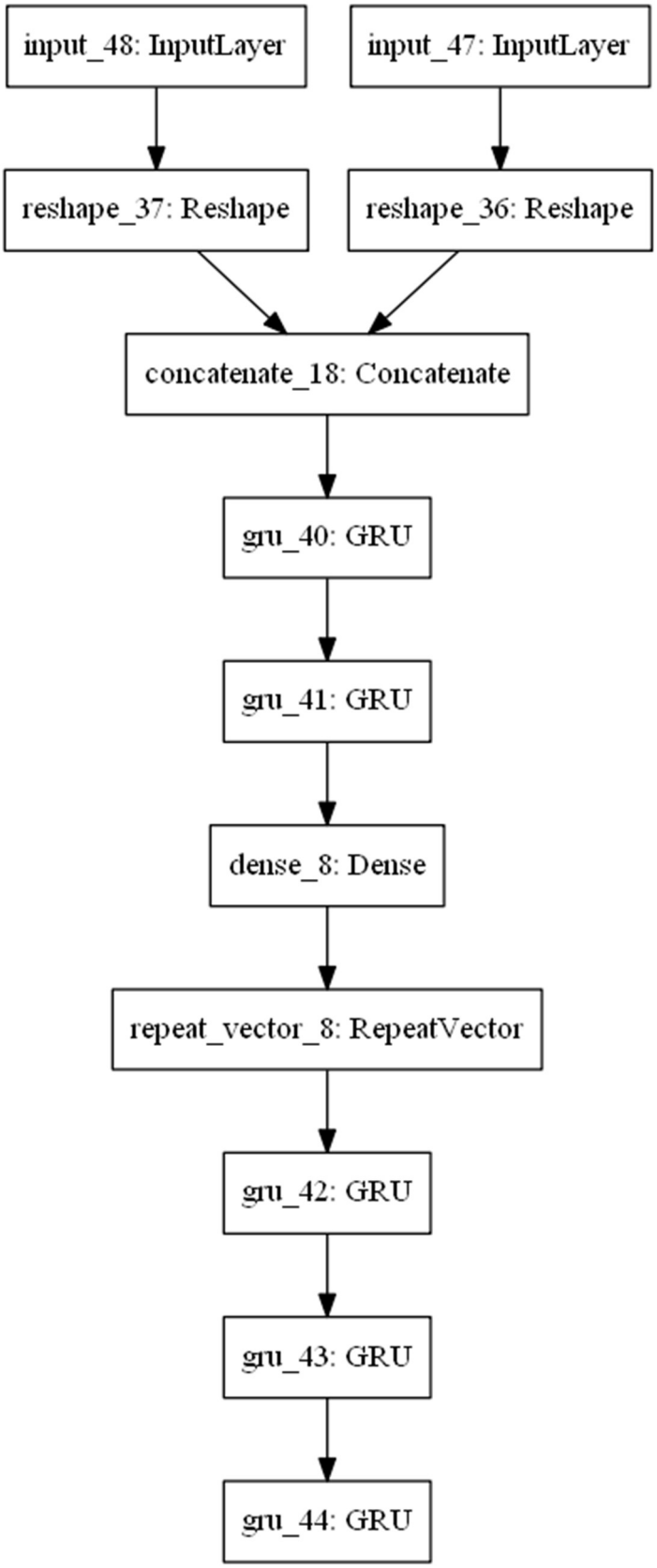

Fig. 24 Modelo del generador que hace uso de neuronas recurrentes

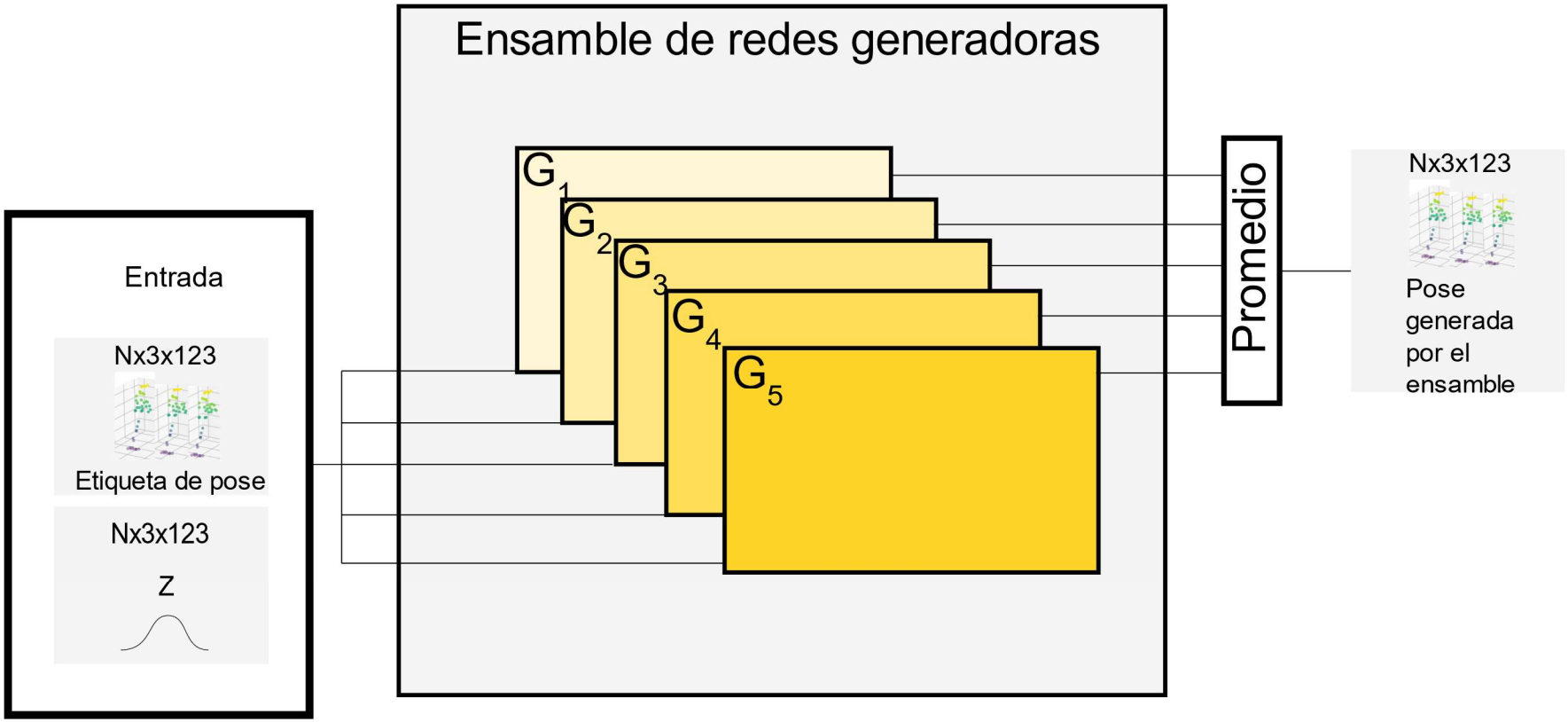

Fig. 25 Diagrama que representa el ensamble de redes neuronales generadoras que se emplea en este trabajo.

IV.II.6. Proceso de entrenamiento de redes neuronales

Para llevar a cabo el entrenamiento de la red neuronal independientemente del algoritmo que se usó para la optimización de los parámetros de la red, consiste en entrenar la red con un subconjunto de entrenamiento, tomado de la base de datos total, y evaluar los datos con un subconjunto de datos de prueba usando la distancia euclidiana y la métrica 3DPCK. Si pasan 2000 iteraciones sin que la métrica de 3DPCK mejore en el subconjunto de prueba el entrenamiento se detiene y se toman los pesos de la red que tuvo el valor más alto de 3DPCK para el subconjunto de prueba.

IV.III. Visualización de poses

Para llevar a cabo la visualización de secuencias se empleó la librería matplotlib [62], esta es una librería de visualización y dibujo de graficas para Python, la cual tiene muchas prestaciones en lo que respecta al dibujo y graficación de funciones, nubes de puntos, graficas, curvas y otros tipos de gráficas, e incluso mapas de bits.

El procedimiento para mostrar las pose obtenidas consiste en empelar un código de Python que nos muestre las nubes de puntos de los subjetos A y B, de tal manera que se puedan apreciar las interacciones que estos tienen durante sus actividades.

El procedimiento descrito en esta sección se usó para mostrar las secuencias obtenidas a través del post procesamiento y después de la inferencia del modelo.

El procedimiento es más bien sencillo, se convierte la estructura de las poses descrita en la sección de definición de la estructura de las poses, en secuencias de puntos con sus respectivas 3 coordenadas, como se muestra en la ecuación 19, donde x, y y z son las coordenadas en el espacio y P_{xyz} la secuencias puntos.

$$P_{xyz} = \begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ \vdots & \vdots & \vdots \\ x_n & y_n & z_n \end{bmatrix} \tag{19}$$

De tal manera que se pueden representar como puntos en 3D, ahora se crea una figura con la función “fig” de la librería matplotlib y se grafica una “scatter plot” de tal forma que queda dibujada la nube de puntos que presenta a cada sujeto en una gráfica dentro d de la figura, como se muestra en la Fig. 26 donde ambas poses se representan con colores que van creciendo en brillo de abajo hacia arriba para una mejor interpretación de las poses que representan, los ejes de estas figuras no tienen unidades al estar normalizadas.

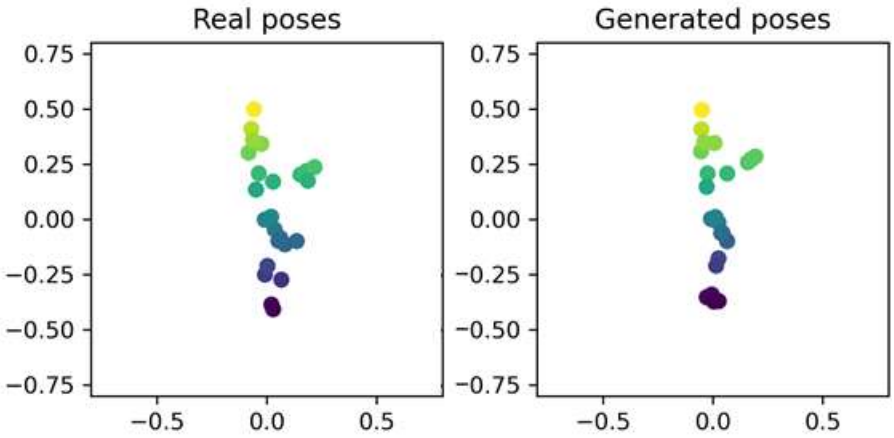

Fig. 26 Representación de la comparación de 2 poses.

IV.IV. Evaluación de resultados

Para evaluar los resultados obtenidos de los modelos propuestos se emplearon las métricas descritas en esta subsección, ya que la mayoría de los métodos son generativos y la capacidad de crear una gran variedad de muestras. las métricas que son inherentes a modelos generativos como el inception score, Frechet inception distance y el K-S, también se usaron para evaluar a los modelos no generativos (Convolucional y denso), a continuación se listan las métricas que se emplearon en las pruebas:

- Distancia euclidiana promedio: se empleó para verificar que tan cerca están los puntos de una pose a los puntos de otra pose de forma general, esta métrica es muy útil para tener una idea general de que similares son las secuencias generadas a las reales.
- 3DPCK: se empleó para calcular la cantidad de puntos generados que están dentro de un cierto umbral de distancia en milímetros cuyo radio se dibuja desde

el punto real. Esta métrica es muy útil para ver qué tan adecuadas son las poses generadas respecto a las reales punto por punto.

- Prueba de Kolmogórov-Smirnov: Esta prueba no paramétrica se empleó para medir la similaridad entra las distribuciones de los puntos de forma individual, el promedio y el máximo de esta métrica se usó para tener un panorama general de que tan parecidas son las distribuciones de los datos generador y los reales.
- Inception distance: la prueba de IS se empleó para determinar qué tan parecidas son las distribuciones de los datos reales y los generados, se entrenó un clasificador Inception V3 para obtener la probabilidad de que cada una de las muestras generadas pertenezca a una clase en particular de los datos reales.
- Fréchet inception distance: Funciona de forma similar a la IS, solo que se comparan las distribuciones estadísticas de los datos obtenidos una capa antes de la salida de red Inception V3, de tal manera que se comparan los primeros 2 momentos estadísticos producto de la clasificación de los datos reales y los generados, esta medida es útil para determinar la similaridad entre la distribución de datos generador y los reales.

Los resultados también fueron evaluados a ojo para determinar su calidad, fe la misma forma, con la intención de explicar la contribución de cada uno de los elementos, se llevaron a cabo pruebas de ablación empleando la base de datos CMU como se puede apreciar en el Anexo 5.

.

V. Resultados

V.I. Prueba base de datos CMU

Para todas las pruebas de esta base de datos mediante el método descrito en la metodología, el tiempo de carga de los datos fue de 8.83 segundos en promedio.

V.I.1. Prueba de red neuronal densa

Para esta ejecución se puso un límite de 24000 iteraciones y la prueba termino en 3000, cada vez que se obtuvo un mejor resultado para los datos de prueba se guardó la red al terminar el entrenamiento.

Se obtuvieron los resultados ilustrados en la Fig. 27 en la cual podemos ver los resultados para los datos de prueba y entrenamiento para el PCK3D y la distancia euclidiana durante el entrenamiento hasta donde se encontró un mejor resultado para los datos de prueba.

El porcentaje de puntos correctos es muy bajo incluso con la distancia euclidiana es menor al umbral de 150 mm, esta es una buena muestra de que el 3DPCK es una forma más robusta de determinar qué tan buena es la predicción generada por las redes neuronales.

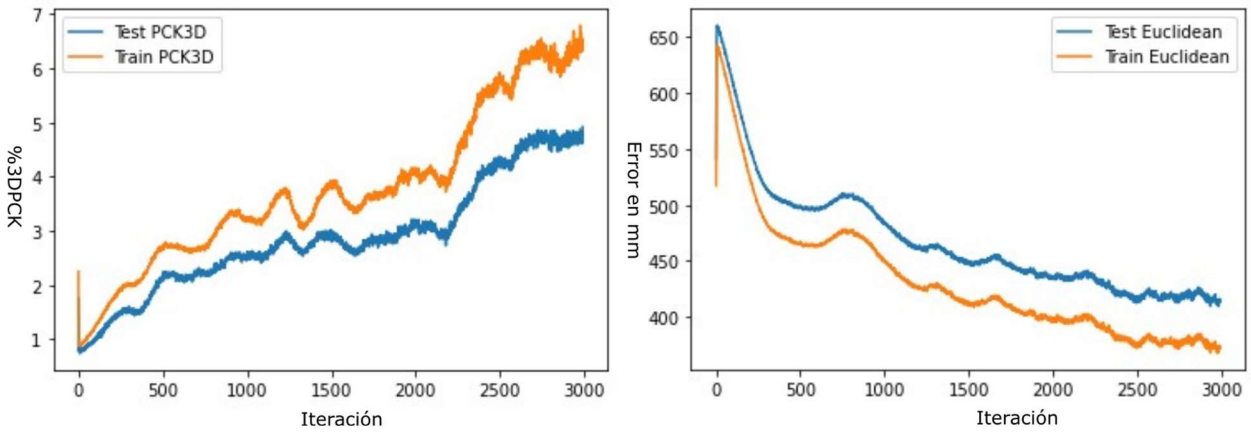

Fig. 27 Gráficas de 3DPCK y distancia euclidiana del modelo durante su entrenamiento.

V.I.2. Prueba de red neuronal convolucional

Se hicieron pruebas con una red neuronal convolucional para intentar mejorar el resultado de la red densa cambiando el tipo de capa, la prueba se ejecutó con un máximo de 24000 iteraciones y la prueba terminó en 11500 iteraciones, cada vez que se los datos de prueba para la métrica de 3DPCK mejoraban, se guardaba la red neuronal. Al terminar se muestran los valores del 3DPCK y la distancia euclidiana ilustrados en la Fig. 28.

El hecho de que el rendimiento incrementara hasta casi 90% en la métrica de 3DPCK parece indicar que emplear redes convolucionales es una buena opción cuando se intenta modelar secuencias de movimiento, por esta razón se empleó en los 2 siguientes experimentos, sin embargo este carece de las características de un modelo generativo.

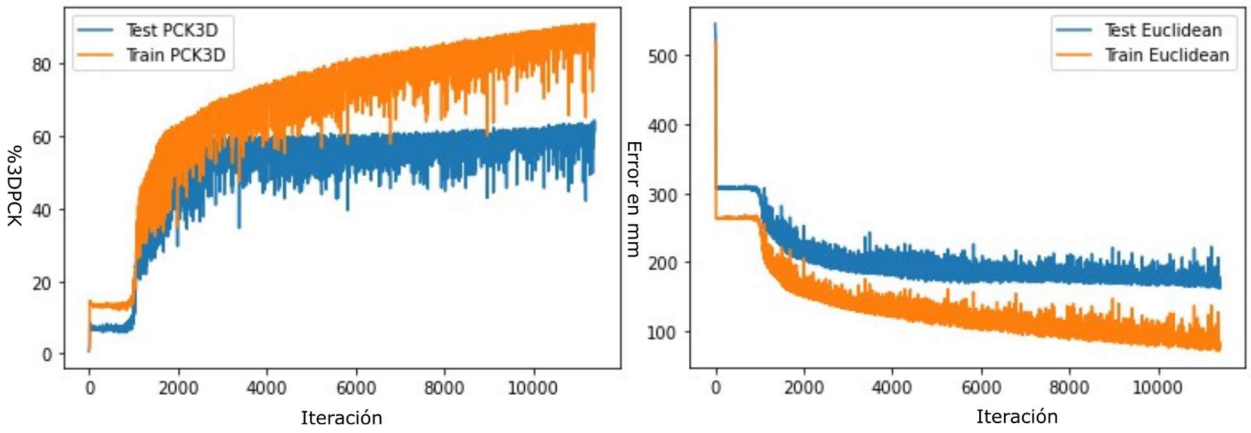

Fig. 28 Gráficas de 3DPCK y distancia euclidiana del modelo durante su entrenamiento.

V.I.3. Prueba de red neuronal AC-CGAN

En el tercer experimento se probó con una red adversarial para conseguir que las salidas de la red incrementaran su variedad, ya que a red convolucional tuvo un buen resultado por lo que esta se usara una similar como generador, los resultados del entrenamiento que tenía como máximo 24000 iteraciones y se detuvo prematuramente ya que dejó de mejorar el 3DPCK de los datos de prueba durante el conteo de la paciencia de 3000 iteraciones como se muestran en la Fig. 29, donde podemos apreciar que la red generativa propuesta no consigue capturar la distribución de los datos que forma la base de datos de poses CMU.

Esto es porque las redes generativas tienen un equilibrio muy delicado, por lo que posteriormente se buscaron otras maneras de guiar el entrenamiento de la red más allá de la segmentación de las poses.

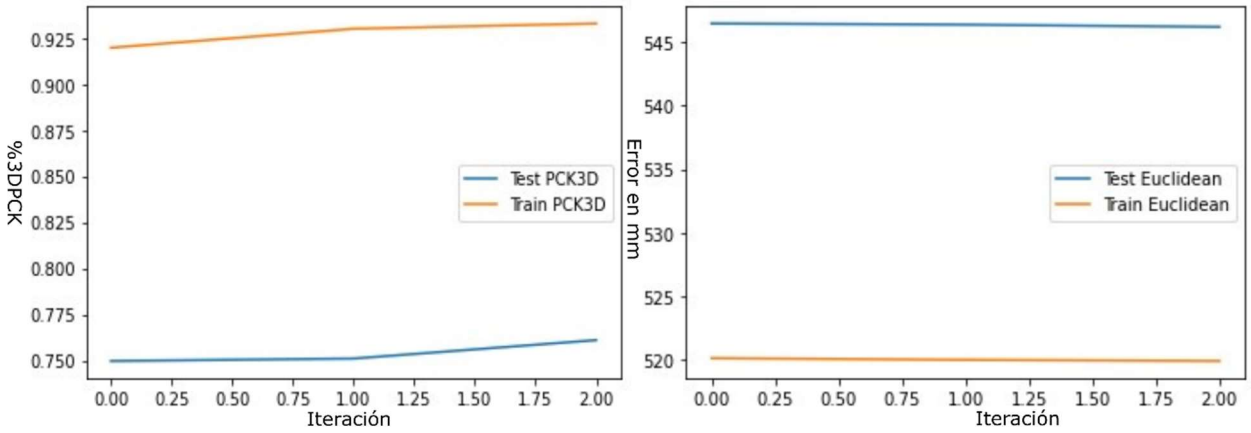

Fig. 29 Gráficas de 3DPCK y distancia euclidiana del modelo durante su entrenamiento.

V.I.4. Prueba de red neuronal AC-CGAN con MSE

En esta prueba se usaron 24000 iteraciones como máximo y se detuvo en la primera iteración, ya que no mejoro terminado el contador de la paciencia. Se puede ver que el resultado en la Fig. 30, el uso del MSE para guiar el entrenamiento del generador consiguió mejorar a casi el doble la métrica de 3DPCK comparado con la red AC-CGAN solo con el clasificador auxiliar de K-means, más adelante veremos como emplear métricas para guiar la posición de los puntos generados puede mejorar dramáticamente el resultado obtenido durante el entrenamiento de la GAN, al igual que el uso de capas recurrentes.

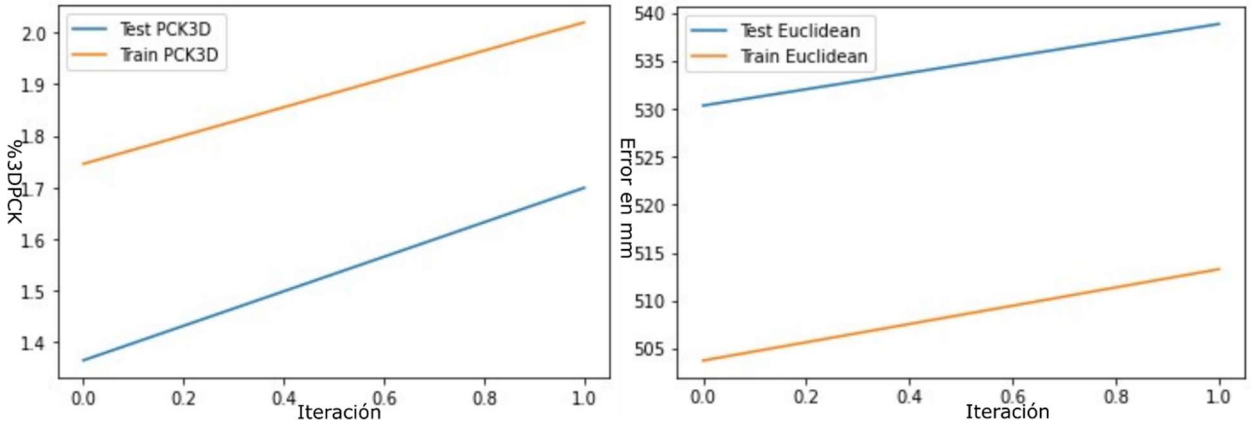

Fig. 30 Gráficas de 3DPCK y distancia euclidiana del modelo durante su entrenamiento.

V.I.5. Prueba de red neuronal recurrente AC-CGAN GRU

En esta prueba se usaron 24000 iteraciones como máximo por red y se detuvo en 9500 iteraciones en promedio para cada red. En la Fig. 31 se pueden observar los valores de métrica de 3DPCK y la distancia euclidiana para los datos de entrenamiento y los datos de prueba durante el entrenamiento para una de las redes, podemos observar que el entrenamiento benefició lentamente los valores de la métrica de 3DPCK para los datos de prueba aun cuando ya no hay un cambio significativo para los valores de la distancia euclidiana, esto se debe a que cuando se mide la distancia euclidiana es posible que haya algunos puntos muy cercanos a los valores esperados, mientras que otros se encuentran a una distancia mucho mayor y al momento de obtener el promedio de distancias se obtiene un valor engañoso, mientras que el 3DPCK es mucho más robusto a puntos alejados del valor esperado, pues la distancia euclidiana se mantiene constante entre las iteraciones. Los puntos más lejanos, en realidad se están aproximando cada vez más a su posición esperada, aunque algunos que estén muy cerca y en iteraciones anteriores se alejaron, se mantuvieron dentro del rango establecido por la métrica 3DPCK.

Aun así, se probó con el ensamble con otros modelos iguales para mejorar los valores de las métricas.

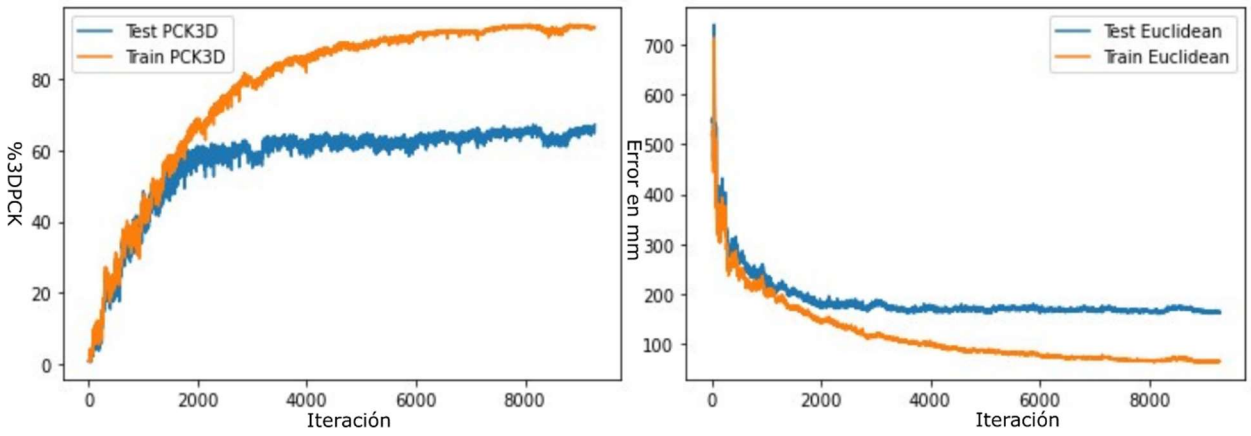

Fig. 31 Gráficas de 3DPCK y distancia euclidiana del modelo durante su entrenamiento.

En la Fig. 32, Fig. 33 y Fig. 34 podemos ver una comparación, respectivamente, entre el fotograma 1 que se encuentra en los datos de entrenamiento, el fotograma 903 y el fotograma 906 que se encuentran en los datos de prueba, podemos observar que en las figuras, donde a) es el movimiento real y b) el generado, la mayor parte de las poses fueron aprendidas correctamente, salvo algunas excepciones. Esto se debe a que la distribución de datos con la que se entrenó el discriminador no contenía suficiente información para que el generador pudiera producir un resultado apropiado para cada caso específico. Cada prueba tomó 1.08 horas en promedio, para un total de 5.23 horas en total para todo el ensamble.

Es fácil confirmar el hecho de que las distribuciones de los datos reales y los producidos por el generador del modelo, son de hecho diferentes mediante la observación de sus histogramas dado un punto, en el caso del punto que representa el cuello de la representación de la persona como nube de puntos en la Fig. 27, se puede apreciar la diferencia entre los datos reales (izquierda) y los datos generados por la red neuronal generadora (derecha) de las coordenadas de un punto en el cuello de la nube de puntos que representa a la persona. sin embargo para confirmar esta observaciones se llevó a cabo la prueba de Kolmogórov-Smirnov o K-S, este prueba es útil para comprar dos distribuciones que son nos normales como es en este caso, especialmente para las coordenadas en X de ellos datos generados, los resultados de esta prueba se encuentran en la Tabla 2.

Tabla 2 Resultados de la prueba K-S

Modelo	Máximo	Mínimo
Red neuronal recurrente	0.02597617	0.25132363

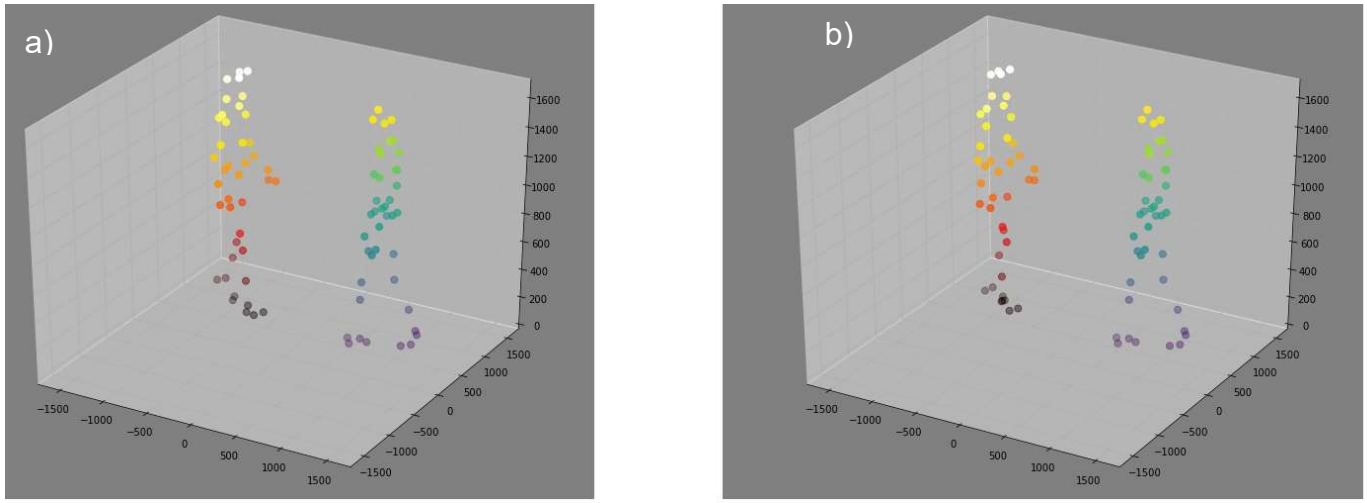

Fig. 32 Resultados sobre datos entrenados, donde el movimiento generado es adecuado para este caso con unidades en mm.

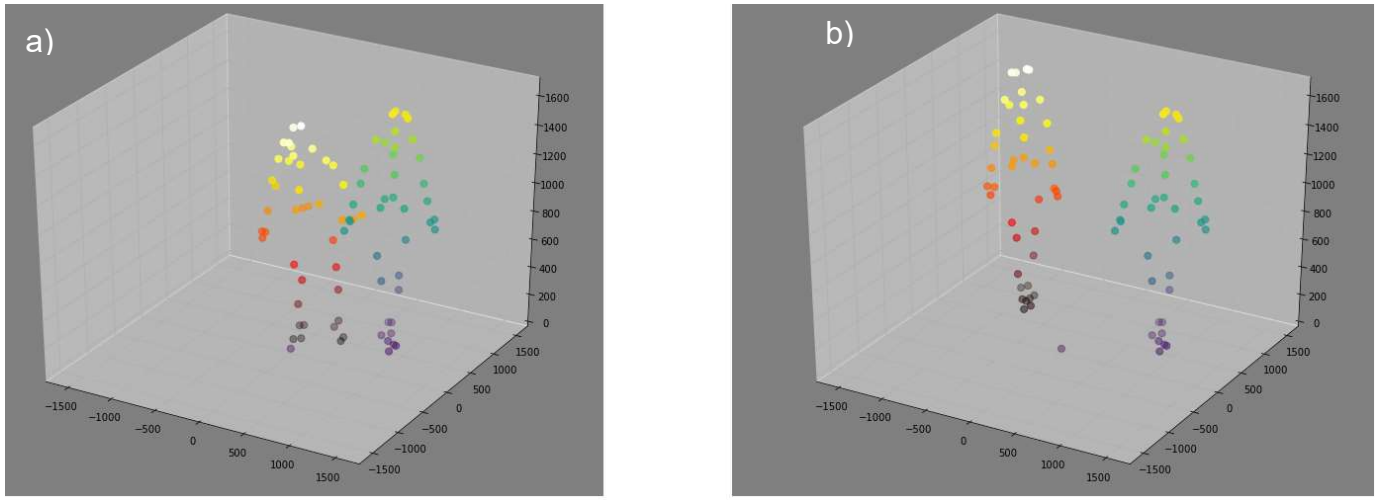

Fig. 33 Resultados sobre datos de prueba, donde el movimiento generado no es adecuado con unidades en mm.

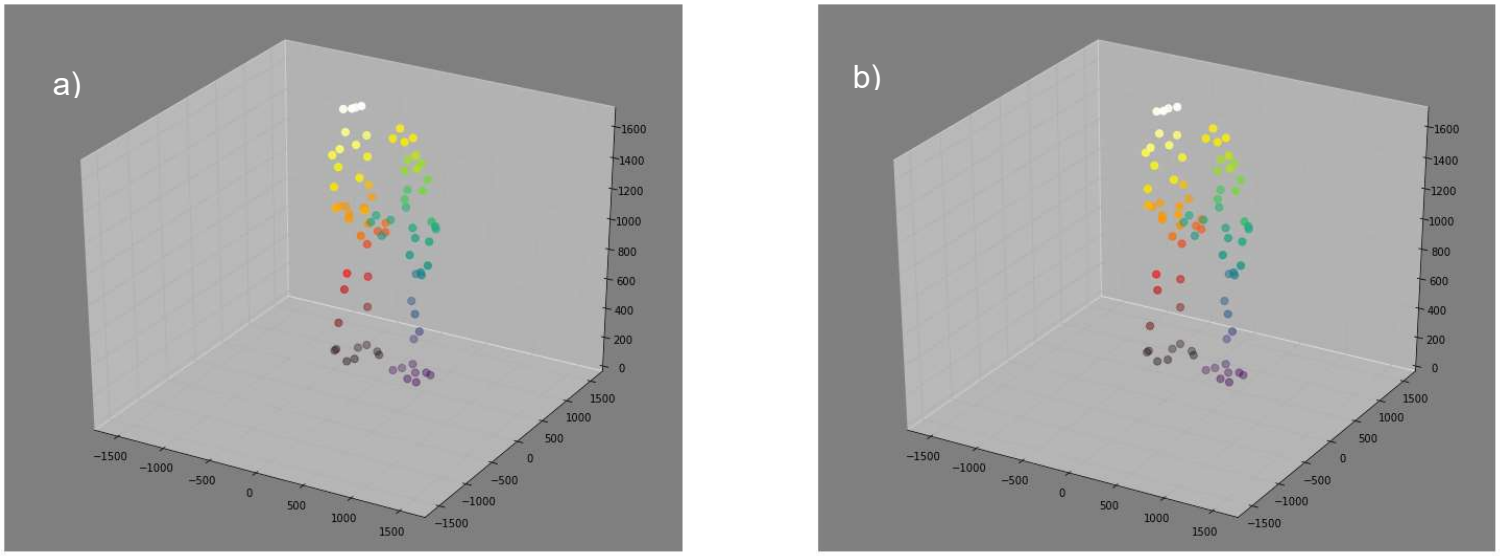

Fig. 34 Resultados sobre los datos de prueba, donde el movimiento generado es adecuado, con unidades en mm.

Se puede deducir que los resultados de la prueba K-S de la tabla 2, donde se tomaron el mejor y el peor de los resultados para cada una de las articulaciones y sus ejes de éstas que las distribuciones reales y generadas son ciertamente similares, aunque existen articulaciones donde las áreas de oportunidad para mejorar los resultados, es decir, un valor de K-S más pequeño.

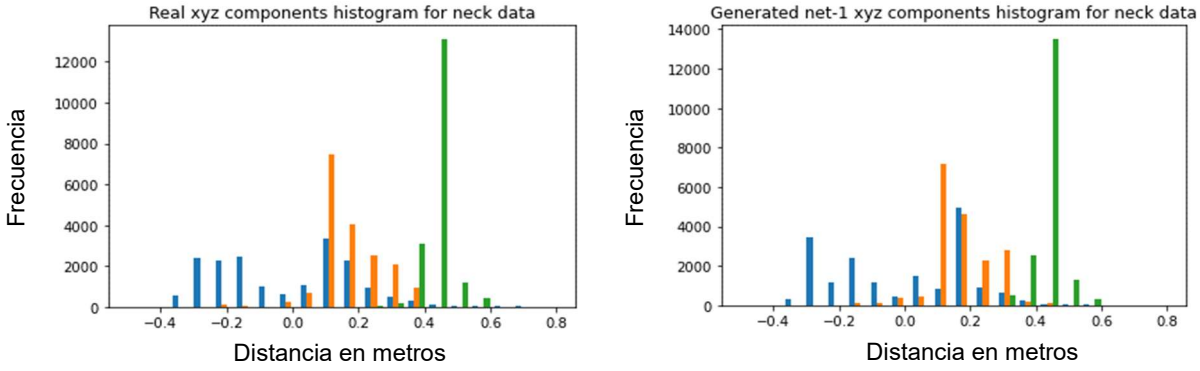

Fig. 35 Diferencia entre la distribución de los componentes X (azul),Y (verde) y Z (naranja) .

Para condensar los resultados de las pruebas, en la Tabla 3 Resultados de las pruebas con la base de datos CMU mostramos la comparación entre los métodos descritos en este trabajo para la base de datos CMU, en rojo se muestran los mejores resultados para cada una de las métricas escogidas.

En cada una de las pruebas mostradas en la Tabla 3 Resultados de las pruebas con la base de datos CMU, se puede apreciar que el mejor resultado para cada métrica en los datos de entrenamiento y prueba está resaltado en rojo, en la prueba de 3DPCK y distancia euclidiana el ensamble fue el que ofreció el mejor resultado; esto es esperado ya que agregar más redes al ensamble y promediar los resultados nos va a permitir modelar distintas características del movimiento con más precisión que una red individual.

En las pruebas de MMD, K-S y FID, algunas redes de este ensamble tuvieron mejores resultados, estos son bastante interesantes, ya que nos muestran que una red individual que pudiera tener más variedad de movimientos, aunque su 3DPCK no fuera el

esperado. Pues con la desventaja de que esto podría traer la generación de movimientos inesperados o súbitos, posiblemente esto debería examinarse en trabajos futuros. El IS no muestra ninguna diferencia entre las redes evaluadas debido a que las clases de cada una de las secuencias están correctamente balanceadas, haciendo que el Inception score sea muy bajo, incluso si el 3DPCK y la distancia euclidiana nos indican que las muestras generadas son de muy alta calidad (hasta 92% similares a movimientos humanos reales).

Tabla 3 Resultados de las pruebas con la base de datos CMU

Modelo	Entrenamiento						Prueba					
	3DPCK	Euclidiana	MMD	K-S	IS	FID	3DPCK	Euclidiana	MMD	K-S	IS	FID
Convolucional	81.6905	104.2202	0.0031	0.1189	1.0022	0.0334	55.1359	217.1226	0.0321	0.1520	1.0025	0.1913
Densa	5.7190	357.7911	0.0709	0.6064	1.0000	1.6945	3.5959	413.3953	0.1194	0.6083	1.0000	2.1829
AC CGAN GRU 1	92.2083	74.3214	**0.0008**	**0.0882**	1.0025	**0.0148**	56.0109	204.3793	0.0316	**0.1440**	**1.0027**	**0.1288**
AC CGAN GRU 3	69.5221	150.4359	0.0054	0.1436	1.0021	0.0524	51.8053	233.2641	0.0385	0.1740	1.0022	0.2700
AC CGAN GRU 3	81.5273	112.4877	0.0024	0.1139	1.0024	0.0270	55.1419	218.5316	0.0285	0.1640	1.0023	0.1903
AC CGAN GRU 4	87.5854	92.2579	0.0021	0.0978	1.0022	0.0245	53.4420	221.2050	0.0296	0.1530	1.0023	0.1936
AC CGAN GRU 5	89.6532	85.0854	0.0013	0.0968	**1.0026**	0.0224	57.1985	214.6689	0.0327	0.1524	1.0024	0.2188
Ensamble AC-CGAN-GRU	**92.3966**	**67.9100**	0.0011	0.0890	1.0022	0.0177	**63.6719**	**187.9192**	**0.0280**	0.1576	1.0020	0.2091
AC-CGAN	0.9477	684.3900	0.3179	0.6824	1.0000	3.1110	0.8830	718.4807	0.3551	0.6805	1.0000	3.4934
AC-CGAN-MSE	2.8614	496.3870	0.1661	0.5403	1.0014	1.9621	2.0918	551.6036	0.2209	0.5539	1.0015	2.4839

V.II. Prueba base de datos Kinect

Las pruebas con los datos recolectados donde se empleó el sensor Kinect, guardan una fuerte similitud con las pruebas efectuadas a la base de datos CMU, por lo que en esta sección únicamente se comentan brevemente algunos aspectos que diferencian las dos bases de datos usadas en este trabajo, el tiempo de carga de la base de datos Kinect fue de 8.84 segundos en promedio.

V.II.1. Prueba de red neuronal densa

Para esta ejecución se puso un límite de 24000 iteraciones y la prueba terminó en 2500, el tiempo de ejecución fue de 0.88 horas.

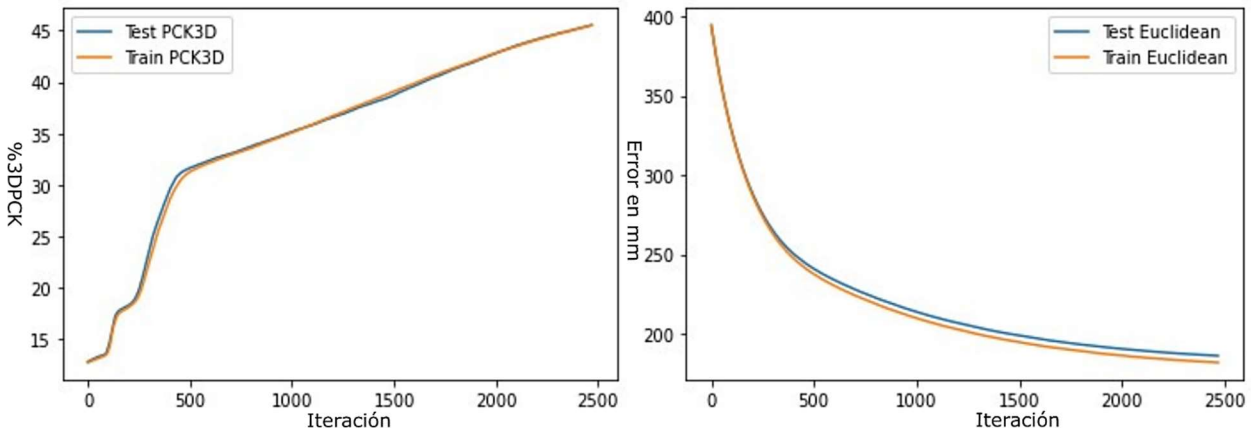

Fig. 36 Gráficas de 3DPCK y distancia euclidiana del modelo durante su entrenamiento.

V.II.2. Prueba de red neuronal convolucional

Para esta ejecución se puso un límite de 24000 iteraciones y la prueba terminó en la iteración 186, el tiempo de ejecución fue de 2.34 horas.

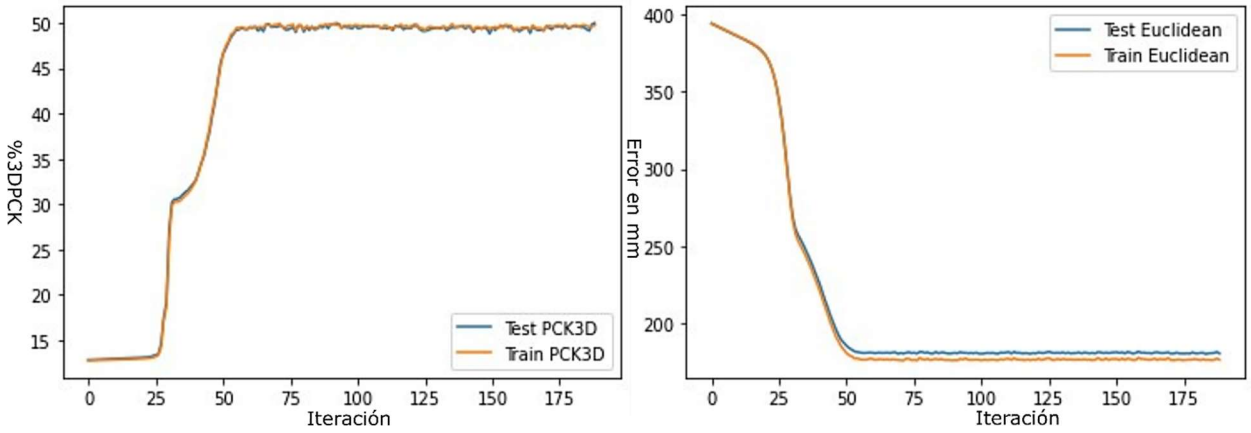

Fig. 37 Gráficas de 3DPCK y distancia euclidiana del modelo durante su entrenamiento.

V.II.3. Prueba de red neuronal AC-CGAN

Para esta ejecución se puso un límite de 24000 iteraciones y la prueba terminó en 9000 iteraciones, el tiempo de ejecución fue de 1.48 horas.

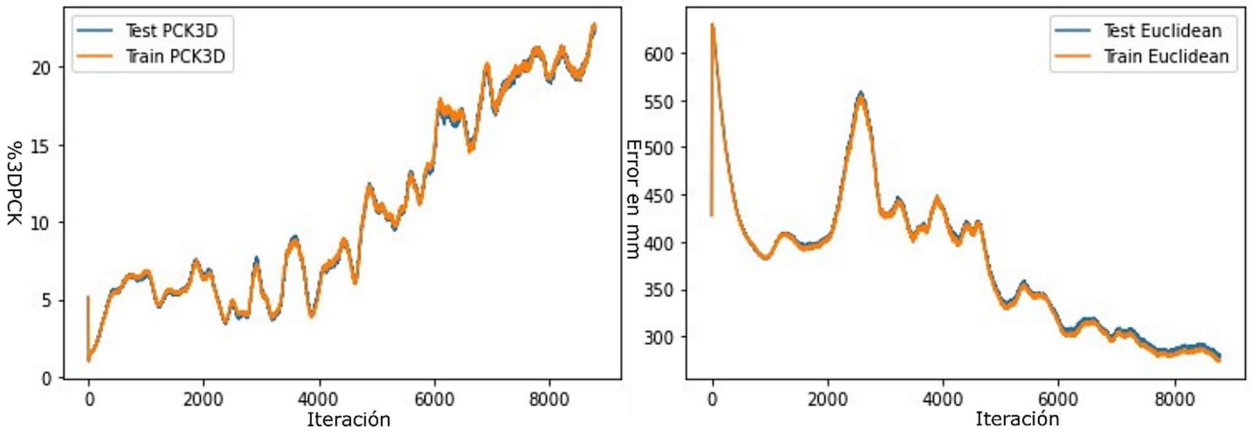

Fig. 38 Gráficas de 3DPCK y distancia euclidiana del modelo durante su entrenamiento.

V.II.4. Prueba de red neuronal AC-CGAN con MSE

Para esta ejecución se puso un límite de 24000 iteraciones y la prueba concluyó en 8500, el tiempo de ejecución fue de 1.36 horas.

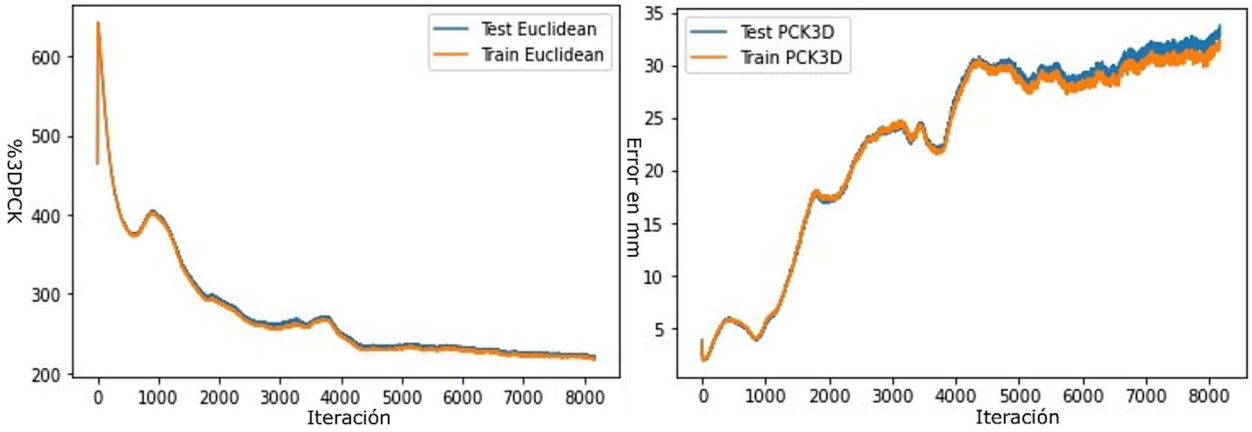

Fig. 39 Gráficas de 3DPCK y distancia euclidiana del modelo durante su entrenamiento.

V.II.5. Prueba de red neuronal recurrente AC-CGAN GRU

Para esta ejecución se puso un límite de 24000 iteraciones y la prueba terminó en 2500, el tiempo de ejecución promedio para cada red del ensamble fue de 1.28 horas. En la Fig. 40 podemos ver el valor de la distancia euclidiana y la métrica 3DPCK durante el entrenamiento del generador del modelo GAN.

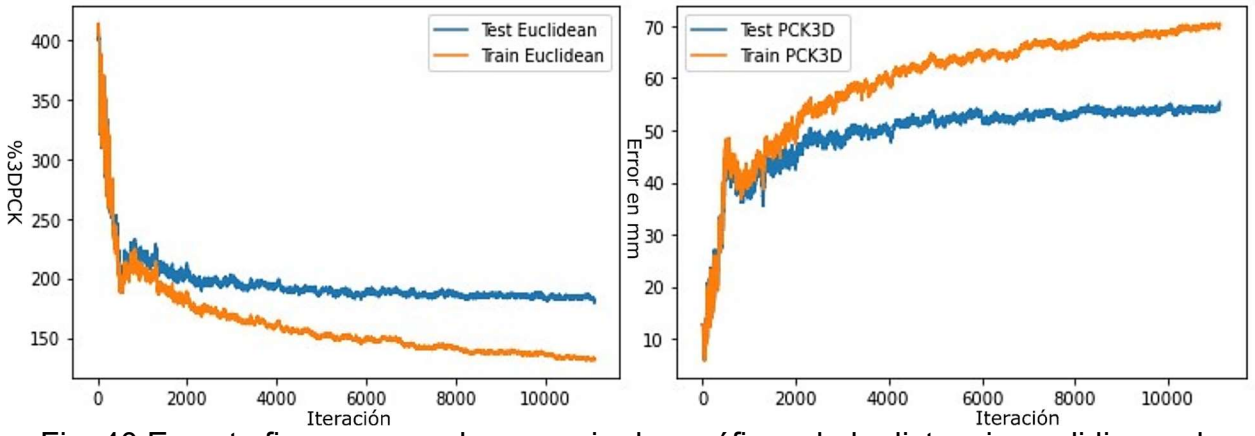

Fig. 40 En esta figura se pueden apreciar las gráficas de la distancia euclidiana y la métrica 3DPCK a través del entrenamiento de una de las redes del ensamble.

En la Fig. 41, sin unidades ya que los dato fueron normalizados, se puede apreciar del lado izquierdo la pose real que fue capturada durante las sesiones donde se obtuvieron los movimientos con ayuda de los participantes, del lado derecho se puede apreciar el movimiento generado por la red neuronal a partir de la etiqueta de entrada, por inspección podemos decir que ambas son similares desde la vista frontal desde la cual se dibujan.

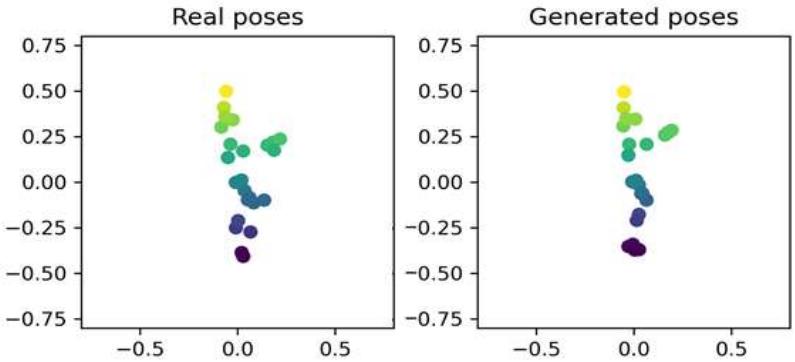

Fig. 41 En la figura se pueden apreciar del lado izquierdo las poses tomadas y del lado derecho las poses reales.

En la Tabla 4 podemos apreciar que la calidad de las mientras, medida en el IS es superior para el ensamble de redes neuronales, ya que puede cubrir una sección más amplia de la distribución de los datos que se le muestran al discriminador, mientras que el FID que representa la diferencia entre las propiedades estadísticas de la distribuciones generadas y las reales es más bajo para uno de los modelos entrenados, se puede deberse a que generaliza mejor algunas de las secciones en la distribución que las demás redes no consiguieron modelar. Al mismo tiempo, las métricas de similaridad como distancia euclidiana y la 3DPCK muestran que los datos producidos por el generador son semejantes a los de la base de datos, tanto de entrenamiento como de prueba.

Tabla 4 Resultados de las pruebas con la base de datos Kinect

Modelo	Entrenamiento						Prueba					
	3DPCK	Euclidiana	MMD	K-S	IS	FID	3DPCK	euclidiana	MMD	K-S	IS	FID
Convolucional	49.0278	185.7921	0.0077	0.5507	1.0000	251.8198	51.3078	182.4810	0.0068	0.5706	1.0000	246.9334
Densa	48.7839	185.8641	0.0077	0.5638	1.0000	264.1390	51.0283	182.7436	0.0068	0.5804	1.0000	258.8530
AC CGAN GRU 1	69.2118	134.2185	0.0006	0.0857	6.8667	8.2022	55.7040	180.2163	0.0029	0.1195	6.1546	16.6613
AC CGAN GRU 2	69.5617	132.7554	0.0005	0.0804	6.7160	6.5855	53.9189	187.0533	0.0035	0.1168	6.5158	16.8364
AC CGAN GRU 3	69.5633	133.1677	0.0008	0.0753	6.7105	6.7577	55.4125	181.4896	0.0034	0.1113	6.6752	14.6643
AC CGAN GRU 4	69.1779	135.2686	0.0006	0.0784	6.7969	7.6971	54.5409	182.9832	0.0027	0.1122	6.6379	15.5485
AC CGAN GRU 5	69.2715	134.8456	0.0008	0.0950	6.9954	8.5022	55.8773	179.4853	0.0027	0.1165	6.9015	14.7539
AC CGAN GRU 6	70.3697	130.7031	0.0006	0.0778	6.7457	**6.5242**	54.7193	185.2091	0.0026	0.1085	6.1017	15.1081
AC CGAN GRU 7	70.6467	130.9254	0.0005	**0.0732**	6.7576	6.6356	55.5131	180.7827	0.0029	**0.1082**	6.4763	**13.5401**
AC CGAN GRU 8	60.5265	157.8719	0.0013	0.1032	6.0854	12.7585	51.7285	190.8334	0.0035	0.1298	5.8709	18.2338
AC CGAN GRU 9	63.5296	149.8268	0.0008	0.0989	6.0863	13.4439	53.0418	188.1028	0.0037	0.1235	5.7847	20.1843
AC CGAN GRU 10	62.9275	150.7246	0.0010	0.1008	5.8970	12.6787	52.6121	189.7044	0.0032	0.1209	6.0852	18.7085
AC CGAN GRU 11	63.3596	150.5618	0.0012	0.0917	6.4238	11.2938	54.0456	185.6185	0.0030	0.1235	6.5764	17.9851
AC CGAN GRU 12	63.5686	149.3647	0.0012	0.0993	6.2746	10.0555	52.5399	187.4892	0.0038	0.1166	6.0614	18.5478
AC CGAN GRU 13	65.4752	144.9142	0.0008	0.0908	5.7948	10.4771	54.8074	182.9759	0.0030	0.1172	5.7503	16.3141
AC CGAN GRU 14	66.9438	141.1901	0.0010	0.0889	6.8939	9.0499	54.7273	183.4362	0.0034	0.1224	6.7268	16.2553
AC CGAN GRU 15	67.7915	139.0700	0.0007	0.0937	6.6756	9.7160	53.4195	186.8003	0.0033	0.1173	6.5624	16.3072
AC CGAN GRU 16	67.4589	139.8768	0.0008	0.0820	6.3256	9.1360	53.8718	186.8518	0.0032	0.1100	5.7854	17.4834
AC CGAN GRU 17	66.9498	140.3638	0.0007	0.0866	6.7789	9.2863	54.7002	183.6111	0.0038	0.1166	6.6384	16.5827
AC CGAN GRU 18	67.7258	138.1418	0.0009	0.0873	6.3308	10.4660	55.5422	185.1570	0.0034	0.1114	6.4381	18.7388
AC CGAN GRU 19	69.9873	132.6190	0.0007	0.0846	6.8455	7.0459	55.4681	184.2361	0.0027	0.1165	6.5968	15.0562
AC CGAN GRU 20	69.1530	134.6391	0.0007	0.0814	6.4894	6.9149	53.9008	187.6926	0.0031	0.1151	6.3078	14.0455
Ensamble	**80.1295**	**97.8135**	**0.0002**	0.1064	**7.6733**	9.3963	**64.5244**	**146.5136**	**0.0021**	0.1461	**7.5493**	18.3928
AC-GAN	24.5135	264.4448	0.0145	0.3477	2.5973	73.9618	25.3684	261.9852	0.0128	0.3523	2.5700	74.7580
AC-GAN-MSE	37.0427	222.0611	0.0108	0.3499	2.4996	96.0446	38.7829	219.3318	0.0095	0.3503	2.4726	94.8535

VI. Discusión

Es posible ver como esta propuesta de generador de poses sintéticas, mediante algoritmos generativos adversariales con clasificadores auxiliares, puede ser empleada para la generación de movimientos para actividades físicas deportivas; también para modelar el comportamiento de actividades cotidianas como lo son las de: saludar, bailar, imitar a una persona, jugar juegos como el de la silla musical [76], imitar los ademanes de un individuo que plática con otro y, probablemente, otros tipos de acciones.

El rendimiento de los algoritmos generativos para producir muestras nuevas se puede obtener al ver que se tienen porcentajes de puntos correctos similares a los porcentajes de puntos de algoritmos tradicionales de redes neuronales, ya que aun cuando la distancia euclidiana sea más corta para redes tradicionales, las redes generativas nos ayudan a producir una gran variedad de salidas distintas ya que no son idénticas entre ejecuciones.

Emplear transformadores [77] para llevar a cabo la inferencia de poses puede ser una opción interesante que deberá ser analizada en trabajos futuros, así como el uso de penalizaciones de gradientes [78] y otras técnicas de regularización que han ido apareciendo durante el desarrollo de este trabajo. Emplear video y audio como entrada sería otra adición importante a este algoritmo, nos podría ayudar a obtener resultados mucho más variados y de mucha más calidad a la hora de hacer la inferencia de las secuencias de poses.

VII. Conclusión

En este trabajo se implementó un método para recolectar, preprocesar, visualizar y generar secuencias movimientos para un sujeto A, a partir de otra secuencias de movimientos de un sujeto B, de tal manera que estos modelen la interacción que ocurre entre ambos sujetos tanto en ambientes cotidianos como deportivos.

Se emplearon diversas métricas para determinar la calidad de las secuencias de movimientos generadas, para medir la semejanza con los movimientos reales se empleó la distancia euclidiana (97.8135 mm) y el 3DPCK (97.8135%) como medidas espaciales, y el K-S (0.1064), el IS (7.6733) y la FID (9.3963) para calcular la similitud entre las distribución de secuencias de poses reales y las generadas, lo cual es útil para determinar también que tan apropiadas eran las poses generadas.

Para determinar qué tan apropiadas eran las poses también se llevó a cabo una inspección visual de las poses generadas mediante el sistema de visualización, que permitió, al mismo tiempo, hacer también una evaluación de carácter cualitativo.

A partir de los criterios seleccionados se determinó que los movimientos generados son ciertamente similares y suficientemente apropiados, cumpliendo así con la premisa propuesta en la hipótesis del proyecto, aun así, existen algunas opciones que se podrían explorar para mejorar la calidad de los resultados, como se menciona en la sección de discusión.

VIII. Bibliografía

[1] T. shinshii Daizdkyd, *Taishō Shinshū Daizōkyō*, SAT 2015. Tokyo, 2015. [Online]. Available: https://21dzk.l.u-tokyo.ac.jp/SAT/index_en.html

[2] M. Shahar, *The Shaolin monastery history, religion,and the chinese martial arts*, 1st ed. Honolulu: University of Hawai'i Press, 208AD. [Online]. Available: https://vk.com/doc278943367_490421157?hash=85d779fb24985b77d6&dl=3200c47a2f2a017690

[3] J. L. Broughton, "The Bodhidharma Anthology: The Earliest Records of Zen," *Journal of the American Oriental Society*, vol. 121, no. 1, pp. 152–153, 2001, doi: 10.2307/606768.

[4] Y. Chun, *116 Wing Chun Dummy Techniques.*, 1st ed. Hong Kong: Leung's Publications, 1981.

[5] "BotBoxer | Interactive Boxing Machines | VR/AR Punching Bag Arcade Games." https://botboxer.com/ (accessed Apr. 25, 2020).

[6] M. Loosemore, J. Lightfoot, D. Palmer-Green, I. Gatt, J. Bilzon, and C. Beardsley, "Boxing injury epidemiology in the Great Britain team: A 5-year surveillance study of medically diagnosed injury incidence and outcome," *British Journal of Sports Medicine*, vol. 49, no. 17, pp. 1100–1107, 2015, doi: 10.1136/bjsports-2015-094755.

[7] B. Bideau, R. Kulpa, N. Vignais, S. Brault, F. Multon, and C. Craig, "Using virtual reality to analyze sports performance," *IEEE Computer Graphics and Applications*, vol. 30, no. 2, pp. 14–21, 2010, doi: 10.1109/MCG.2009.134.

[8] L. Zhang *et al.*, "KaraKter: An autonomously interacting Karate Kumite character for VR-based training and research," *Computers and Graphics (Pergamon)*, vol. 72, pp. 59–69, 2018, doi: 10.1016/j.cag.2018.01.008.

[9] K. Petri *et al.*, "Training using virtual reality improves response behavior in karate kumite," *Sports Engineering*, vol. 22, no. 2, 2019, doi: 10.1007/s12283-019-0299-0.

[10] B. Tomlinson, "From linear to interactive animation," *Computers in Entertainment*, vol. 3, no. 1, p. 5, 2005, doi: 10.1145/1057270.1057282.

[11] L. G. Appelbaum and G. Erickson, "Sports vision training: A review of the state-of-the-art in digital training techniques," *International Review of Sport and Exercise Psychology*, vol. 11, no. 1, pp. 160–189, 2018, doi: 10.1080/1750984X.2016.1266376.

[12] A. Safonova, J. K. Hodgins, and N. S. Pollard, "Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces," *ACM SIGGRAPH 2004 Papers, SIGGRAPH 2004*, pp. 514–521, 2004, doi: 10.1145/1186562.1015754.

[13] B. T. Saragiotto, C. di Pierro, and A. D. Lopes, "Risk factors and injury prevention in elite athletes: a descriptive study of the opinions of physical therapists, doctors and trainers," *Brazilian Journal of Physical Therapy*, vol. 18, no. 2, pp. 137–143, 2014, doi: 10.1590/s1413-35552012005000147.

[14] A. Badets, Y. Blandin, and C. H. Shea, “Intention in motor learning through observation,” *Quarterly Journal of Experimental Psychology*, vol. 59, no. 2, pp. 377–386, 2006, doi: 10.1080/02724980443000773.
[15] A. Singh, “Analysis of force, time, energy, psychological demand and safety of common kicks in martial arts,” vol. 15423, 2017, doi: 10.31274/etd-180810-5045.
[16] Y. Nishimura, Y. Nakamura, and H. Ishiguro, “Human interaction behavior modeling using Generative Adversarial Networks,” *Neural Networks*, vol. 132, pp. 521–531, 2020, doi: 10.1016/j.neunet.2020.09.019.
[17] B. Wu, C. Liu, C. T. Ishi, and H. Ishiguro, “Modeling the Conditional Distribution of Co-Speech Upper Body Gesture Jointly Using Conditional-GAN and Unrolled-GAN,” *Electronics*, vol. 10, no. 3, p. 228, Jan. 2021, doi: 10.3390/electronics10030228.
[18] H. Y. Lee *et al.*, “Dancing to music,” *arXiv*, no. NeurIPS, pp. 1–11, 2019, doi: 10.4135/9781446251409.n4.
[19] L. Yu, J. Yu, and Q. Ling, “Deep Neural Network Based 3D Articulatory Movement Prediction Using Both Text and Audio Inputs,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2019, vol. 11295 LNCS, pp. 68–79. doi: doi.org/10.1007/978-3-030-05710-7_6.
[20] Z. Cao, G. Hidalgo, T. Simon, S. E. Wei, and Y. Sheikh, “OpenPose: Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 1, pp. 172–186, 2021, doi: 10.1109/TPAMI.2019.2929257.
[21] T. Simon, H. Joo, I. Matthews, and Y. Sheikh, “Hand keypoint detection in single images using multiview bootstrapping,” in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017, vol. 2017-Janua, pp. 4645–4653. doi: 10.1109/CVPR.2017.494.
[22] S. Xia, C. Wang, J. Chai, and J. Hodgins, “Realtime style transfer for unlabeled heterogeneous human motion,” *ACM Transactions on Graphics*, vol. 34, no. 4, pp. 1–10, Jul. 2015, doi: 10.1145/2766999.
[23] M. Lau, Z. Bar-Joseph, and J. Kuffner, “Modeling Spatial and Temporal Variation in Motion Data,” *ACM Transactions on Graphics*, vol. 28, no. 5, pp. 1–10, Dec. 2009, doi: 10.1145/1618452.1618517.
[24] R. Zhao and Q. Ji, “An adversarial hierarchical hidden markov model for human pose modeling and generation,” *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, pp. 2636–2643, 2018.
[25] K. Kania, M. Kowalski, and T. Trzciński, “TrajeVAE -- Controllable Human Motion Generation from Trajectories,” 2021, [Online]. Available: http://arxiv.org/abs/2104.00351
[26] R. Zhao, H. Su, and Q. Ji, “Bayesian adversarial human motion synthesis,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 6224–6233, 2020, doi: 10.1109/CVPR42600.2020.00626.
[27] M. Mirza and S. Osindero, “Conditional Generative Adversarial Nets,” pp. 1–7, 2014, [Online]. Available: https://arxiv.org/abs/1411.1784

[28] S. Starke, H. Zhang, T. Komura, and J. Saito, “Neural state machine for character-scene interactions,” *ACM Transactions on Graphics*, vol. 38, no. 6, pp. 1–14, Nov. 2019, doi: 10.1145/3355089.3356505.
[29] W. Xi, G. Devineau, F. Moutarde, and J. Yang, “Generative Model for Skeletal Human Movements Based on Conditional DC-GAN Applied to Pseudo-Images,” *Algorithms*, vol. 13, no. 12, p. 319, Dec. 2020, doi: 10.3390/a13120319.
[30] C. M. University, “CMU Graphics Lab Motion Capture Database.” http://mocap.cs.cmu.edu/
[31] L. A. Meerhoff, H. J. de Poel, T. W. D. Jowett, and C. Button, “Walking with avatars: Gait-related visual information for following a virtual leader,” *Human Movement Science*, vol. 66, no. December 2018, pp. 173–185, 2019, doi: 10.1016/j.humov.2019.04.003.
[32] Z. Wang, J. Chai, and S. Xia, “Combining Recurrent Neural Networks and Adversarial Training for Human Motion Synthesis and Control,” *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–1, 2019, doi: 10.1109/tvcg.2019.2938520.
[33] C. Zhou and W. Li, “Pose Comparison Based on Part Affinity Fields∗,” *2019 2nd IEEE International Conference on Information Communication and Signal Processing, ICICSP 2019*, pp. 519–523, 2019, doi: 10.1109/ICICSP48821.2019.8958597.
[34] F. Zhang, “Fast Human pose estimation,” *Studies in Systems, Decision and Control*, vol. 11, pp. 121–133, 2019, doi: 10.1007/978-3-319-11325-8_6.
[35] M. Pustišek, Y. Wei, Y. Sun, A. Umek, and A. Kos, “The role of technology for accelerated motor learning in sport,” *Personal and Ubiquitous Computing*, 2019, doi: 10.1007/s00779-019-01274-5.
[36] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, “Generative Adversarial Networks: An Overview,” *IEEE Signal Processing Magazine*, vol. 35, no. 1, pp. 53–65, 2018, doi: 10.1109/MSP.2017.2765202.
[37] H. Zhang, V. Sindagi, and V. M. Patel, “Image De-raining Using a Conditional Generative Adversarial Network,” *IEEE Transactions on Circuits and Systems for Video Technology*, pp. 1–1, 2019, doi: 10.1109/tcsvt.2019.2920407.
[38] A. S. Jo and Geun-Sik, “GOHAG: GANs Orchestration for Human Actions Generation,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2020, vol. 12033 LNAI, pp. 514–524. doi: 10.1007/978-3-030-41964-6_44.
[39] AltexSoft, “Preparing Your Dataset for Machine Learning: 8 Basic Techniques That Make Your Data Better,” *2017*, 2017. https://www.altexsoft.com/blog/datascience/preparing-your-dataset-for-machine-learning-8-basic-techniques-that-make-your-data-better/ (accessed May 24, 2020).
[40] Aito, “Aito.ai.” https://aito.ai/ (accessed May 24, 2020).
[41] A. Gonfalonieri, “How to Build A Data Set For Your Machine Learning Project,” *14-02-2019*. https://towardsdatascience.com/how-to-build-a-data-set-for-your-machine-learning-project-5b3b871881ac (accessed May 24, 2020).

[42] S. Yonemoto, A. Matsumoto, D. Arita, and R.-I. Taniguchi, “A real-time motion capture system with multiple camera fusion,” in *Proceedings 10th International Conference on Image Analysis and Processing*, pp. 600–605. doi: 10.1109/ICIAP.1999.797662.
[43] H. Yoshimoto, N. Date, and S. Yonemoto, “Vision-based real-time motion capture system using multiple cameras,” in *Proceedings of IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems, MFI2003.*, pp. 247–251. doi: 10.1109/MFI-2003.2003.1232665.
[44] V. Ganapathi, C. Plagemann, D. Koller, and S. Thrun, “Real time motion capture using a single time-of-flight camera,” in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Jun. 2010, pp. 755–762. doi: 10.1109/CVPR.2010.5540141.
[45] V. M. S. Limited, “Vicon iQ.” Vicon Motion Systems Limited, 2005.
[46] J. Shotton *et al.*, “Real-Time human pose recognition in parts from single depth images,” *Communications of the ACM*, vol. 56, no. 1, pp. 116–124, 2013, doi: 10.1145/2398356.2398381.
[47] “MEMS (MIcroElectroMechanical Systems).” https://www.etsist.upm.es/estaticos/ingeniatic/index.php/tecnologias/item/516-mems-microelectromechanical-systems.html (accessed Dec. 29, 2021).
[48] Y. Kim, “Desirable characteristics of learning companions,” *International Journal of Artificial Intelligence in Education*, vol. 17, no. 4, pp. 371–388, 2007.
[49] D. J. C. MacKay, *Information Theory, Inference, and Learning Algorithms*, vol. 1. New York, Ny: Cambridge University Press, 2003. [Online]. Available: https://www.biblio.com/book/information-theory-inference-learning-algorithms-david/d/1403942491
[50] Y. Shen, “Loss Functions for Binary Classification and Class,” 2005.
[51] R. M. Abarca, *The Concise Encyclopedia of Statistics*. New York, NY: Springer New York, 2008. doi: 10.1007/978-0-387-32833-1.
[52] M. Borovcnik, H.-J. Bentz, and R. Kapadia, *Machine learning a Probabilistic Perspective*. 1991. doi: 10.1007/978-94-011-3532-0_2.
[53] Q. Dang, J. Yin, B. Wang, and W. Zheng, “Deep learning based human pose estimation: A survey,” *Tsinghua Science and Technology*, vol. 24, no. 6, pp. 663–676, 2019, doi: 10.26599/TST.2018.9010100.
[54] Chakravarti and I. Mohan, *Handbook of Methods of Applied Statistics, Vol.1*, 1st ed. New York: John Wiley and Sons, 1967.
[55] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, “Improved techniques for training GANs,” *Advances in Neural Information Processing Systems*, pp. 2234–2242, 2016.
[56] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the Inception Architecture for Computer Vision,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-Decem, pp. 2818–2826, 2016, doi: 10.1109/CVPR.2016.308.
[57] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, “GANs trained by a two time-scale update rule converge to a local Nash equilibrium,” *Advances in Neural Information Processing Systems*, vol. 2017-Decem, no. Nips, pp. 6627–6638, 2017.

[58] “rfc4180.” https://datatracker.ietf.org/doc/html/rfc4180#page-1 (accessed Dec. 11, 2021).
[59] “Release Spyder 5.0.0 · spyder-ide/spyder · GitHub.” https://github.com/spyder-ide/spyder/releases/tag/v5.0.0 (accessed Dec. 11, 2021).
[60] “Python Release Python 3.6.0 | Python.org.” https://www.python.org/downloads/release/python-360/ (accessed Dec. 11, 2021).
[61] “NumPy.” https://numpy.org/ (accessed Feb. 02, 2022).
[62] “Matplotlib — Visualization with Python.” https://matplotlib.org/ (accessed Dec. 12, 2021).
[63] “Release TensorFlow 2.4.0 · tensorflow/tensorflow · GitHub.” https://github.com/tensorflow/tensorflow/releases/tag/v2.4.0 (accessed Dec. 11, 2021).
[64] “Release Keras 2.4.0 · keras-team/keras · GitHub.” https://github.com/keras-team/keras/releases/tag/2.4.0 (accessed Dec. 11, 2021).
[65] “Visual Studio 2019 version 16.11 Release Notes | Microsoft Docs.” https://docs.microsoft.com/en-us/visualstudio/releases/2019/release-notes (accessed Dec. 11, 2021).
[66] “Download Kinect for Windows SDK 2.0 from Official Microsoft Download Center.” https://www.microsoft.com/en-us/download/details.aspx?id=44561 (accessed Dec. 11, 2021).
[67] Python Software Foundation, “pickle — Python object serialization.” [Online]. Available: https://docs.python.org/3/library/pickle.html
[68] A. L. Maas, A. Y. Hannun, and A. Y. Ng, “Rectifier nonlinearities improve neural network acoustic models,” *in ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, vol. 28, 2013.
[69] M. D. Zeiler, “ADADELTA: An Adaptive Learning Rate Method,” 2012, [Online]. Available: http://arxiv.org/abs/1212.5701
[70] C. Y. Zhang, J. Hu, L. Yang, C. L. P. Chen, and Z. Yao, “Graph deconvolutional networks,” *Information Sciences*, vol. 518, pp. 330–340, 2020, doi: 10.1016/j.ins.2020.01.028.
[71] C. Garbin, X. Zhu, and O. Marques, “Dropout vs. batch normalization: an empirical study of their impact to deep learning,” *Multimedia Tools and Applications*, vol. 79, no. 19–20, pp. 12777–12815, May 2020, doi: 10.1007/s11042-019-08453-9.
[72] K. Sricharan, R. Bala1, M. Shreve, H. Ding, K. Saketh, and J. Sun, “Semi-supervised conditional GANs,” *arXiv*, pp. 1–23, 2017.
[73] H. Hewamalage, C. Bergmeir, and K. Bandara, “Recurrent Neural Networks for Time Series Forecasting: Current status and future directions,” *International Journal of Forecasting*, vol. 37, no. 1, pp. 388–427, 2021, doi: 10.1016/j.ijforecast.2020.06.008.
[74] R. Matteo and V. Giorgio, *Ensemble methods: a review*, no. May. 2001.
[75] A. Krogh and P. Sollich, “Statistical mechanics of ensemble learning,” *Physical Review E - Statistical Physics, Plasmas, Fluids, and Related Interdisciplinary Topics*, vol. 55, no. 1, pp. 811–825, 1997, doi: 10.1103/PhysRevE.55.811.
[76] A. & Méndez Giménez and C. Méndez Giménez, *Los Juegos En El Curriculum De La Educacion Fisica (5ª Ed.*, 5ª ED. Paidotribo, 2007. Accessed: Dec. 12,

2021. [Online]. Available: http://www.paidotribo.com/es/juegos/353-juegos-en-el-curriculum-de-la-ef-los.html
[77] G. Zerveas, S. Jayaraman, D. Patel, A. Bhamidipaty, and C. Eickhoff, “A Transformer-based Framework for Multivariate Time Series Representation Learning,” *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 2114–2124, Aug. 2021, doi: 10.1145/3447548.3467401.
[78] S. Pei, R. Y. da Xu, and G. Meng, “dp-GAN : Alleviating Mode Collapse in GAN via Diversity Penalty Module,” 2021, [Online]. Available: http://arxiv.org/abs/2108.02353

IX. Anexos

Anexo 1

Parámetros modelo denso

Layer (type)	Output Shape	Param #
input_48 (InputLayer)	[(None, 3, 123)]	0
flatten_25 (Flatten)	(None, 369)	0
dense_32 (Dense)	(None, 369)	136530
leaky_re_lu_125 (LeakyReLU)	(None, 369)	0
dense_33 (Dense)	(None, 369)	136530
leaky_re_lu_126 (LeakyReLU)	(None, 369)	0
dense_34 (Dense)	(None, 369)	136530
leaky_re_lu_127 (LeakyReLU)	(None, 369)	0
dense_35 (Dense)	(None, 369)	136530
leaky_re_lu_128 (LeakyReLU)	(None, 369)	0
dense_36 (Dense)	(None, 369)	136530
leaky_re_lu_129 (LeakyReLU)	(None, 369)	0
dense_37 (Dense)	(None, 369)	136530
leaky_re_lu_130 (LeakyReLU)	(None, 369)	0
dense_38 (Dense)	(None, 369)	136530
leaky_re_lu_131 (LeakyReLU)	(None, 369)	0
dense_39 (Dense)	(None, 369)	136530
leaky_re_lu_132 (LeakyReLU)	(None, 369)	0
dense_40 (Dense)	(None, 369)	136530
leaky_re_lu_133 (LeakyReLU)	(None, 369)	0
dense_41 (Dense)	(None, 369)	136530
leaky_re_lu_134 (LeakyReLU)	(None, 369)	0
reshape_29 (Reshape)	(None, 3, 123)	0

Total params: 1,365,300
Trainable params: 1,365,300
Non-trainable params: 0

Densa Model parameters:
{'batch_input_shape': (None, 3, 123), 'dtype': 'float32', 'sparse': False, 'ragged': False, 'name': 'input_144'}
{'name': 'flatten_76', 'trainable': True, 'dtype': 'float32', 'data_format': 'channels_last'}
{'name': 'dense_107', 'trainable': True, 'dtype': 'float32', 'units': 369, 'activation': 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'RandomUniform', 'config': {'minval': -0.0001, 'maxval': 0.0001, 'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None}
{'name': 'leaky_re_lu_385', 'trainable': True, 'dtype': 'float32', 'alpha': 0.30000001192092896}
{'name': 'dense_108', 'trainable': True, 'dtype': 'float32', 'units': 369, 'activation': 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'RandomUniform', 'config': {'minval': -0.0001, 'maxval': 0.0001, 'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None}
{'name': 'leaky_re_lu_386', 'trainable': True, 'dtype': 'float32', 'alpha': 0.30000001192092896}
{'name': 'dense_109', 'trainable': True, 'dtype': 'float32', 'units': 369, 'activation': 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'RandomUniform', 'config': {'minval': -0.0001, 'maxval': 0.0001, 'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None}
{'name': 'leaky_re_lu_387', 'trainable': True, 'dtype': 'float32', 'alpha': 0.30000001192092896}
{'name': 'dense_110', 'trainable': True, 'dtype': 'float32', 'units': 369, 'activation': 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'RandomUniform', 'config': {'minval': -0.0001, 'maxval': 0.0001, 'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None}
{'name': 'leaky_re_lu_388', 'trainable': True, 'dtype': 'float32', 'alpha': 0.30000001192092896}
{'name': 'dense_111', 'trainable': True, 'dtype': 'float32', 'units': 369, 'activation': 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'RandomUniform', 'config': {'minval': -0.0001, 'maxval': 0.0001, 'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None}
{'name': 'leaky_re_lu_389', 'trainable': True, 'dtype': 'float32', 'alpha': 0.30000001192092896}
{'name': 'dense_112', 'trainable': True, 'dtype': 'float32', 'units': 369, 'activation': 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'RandomUniform', 'config': {'minval': -0.0001, 'maxval': 0.0001, 'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None}
{'name': 'leaky_re_lu_390', 'trainable': True, 'dtype': 'float32', 'alpha': 0.30000001192092896}
{'name': 'dense_113', 'trainable': True, 'dtype': 'float32', 'units': 369, 'activation': 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'RandomUniform', 'config': {'minval': -0.0001, 'maxval': 0.0001, 'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None}
{'name': 'leaky_re_lu_391', 'trainable': True, 'dtype': 'float32', 'alpha': 0.30000001192092896}

{'name': 'dense_114', 'trainable': True, 'dtype': 'float32', 'units': 369, 'activation': 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'RandomUniform', 'config': {'minval': -0.0001, 'maxval': 0.0001, 'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None}
{'name': 'leaky_re_lu_392', 'trainable': True, 'dtype': 'float32', 'alpha': 0.30000001192092896}
{'name': 'dense_115', 'trainable': True, 'dtype': 'float32', 'units': 369, 'activation': 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'RandomUniform', 'config': {'minval': -0.0001, 'maxval': 0.0001, 'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None}
{'name': 'leaky_re_lu_393', 'trainable': True, 'dtype': 'float32', 'alpha': 0.30000001192092896}
{'name': 'dense_116', 'trainable': True, 'dtype': 'float32', 'units': 369, 'activation': 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'RandomUniform', 'config': {'minval': -0.0001, 'maxval': 0.0001, 'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None}
{'name': 'leaky_re_lu_394', 'trainable': True, 'dtype': 'float32', 'alpha': 0.30000001192092896}
{'name': 'reshape_88', 'trainable': True, 'dtype': 'float32', 'target_shape': (3, 123)}

Anexo 2

Parámetros modelo convolucional

Layer (type)	Output Shape	Param #
input_43 (InputLayer)	[(None, 3, 123, 1)]	0
conv2d_transpose_12 (Conv2DT	(None, 3, 123, 32)	544
conv2d_transpose_13 (Conv2DT	(None, 3, 123, 32)	16416
leaky_re_lu_100 (LeakyReLU)	(None, 3, 123, 32)	0
conv2d_transpose_14 (Conv2DT	(None, 3, 123, 32)	16416
leaky_re_lu_101 (LeakyReLU)	(None, 3, 123, 32)	0
conv2d_transpose_15 (Conv2DT	(None, 3, 123, 32)	16416
leaky_re_lu_102 (LeakyReLU)	(None, 3, 123, 32)	0
conv2d_transpose_16 (Conv2DT	(None, 3, 123, 32)	16416
leaky_re_lu_103 (LeakyReLU)	(None, 3, 123, 32)	0
conv2d_transpose_17 (Conv2DT	(None, 3, 123, 32)	16416
leaky_re_lu_104 (LeakyReLU)	(None, 3, 123, 32)	0
conv2d_transpose_18 (Conv2DT	(None, 3, 123, 32)	16416
leaky_re_lu_105 (LeakyReLU)	(None, 3, 123, 32)	0
conv2d_transpose_19 (Conv2DT	(None, 3, 123, 32)	16416
leaky_re_lu_106 (LeakyReLU)	(None, 3, 123, 32)	0
conv2d_transpose_20 (Conv2DT	(None, 3, 123, 32)	16416
leaky_re_lu_107 (LeakyReLU)	(None, 3, 123, 32)	0
conv2d_transpose_21 (Conv2DT	(None, 3, 123, 32)	16416
leaky_re_lu_108 (LeakyReLU)	(None, 3, 123, 32)	0
conv2d_transpose_22 (Conv2DT	(None, 3, 123, 32)	16416
leaky_re_lu_109 (LeakyReLU)	(None, 3, 123, 32)	0
flatten_21 (Flatten)	(None, 11808)	0
dense_31 (Dense)	(None, 369)	4357521
leaky_re_lu_110 (LeakyReLU)	(None, 369)	0
reshape_26 (Reshape)	(None, 3, 123)	0

Total params: 4,522,225
Trainable params: 4,522,225
Non-trainable params: 0


```
{'batch_input_shape': (None, 3, 123, 1), 'dtype': 'float32', 'sparse': False, 'ragged': False, 'name': 'input_191'}
```

```
{'name': 'conv2d_transpose_81', 'trainable': True, 'dtype': 'float32', 'filters': 32, 'kernel_size': (4, 4), 'strides': (1, 1), 'padding': 'same', 'data_format': 'channels_last', 'dilation_rate': (1, 1), 'groups': 1, 'activation': 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None, 'output_padding': None}
```

{'name': 'conv2d_transpose_82', 'trainable': True, 'dtype': 'float32', 'filters': 32, 'kernel_size': (4, 4), 'strides': (1, 1), 'padding': 'same', 'data_format': 'channels_last', 'dilation_rate': (1, 1), 'groups': 1, 'activation': 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None, 'output_padding': None}
{'name': 'leaky_re_lu_509', 'trainable': True, 'dtype': 'float32', 'alpha': 0.30000001192092896}
{'name': 'conv2d_transpose_83', 'trainable': True, 'dtype': 'float32', 'filters': 32, 'kernel_size': (4, 4), 'strides': (1, 1), 'padding': 'same', 'data_format': 'channels_last', 'dilation_rate': (1, 1), 'groups': 1, 'activation': 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None, 'output_padding': None}
{'name': 'leaky_re_lu_510', 'trainable': True, 'dtype': 'float32', 'alpha': 0.30000001192092896}
{'name': 'conv2d_transpose_84', 'trainable': True, 'dtype': 'float32', 'filters': 32, 'kernel_size': (4, 4), 'strides': (1, 1), 'padding': 'same', 'data_format': 'channels_last', 'dilation_rate': (1, 1), 'groups': 1, 'activation': 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None, 'output_padding': None}
{'name': 'leaky_re_lu_511', 'trainable': True, 'dtype': 'float32', 'alpha': 0.30000001192092896}
{'name': 'conv2d_transpose_85', 'trainable': True, 'dtype': 'float32', 'filters': 32, 'kernel_size': (4, 4), 'strides': (1, 1), 'padding': 'same', 'data_format': 'channels_last', 'dilation_rate': (1, 1), 'groups': 1, 'activation': 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None, 'output_padding': None}
{'name': 'leaky_re_lu_512', 'trainable': True, 'dtype': 'float32', 'alpha': 0.30000001192092896}
{'name': 'conv2d_transpose_86', 'trainable': True, 'dtype': 'float32', 'filters': 32, 'kernel_size': (4, 4), 'strides': (1, 1), 'padding': 'same', 'data_format': 'channels_last', 'dilation_rate': (1, 1), 'groups': 1, 'activation': 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None, 'output_padding': None}
{'name': 'leaky_re_lu_513', 'trainable': True, 'dtype': 'float32', 'alpha': 0.30000001192092896}
{'name': 'conv2d_transpose_87', 'trainable': True, 'dtype': 'float32', 'filters': 32, 'kernel_size': (4, 4), 'strides': (1, 1), 'padding': 'same', 'data_format': 'channels_last', 'dilation_rate': (1, 1), 'groups': 1, 'activation': 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None, 'output_padding': None}
{'name': 'leaky_re_lu_514', 'trainable': True, 'dtype': 'float32', 'alpha': 0.30000001192092896}
{'name': 'conv2d_transpose_88', 'trainable': True, 'dtype': 'float32', 'filters': 32, 'kernel_size': (4, 4), 'strides': (1, 1), 'padding': 'same', 'data_format': 'channels_last', 'dilation_rate': (1, 1), 'groups': 1, 'activation': 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None, 'output_padding': None}
{'name': 'leaky_re_lu_515', 'trainable': True, 'dtype': 'float32', 'alpha': 0.30000001192092896}
{'name': 'conv2d_transpose_89', 'trainable': True, 'dtype': 'float32', 'filters': 32, 'kernel_size': (4, 4), 'strides': (1, 1), 'padding': 'same', 'data_format': 'channels_last', 'dilation_rate': (1, 1), 'groups': 1, 'activation': 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None, 'output_padding': None}
{'name': 'leaky_re_lu_516', 'trainable': True, 'dtype': 'float32', 'alpha': 0.30000001192092896}
{'name': 'conv2d_transpose_90', 'trainable': True, 'dtype': 'float32', 'filters': 32, 'kernel_size': (4, 4), 'strides': (1, 1), 'padding': 'same', 'data_format': 'channels_last', 'dilation_rate': (1, 1), 'groups': 1, 'activation': 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None, 'output_padding': None}
{'name': 'leaky_re_lu_517', 'trainable': True, 'dtype': 'float32', 'alpha': 0.30000001192092896}

{'name': 'conv2d_transpose_91', 'trainable': True, 'dtype': 'float32', 'filters': 32, 'kernel_size': (4, 4), 'strides': (1, 1), 'padding': 'same', 'data_format': 'channels_last', 'dilation_rate': (1, 1), 'groups': 1, 'activation': 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None, 'output_padding': None}
{'name': 'leaky_re_lu_518', 'trainable': True, 'dtype': 'float32', 'alpha': 0.30000001192092896}
{'name': 'flatten_101', 'trainable': True, 'dtype': 'float32', 'data_format': 'channels_last'}
{'name': 'dense_148', 'trainable': True, 'dtype': 'float32', 'units': 369, 'activation': 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'RandomUniform', 'config': {'minval': -0.0001, 'maxval': 0.0001, 'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None}
{'name': 'leaky_re_lu_519', 'trainable': True, 'dtype': 'float32', 'alpha': 0.30000001192092896}
{'name': 'reshape_117', 'trainable': True, 'dtype': 'float32', 'target_shape': (3, 123)}

Anexo 3

Parámetros modelo AC-CGAN y MSE

Parámetros generador AC-CGAN

Model: "model_61"

Layer (type)	Output Shape	Param #	Connected to
input_37 (InputLayer)	[(None, 3, 123)]	0	
input_38 (InputLayer)	[(None, 3, 123)]	0	
reshape_23 (Reshape)	(None, 3, 123, 1)	0	input_37[0][0]
reshape_24 (Reshape)	(None, 3, 123, 1)	0	input_38[0][0]
conv2d_transpose_6 (Conv2DTrans	(None, 3, 123, 8)	80	reshape_23[0][0]
conv2d_transpose_7 (Conv2DTrans	(None, 3, 123, 8)	80	reshape_24[0][0]
leaky_re_lu_85 (LeakyReLU)	(None, 3, 123, 8)	0	conv2d_transpose_6[0][0]
leaky_re_lu_86 (LeakyReLU)	(None, 3, 123, 8)	0	conv2d_transpose_7[0][0]
dropout_6 (Dropout)	(None, 3, 123, 8)	0	leaky_re_lu_85[0][0]
dropout_7 (Dropout)	(None, 3, 123, 8)	0	leaky_re_lu_86[0][0]
batch_normalization_6	(BatchNor (None, 3, 123, 8)	32	dropout_6[0][0]
batch_normalization_7	(BatchNor (None, 3, 123, 8)	32	dropout_7[0][0]
multiply_2 (Multiply)	(None, 3, 123, 8) 0		batch_normalization_6[0][0] batch_normalization_7[0][0]
conv2d_transpose_8 (Conv2DTrans	(None, 3, 123, 4)	292	multiply_2[0][0]
leaky_re_lu_87 (LeakyReLU)	(None, 3, 123, 4)	0	conv2d_transpose_8[0][0]
dropout_8 (Dropout)	(None, 3, 123, 4)	0	leaky_re_lu_87[0][0]
batch_normalization_8 (BatchNor	(None, 3, 123, 4)	16	dropout_8[0][0]
conv2d_transpose_9 (Conv2DTrans	(None, 3, 123, 8)	296	batch_normalization_8[0][0]
leaky_re_lu_88 (LeakyReLU)	(None, 3, 123, 8)	0	conv2d_transpose_9[0][0]
dropout_9 (Dropout)	(None, 3, 123, 8)	0	leaky_re_lu_88[0][0]
batch_normalization_9 (BatchNor	(None, 3, 123, 8)	32	dropout_9[0][0]
conv2d_transpose_10 (Conv2DTran	(None, 3, 123, 16)	1168	batch_normalization_9[0][0]
leaky_re_lu_89 (LeakyReLU) conv2d_transpose_10[0][0]	(None, 3, 123, 16)	0	
dropout_10 (Dropout) leaky_re_lu_89[0][0]	(None, 3, 123, 16)	0	
batch_normalization_10 (BatchNo	(None, 3, 123, 16)	64	dropout_10[0][0]
conv2d_transpose_11 (Conv2DTran	(None, 3, 123, 32)	4640	batch_normalization_10[0][0]
leaky_re_lu_90 (LeakyReLU) conv2d_transpose_11[0][0]	(None, 3, 123, 32)	0	

dropout_11 (Dropout) leaky_re_lu_90[0][0]	(None, 3, 123, 32)	0	
batch_normalization_11 (BatchNo	(None, 3, 123, 32)	128	dropout_11[0][0]
flatten_18 (Flatten)	(None, 11808)	0	batch_normalization_11[0][0]
dense_18 (Dense)	(None, 369)	4357521	flatten_18[0][0]
leaky_re_lu_91 (LeakyReLU)	(None, 369)	0	dense_18[0][0]
reshape_25 (Reshape) leaky_re_lu_91[0][0]	(None, 3, 123)	0	

==========

Total params: 4,364,381
Trainable params: 4,364,229
Non-trainable params: 152

3DCPK Generator SC AC-CGAN GRU Model parameters:
{'batch_input_shape': (None, 3, 123), 'dtype': 'float32', 'sparse': False, 'ragged': False, 'name': 'input_185'}
{'batch_input_shape': (None, 3, 123), 'dtype': 'float32', 'sparse': False, 'ragged': False, 'name': 'input_186'}
{'name': 'reshape_114', 'trainable': True, 'dtype': 'float32', 'target_shape': (3, 123, 1)}
{'name': 'reshape_115', 'trainable': True, 'dtype': 'float32', 'target_shape': (3, 123, 1)}
{'name': 'conv2d_transpose_75', 'trainable': True, 'dtype': 'float32', 'filters': 8, 'kernel_size': (3, 3), 'strides': (1, 1), 'padding': 'same', 'data_format': 'channels_last', 'dilation_rate': (1, 1), 'groups': 1, 'activation': 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None, 'output_padding': None}
{'name': 'conv2d_transpose_76', 'trainable': True, 'dtype': 'float32', 'filters': 8, 'kernel_size': (3, 3), 'strides': (1, 1), 'padding': 'same', 'data_format': 'channels_last', 'dilation_rate': (1, 1), 'groups': 1, 'activation': 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None, 'output_padding': None}
{'name': 'leaky_re_lu_494', 'trainable': True, 'dtype': 'float32', 'alpha': 0.30000001192092896}
{'name': 'leaky_re_lu_495', 'trainable': True, 'dtype': 'float32', 'alpha': 0.30000001192092896}
{'name': 'dropout_42', 'trainable': True, 'dtype': 'float32', 'rate': 0.4, 'noise_shape': None, 'seed': None}
{'name': 'dropout_43', 'trainable': True, 'dtype': 'float32', 'rate': 0.4, 'noise_shape': None, 'seed': None}
{'name': 'batch_normalization_50', 'trainable': True, 'dtype': 'float32', 'axis': ListWrapper([3]), 'momentum': 0.7, 'epsilon': 0.001, 'center': True, 'scale': True, 'beta_initializer': {'class_name': 'Zeros', 'config': {}}, 'gamma_initializer': {'class_name': 'Ones', 'config': {}}, 'moving_mean_initializer': {'class_name': 'Zeros', 'config': {}}, 'moving_variance_initializer': {'class_name': 'Ones', 'config': {}}, 'beta_regularizer': None, 'gamma_regularizer': None, 'beta_constraint': None, 'gamma_constraint': None}
{'name': 'batch_normalization_51', 'trainable': True, 'dtype': 'float32', 'axis': ListWrapper([3]), 'momentum': 0.7, 'epsilon': 0.001, 'center': True, 'scale': True, 'beta_initializer': {'class_name': 'Zeros', 'config': {}}, 'gamma_initializer': {'class_name': 'Ones', 'config': {}}, 'moving_mean_initializer': {'class_name': 'Zeros', 'config': {}}, 'moving_variance_initializer': {'class_name': 'Ones', 'config': {}}, 'beta_regularizer': None, 'gamma_regularizer': None, 'beta_constraint': None, 'gamma_constraint': None}
{'name': 'multiply_14', 'trainable': True, 'dtype': 'float32'}
{'name': 'conv2d_transpose_77', 'trainable': True, 'dtype': 'float32', 'filters': 4, 'kernel_size': (3, 3), 'strides': (1, 1), 'padding': 'same', 'data_format': 'channels_last', 'dilation_rate': (1, 1), 'groups': 1, 'activation': 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None, 'output_padding': None}
{'name': 'leaky_re_lu_496', 'trainable': True, 'dtype': 'float32', 'alpha': 0.30000001192092896}

{'name': 'dropout_44', 'trainable': True, 'dtype': 'float32', 'rate': 0.6, 'noise_shape': None, 'seed': None}
{'name': 'batch_normalization_52', 'trainable': True, 'dtype': 'float32', 'axis': ListWrapper([3]), 'momentum': 0.7, 'epsilon': 0.001, 'center': True, 'scale': True, 'beta_initializer': {'class_name': 'Zeros', 'config': {}}, 'gamma_initializer': {'class_name': 'Ones', 'config': {}}, 'moving_mean_initializer': {'class_name': 'Zeros', 'config': {}}, 'moving_variance_initializer': {'class_name': 'Ones', 'config': {}}, 'beta_regularizer': None, 'gamma_regularizer': None, 'beta_constraint': None, 'gamma_constraint': None}
{'name': 'conv2d_transpose_78', 'trainable': True, 'dtype': 'float32', 'filters': 8, 'kernel_size': (3, 3), 'strides': (1, 1), 'padding': 'same', 'data_format': 'channels_last', 'dilation_rate': (1, 1), 'groups': 1, 'activation': 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None, 'output_padding': None}
{'name': 'leaky_re_lu_497', 'trainable': True, 'dtype': 'float32', 'alpha': 0.30000001192092896}
{'name': 'dropout_45', 'trainable': True, 'dtype': 'float32', 'rate': 0.6, 'noise_shape': None, 'seed': None}
{'name': 'batch_normalization_53', 'trainable': True, 'dtype': 'float32', 'axis': ListWrapper([3]), 'momentum': 0.7, 'epsilon': 0.001, 'center': True, 'scale': True, 'beta_initializer': {'class_name': 'Zeros', 'config': {}}, 'gamma_initializer': {'class_name': 'Ones', 'config': {}}, 'moving_mean_initializer': {'class_name': 'Zeros', 'config': {}}, 'moving_variance_initializer': {'class_name': 'Ones', 'config': {}}, 'beta_regularizer': None, 'gamma_regularizer': None, 'beta_constraint': None, 'gamma_constraint': None}
{'name': 'conv2d_transpose_79', 'trainable': True, 'dtype': 'float32', 'filters': 16, 'kernel_size': (3, 3), 'strides': (1, 1), 'padding': 'same', 'data_format': 'channels_last', 'dilation_rate': (1, 1), 'groups': 1, 'activation': 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None, 'output_padding': None}
{'name': 'leaky_re_lu_498', 'trainable': True, 'dtype': 'float32', 'alpha': 0.30000001192092896}
{'name': 'dropout_46', 'trainable': True, 'dtype': 'float32', 'rate': 0.6, 'noise_shape': None, 'seed': None}
{'name': 'batch_normalization_54', 'trainable': True, 'dtype': 'float32', 'axis': ListWrapper([3]), 'momentum': 0.7, 'epsilon': 0.001, 'center': True, 'scale': True, 'beta_initializer': {'class_name': 'Zeros', 'config': {}}, 'gamma_initializer': {'class_name': 'Ones', 'config': {}}, 'moving_mean_initializer': {'class_name': 'Zeros', 'config': {}}, 'moving_variance_initializer': {'class_name': 'Ones', 'config': {}}, 'beta_regularizer': None, 'gamma_regularizer': None, 'beta_constraint': None, 'gamma_constraint': None}
{'name': 'conv2d_transpose_80', 'trainable': True, 'dtype': 'float32', 'filters': 32, 'kernel_size': (3, 3), 'strides': (1, 1), 'padding': 'same', 'data_format': 'channels_last', 'dilation_rate': (1, 1), 'groups': 1, 'activation': 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None, 'output_padding': None}
{'name': 'leaky_re_lu_499', 'trainable': True, 'dtype': 'float32', 'alpha': 0.30000001192092896}
{'name': 'dropout_47', 'trainable': True, 'dtype': 'float32', 'rate': 0.6, 'noise_shape': None, 'seed': None}
{'name': 'batch_normalization_55', 'trainable': True, 'dtype': 'float32', 'axis': ListWrapper([3]), 'momentum': 0.7, 'epsilon': 0.001, 'center': True, 'scale': True, 'beta_initializer': {'class_name': 'Zeros', 'config': {}}, 'gamma_initializer': {'class_name': 'Ones', 'config': {}}, 'moving_mean_initializer': {'class_name': 'Zeros', 'config': {}}, 'moving_variance_initializer': {'class_name': 'Ones', 'config': {}}, 'beta_regularizer': None, 'gamma_regularizer': None, 'beta_constraint': None, 'gamma_constraint': None}
{'name': 'flatten_98', 'trainable': True, 'dtype': 'float32', 'data_format': 'channels_last'}
{'name': 'dense_135', 'trainable': True, 'dtype': 'float32', 'units': 369, 'activation': 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None}
{'name': 'leaky_re_lu_500', 'trainable': True, 'dtype': 'float32', 'alpha': 0.30000001192092896}
{'name': 'reshape_116', 'trainable': True, 'dtype': 'float32', 'target_shape': (3, 123)}

Parámetros discriminador AC-CGAN

Model: "model_66"

Layer (type)	Output Shape	Param #	Connected to
input_39 (InputLayer)	[(None, 3, 123)]	0	
input_40 (InputLayer)	[(None, 3, 123)]	0	
flatten_19 (Flatten)	(None, 369)	0	input_39[0][0]
flatten_20 (Flatten)	(None, 369)	0	input_40[0][0]
dense_19 (Dense)	(None, 369)	136530	flatten_19[0][0]
dense_20 (Dense)	(None, 369)	136530	flatten_20[0][0]
multiply_3 (Multiply)	(None, 369)	0	dense_19[0][0] dense_20[0][0]
dense_21 (Dense)	(None, 1476)	546120	multiply_3[0][0]
dense_25 (Dense)	(None, 1476)	546120	multiply_3[0][0]
leaky_re_lu_92 (LeakyReLU)	(None, 1476)	0	dense_21[0][0]
leaky_re_lu_96 (LeakyReLU)	(None, 1476)	0	dense_25[0][0]
dense_22 (Dense)	(None, 369)	545013	leaky_re_lu_92[0][0]
dense_26 (Dense)	(None, 369)	545013	leaky_re_lu_96[0][0]
leaky_re_lu_93 (LeakyReLU)	(None, 369)	0	dense_22[0][0]
leaky_re_lu_97 (LeakyReLU)	(None, 369)	0	dense_26[0][0]
dense_23 (Dense)	(None, 369)	136530	leaky_re_lu_93[0][0]
dense_27 (Dense)	(None, 369)	136530	leaky_re_lu_97[0][0]
leaky_re_lu_94 (LeakyReLU)	(None, 369)	0	dense_23[0][0]
leaky_re_lu_98 (LeakyReLU)	(None, 369)	0	dense_27[0][0]
dense_24 (Dense)	(None, 369)	136530	leaky_re_lu_94[0][0]
dense_28 (Dense)	(None, 369)	36530	leaky_re_lu_98[0][0]
leaky_re_lu_95 (LeakyReLU)	(None, 369)	0	dense_24[0][0]
leaky_re_lu_99 (LeakyReLU)	(None, 369)	0	dense_28[0][0]
dense_29 (Dense)	(None, 1)	370	leaky_re_lu_95[0][0]
dense_30 (Dense)	(None, 288)	106560	leaky_re_lu_99[0][0]

Total params: 3,108,376
Trainable params: 0
Non-trainable params: 3,108,376

Validity dicriminator AC-CGAN Model parameters:
{'batch_input_shape': (None, 3, 123), 'dtype': 'float32', 'sparse': False, 'ragged': False, 'name': 'input_187'}
{'batch_input_shape': (None, 3, 123), 'dtype': 'float32', 'sparse': False, 'ragged': False, 'name': 'input_188'}

{'name': 'flatten_99', 'trainable': True, 'dtype': 'float32', 'data_format': 'channels_last'}
{'name': 'flatten_100', 'trainable': True, 'dtype': 'float32', 'data_format': 'channels_last'}
{'name': 'dense_136', 'trainable': True, 'dtype': 'float32', 'units': 369, 'activation': 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None}
{'name': 'dense_137', 'trainable': True, 'dtype': 'float32', 'units': 369, 'activation': 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None}
{'name': 'multiply_15', 'trainable': True, 'dtype': 'float32'}
{'name': 'dense_138', 'trainable': True, 'dtype': 'float32', 'units': 1476, 'activation': 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None}
{'name': 'leaky_re_lu_501', 'trainable': True, 'dtype': 'float32', 'alpha': 0.30000001192092896}
{'name': 'dense_139', 'trainable': True, 'dtype': 'float32', 'units': 369, 'activation': 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None}
{'name': 'leaky_re_lu_502', 'trainable': True, 'dtype': 'float32', 'alpha': 0.30000001192092896}
{'name': 'dense_140', 'trainable': True, 'dtype': 'float32', 'units': 369, 'activation': 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None}
{'name': 'leaky_re_lu_503', 'trainable': True, 'dtype': 'float32', 'alpha': 0.30000001192092896}
{'name': 'dense_141', 'trainable': True, 'dtype': 'float32', 'units': 369, 'activation': 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None}
{'name': 'leaky_re_lu_504', 'trainable': True, 'dtype': 'float32', 'alpha': 0.30000001192092896}
{'name': 'dense_146', 'trainable': True, 'dtype': 'float32', 'units': 1, 'activation': 'sigmoid', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None}
K-means dicriminator AC-CGAN Model parameters:
{'batch_input_shape': (None, 3, 123), 'dtype': 'float32', 'sparse': False, 'ragged': False, 'name': 'input_187'}
{'batch_input_shape': (None, 3, 123), 'dtype': 'float32', 'sparse': False, 'ragged': False, 'name': 'input_188'}
{'name': 'flatten_99', 'trainable': True, 'dtype': 'float32', 'data_format': 'channels_last'}
{'name': 'flatten_100', 'trainable': True, 'dtype': 'float32', 'data_format': 'channels_last'}
{'name': 'dense_136', 'trainable': True, 'dtype': 'float32', 'units': 369, 'activation': 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None}
{'name': 'dense_137', 'trainable': True, 'dtype': 'float32', 'units': 369, 'activation': 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None}
{'name': 'multiply_15', 'trainable': True, 'dtype': 'float32'}

{'name': 'dense_142', 'trainable': True, 'dtype': 'float32', 'units': 1476, 'activation': 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None}
{'name': 'leaky_re_lu_505', 'trainable': True, 'dtype': 'float32', 'alpha': 0.30000001192092896}
{'name': 'dense_143', 'trainable': True, 'dtype': 'float32', 'units': 369, 'activation': 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None}
{'name': 'leaky_re_lu_506', 'trainable': True, 'dtype': 'float32', 'alpha': 0.30000001192092896}
{'name': 'dense_144', 'trainable': True, 'dtype': 'float32', 'units': 369, 'activation': 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None}
{'name': 'leaky_re_lu_507', 'trainable': True, 'dtype': 'float32', 'alpha': 0.30000001192092896}
{'name': 'dense_145', 'trainable': True, 'dtype': 'float32', 'units': 369, 'activation': 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None}
{'name': 'leaky_re_lu_508', 'trainable': True, 'dtype': 'float32', 'alpha': 0.30000001192092896}
{'name': 'dense_147', 'trainable': True, 'dtype': 'float32', 'units': 288, 'activation': 'softmax', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None}
[<KerasTensor: shape=(None, 1) dtype=float32 (created by layer 'dense_146')>, <KerasTensor: shape=(None, 288) dtype=float32 (created by layer 'dense_147')>]
[<KerasTensor: shape=(None, 1) dtype=float32 (created by layer 'model_293')>, <KerasTensor: shape=(None, 288) dtype=float32 (created by layer 'model_293')>]

Anexo 4

Parámetros modelo AC-CGAN GRU

Parámetros generador AC-CGAN GRU

Model: "model_42"

Layer (type)	Output Shape	Param #	Connected to
input_26 (InputLayer)	[(None, 3, 123)]	0	
input_25 (InputLayer)	[(None, 3, 123)]	0	
reshape_17 (Reshape)	(None, 3, 123)	0	input_26[0][0]
reshape_16 (Reshape)	(None, 3, 123)	0	input_25[0][0]
concatenate_8 (Concatenate)	(None, 3, 246)	0	reshape_17[0][0] reshape_16[0][0]
gru_24 (GRU)	(None, 3, 246)	364572	concatenate_8[0][0]
gru_25 (GRU)	(None, 3, 123)	136899	gru_24[0][0]
gru_26 (GRU)	(None, 3, 41)	20418	gru_25[0][0]
dense_4 (Dense)	(None, 3, 41)	1722	gru_26[0][0]
gru_27 (GRU)	(None, 3, 41)	10332	dense_4[0][0]
gru_28 (GRU)	(None, 3, 82)	30750	gru_27[0][0]
gru_29 (GRU)	(None, 3, 123)	76383	gru_28[0][0]

Total params: 641,076
Trainable params: 641,076
Non-trainable params: 0

generator AC-CGAN GRU Model parameters:
{'batch_input_shape': (None, 3, 123), 'dtype': 'float32', 'sparse': False, 'ragged': False, 'name': 'input_174'}
{'batch_input_shape': (None, 3, 123), 'dtype': 'float32', 'sparse': False, 'ragged': False, 'name': 'input_173'}
{'name': 'reshape_108', 'trainable': True, 'dtype': 'float32', 'target_shape': (3, 123)}
{'name': 'reshape_107', 'trainable': True, 'dtype': 'float32', 'target_shape': (3, 123)}
{'name': 'concatenate_48', 'trainable': True, 'dtype': 'float32', 'axis': 2}
{'name': 'gru_114', 'trainable': True, 'dtype': 'float32', 'return_sequences': True, 'return_state': False, 'go_backwards': False, 'stateful': False, 'unroll': False, 'time_major': False, 'units': 246, 'activation': 'tanh', 'recurrent_activation': 'sigmoid', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'recurrent_initializer': {'class_name':

'Orthogonal', 'config': {'gain': 1.0, 'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'recurrent_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'recurrent_constraint': None, 'bias_constraint': None, 'dropout': 0.0, 'recurrent_dropout': 0.0, 'implementation': 2, 'reset_after': True}

{'name': 'gru_115', 'trainable': True, 'dtype': 'float32', 'return_sequences': True, 'return_state': False, 'go_backwards': False, 'stateful': False, 'unroll': False, 'time_major': False, 'units': 123, 'activation': 'tanh', 'recurrent_activation': 'sigmoid', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'recurrent_initializer': {'class_name': 'Orthogonal', 'config': {'gain': 1.0, 'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'recurrent_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'recurrent_constraint': None, 'bias_constraint': None, 'dropout': 0.0, 'recurrent_dropout': 0.0, 'implementation': 2, 'reset_after': True}

{'name': 'gru_116', 'trainable': True, 'dtype': 'float32', 'return_sequences': True, 'return_state': False, 'go_backwards': False, 'stateful': False, 'unroll': False, 'time_major': False, 'units': 41, 'activation': 'tanh', 'recurrent_activation': 'sigmoid', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'recurrent_initializer': {'class_name': 'Orthogonal', 'config': {'gain': 1.0, 'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'recurrent_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'recurrent_constraint': None, 'bias_constraint': None, 'dropout': 0.0, 'recurrent_dropout': 0.0, 'implementation': 2, 'reset_after': True}

{'name': 'dense_121', 'trainable': True, 'dtype': 'float32', 'units': 41, 'activation': 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None}

{'name': 'gru_117', 'trainable': True, 'dtype': 'float32', 'return_sequences': True, 'return_state': False, 'go_backwards': False, 'stateful': False, 'unroll': False, 'time_major': False, 'units': 41, 'activation': 'tanh', 'recurrent_activation': 'sigmoid', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'recurrent_initializer': {'class_name': 'Orthogonal', 'config': {'gain': 1.0, 'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'recurrent_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'recurrent_constraint': None, 'bias_constraint': None, 'dropout': 0.0, 'recurrent_dropout': 0.0, 'implementation': 2, 'reset_after': True}

{'name': 'gru_118', 'trainable': True, 'dtype': 'float32', 'return_sequences': True, 'return_state': False, 'go_backwards': False, 'stateful': False, 'unroll': False, 'time_major': False, 'units': 82, 'activation': 'tanh', 'recurrent_activation': 'sigmoid', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'recurrent_initializer': {'class_name': 'Orthogonal', 'config': {'gain': 1.0, 'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'recurrent_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'recurrent_constraint': None, 'bias_constraint': None, 'dropout': 0.0, 'recurrent_dropout': 0.0, 'implementation': 2, 'reset_after': True}

{'name': 'gru_119', 'trainable': True, 'dtype': 'float32', 'return_sequences': True, 'return_state': False, 'go_backwards': False, 'stateful': False, 'unroll': False, 'time_major': False, 'units': 123, 'activation': 'tanh', 'recurrent_activation': 'sigmoid', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'recurrent_initializer': {'class_name': 'Orthogonal', 'config': {'gain': 1.0, 'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'recurrent_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'recurrent_constraint': None, 'bias_constraint': None, 'dropout': 0.0, 'recurrent_dropout': 0.0, 'implementation': 2, 'reset_after': True}

Discriminador AC-CGAN GRU

Model: "model_48"

Layer (type)	Output Shape	Param #	Connected to
input_27 (InputLayer)	[(None, 3, 123)]	0	
input_28 (InputLayer)	[(None, 3, 123)]	0	
reshape_18 (Reshape)	(None, 3, 123, 1)	0	input_27[0][0]
reshape_19 (Reshape)	(None, 3, 123, 1)	0	input_28[0][0]
concatenate_9 (Concatenate)	(None, 3, 246, 1)	0	reshape_18[0][0] reshape_19[0][0]
conv2d_68 (Conv2D)	(None, 2, 123, 64)	640	concatenate_9[0][0]
conv2d_79 (Conv2D)	(None, 2, 123, 64)	640	concatenate_9[0][0]
conv2d_69 (Conv2D)	(None, 1, 62, 64)	36928	conv2d_68[0][0]
conv2d_80 (Conv2D)	(None, 1, 62, 64)	36928	conv2d_79[0][0]
leaky_re_lu_56 (LeakyReLU)	(None, 1, 62, 64)	0	conv2d_69[0][0]
conv2d_74 (Conv2D)	(None, 2, 123, 64)	640	concatenate_9[0][0]
leaky_re_lu_65 (LeakyReLU)	(None, 1, 62, 64)	0	conv2d_80[0][0]
conv2d_70 (Conv2D)	(None, 1, 31, 64)	36928	leaky_re_lu_56[0][0]
conv2d_75 (Conv2D)	(None, 1, 62, 64)	36928	conv2d_74[0][0]
conv2d_81 (Conv2D)	(None, 1, 31, 64)	36928	leaky_re_lu_65[0][0]
leaky_re_lu_57 (LeakyReLU)	(None, 1, 31, 64)	0	conv2d_70[0][0]
leaky_re_lu_61 (LeakyReLU)	(None, 1, 62, 64)	0	conv2d_75[0][0]
leaky_re_lu_66 (LeakyReLU)	(None, 1, 31, 64)	0	conv2d_81[0][0]
conv2d_71 (Conv2D)	(None, 1, 16, 64)	36928	leaky_re_lu_57[0][0]
conv2d_76 (Conv2D)	(None, 1, 31, 64)	36928	leaky_re_lu_61[0][0]
conv2d_82 (Conv2D)	(None, 1, 16, 64)	36928	leaky_re_lu_66[0][0]
leaky_re_lu_58 (LeakyReLU)	(None, 1, 16, 64)	0	conv2d_71[0][0]
leaky_re_lu_62 (LeakyReLU)	(None, 1, 31, 64)	0	conv2d_76[0][0]
leaky_re_lu_67 (LeakyReLU)	(None, 1, 16, 64)	0	conv2d_82[0][0]
conv2d_72 (Conv2D)	(None, 1, 8, 64)	36928	leaky_re_lu_58[0][0]
conv2d_77 (Conv2D)	(None, 1, 16, 64)	36928	leaky_re_lu_62[0][0]
conv2d_83 (Conv2D)	(None, 1, 8, 64)	36928	leaky_re_lu_67[0][0]
leaky_re_lu_59 (LeakyReLU)	(None, 1, 8, 64)	0	conv2d_72[0][0]
leaky_re_lu_63 (LeakyReLU)	(None, 1, 16, 64)	0	conv2d_77[0][0]
leaky_re_lu_68 (LeakyReLU)	(None, 1, 8, 64)	0	conv2d_83[0][0]
conv2d_73 (Conv2D)	(None, 1, 4, 64)	36928	leaky_re_lu_59[0][0]

conv2d_78 (Conv2D)	(None, 1, 8, 64)	36928	leaky_re_lu_63[0][0]
conv2d_84 (Conv2D)	(None, 1, 4, 64)	36928	leaky_re_lu_68[0][0]
leaky_re_lu_60 (LeakyReLU)	(None, 1, 4, 64)	0	conv2d_73[0][0]
leaky_re_lu_64 (LeakyReLU)	(None, 1, 8, 64)	0	conv2d_78[0][0]
leaky_re_lu_69 (LeakyReLU)	(None, 1, 4, 64)	0	conv2d_84[0][0]
flatten_12 (Flatten)	(None, 256)	0	leaky_re_lu_60[0][0]
flatten_13 (Flatten)	(None, 512)	0	leaky_re_lu_64[0][0]
flatten_14 (Flatten)	(None, 256)	0	leaky_re_lu_69[0][0]
gan (Dense)	(None, 1)	257	flatten_12[0][0]
kmeans (Dense)	(None, 288)	147744	flatten_13[0][0]
pck (Dense)	(None, 3)	771	flatten_14[0][0]

==========

Total params: 667,684
Trainable params: 0
Non-trainable params: 667,684

_____

Validity discriminators AC-CGAN GRU Model parameters:
{'batch_input_shape': (None, 3, 123), 'dtype': 'float32', 'sparse': False, 'ragged': False, 'name': 'input_197'}
{'batch_input_shape': (None, 3, 123), 'dtype': 'float32', 'sparse': False, 'ragged': False, 'name': 'input_198'}
{'name': 'reshape_121', 'trainable': True, 'dtype': 'float32', 'target_shape': (3, 123, 1)}
{'name': 'reshape_122', 'trainable': True, 'dtype': 'float32', 'target_shape': (3, 123, 1)}
{'name': 'concatenate_51', 'trainable': True, 'dtype': 'float32', 'axis': 2}
{'name': 'conv2d_425', 'trainable': True, 'dtype': 'float32', 'filters': 64, 'kernel_size': (3, 3), 'strides': (2, 2), 'padding': 'same', 'data_format': 'channels_last', 'dilation_rate': (1, 1), 'groups': 1, 'activation': 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None}
{'name': 'conv2d_426', 'trainable': True, 'dtype': 'float32', 'filters': 64, 'kernel_size': (3, 3), 'strides': (2, 2), 'padding': 'same', 'data_format': 'channels_last', 'dilation_rate': (1, 1), 'groups': 1, 'activation': 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None}
{'name': 'leaky_re_lu_544', 'trainable': True, 'dtype': 'float32', 'alpha': 0.30000001192092896}
{'name': 'conv2d_427', 'trainable': True, 'dtype': 'float32', 'filters': 64, 'kernel_size': (3, 3), 'strides': (2, 2), 'padding': 'same', 'data_format': 'channels_last', 'dilation_rate': (1, 1), 'groups': 1, 'activation': 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None}
{'name': 'leaky_re_lu_545', 'trainable': True, 'dtype': 'float32', 'alpha': 0.30000001192092896}
{'name': 'conv2d_428', 'trainable': True, 'dtype': 'float32', 'filters': 64, 'kernel_size': (3, 3), 'strides': (2, 2), 'padding': 'same', 'data_format': 'channels_last', 'dilation_rate': (1, 1), 'groups': 1, 'activation': 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None}
{'name': 'leaky_re_lu_546', 'trainable': True, 'dtype': 'float32', 'alpha': 0.30000001192092896}
{'name': 'conv2d_429', 'trainable': True, 'dtype': 'float32', 'filters': 64, 'kernel_size': (3, 3), 'strides': (2, 2), 'padding': 'same', 'data_format': 'channels_last', 'dilation_rate': (1, 1),

'groups': 1, 'activation': 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None}
{'name': 'leaky_re_lu_547', 'trainable': True, 'dtype': 'float32', 'alpha': 0.30000001192092896}
{'name': 'conv2d_430', 'trainable': True, 'dtype': 'float32', 'filters': 64, 'kernel_size': (3, 3), 'strides': (2, 2), 'padding': 'same', 'data_format': 'channels_last', 'dilation_rate': (1, 1), 'groups': 1, 'activation': 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None}
{'name': 'leaky_re_lu_548', 'trainable': True, 'dtype': 'float32', 'alpha': 0.30000001192092896}
{'name': 'flatten_106', 'trainable': True, 'dtype': 'float32', 'data_format': 'channels_last'}
{'name': 'gan', 'trainable': True, 'dtype': 'float32', 'units': 1, 'activation': 'sigmoid', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None}
K-means dicriminator AC-CGAN GRU Model parameters:
{'batch_input_shape': (None, 3, 123), 'dtype': 'float32', 'sparse': False, 'ragged': False, 'name': 'input_197'}
{'batch_input_shape': (None, 3, 123), 'dtype': 'float32', 'sparse': False, 'ragged': False, 'name': 'input_198'}
{'name': 'reshape_121', 'trainable': True, 'dtype': 'float32', 'target_shape': (3, 123, 1)}
{'name': 'reshape_122', 'trainable': True, 'dtype': 'float32', 'target_shape': (3, 123, 1)}
{'name': 'concatenate_51', 'trainable': True, 'dtype': 'float32', 'axis': 2}
{'name': 'conv2d_431', 'trainable': True, 'dtype': 'float32', 'filters': 64, 'kernel_size': (3, 3), 'strides': (2, 2), 'padding': 'same', 'data_format': 'channels_last', 'dilation_rate': (1, 1), 'groups': 1, 'activation': 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None}
{'name': 'conv2d_432', 'trainable': True, 'dtype': 'float32', 'filters': 64, 'kernel_size': (3, 3), 'strides': (2, 2), 'padding': 'same', 'data_format': 'channels_last', 'dilation_rate': (1, 1), 'groups': 1, 'activation': 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None}
{'name': 'leaky_re_lu_549', 'trainable': True, 'dtype': 'float32', 'alpha': 0.30000001192092896}
{'name': 'conv2d_433', 'trainable': True, 'dtype': 'float32', 'filters': 64, 'kernel_size': (3, 3), 'strides': (2, 2), 'padding': 'same', 'data_format': 'channels_last', 'dilation_rate': (1, 1), 'groups': 1, 'activation': 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None}
{'name': 'leaky_re_lu_550', 'trainable': True, 'dtype': 'float32', 'alpha': 0.30000001192092896}
{'name': 'conv2d_434', 'trainable': True, 'dtype': 'float32', 'filters': 64, 'kernel_size': (3, 3), 'strides': (2, 2), 'padding': 'same', 'data_format': 'channels_last', 'dilation_rate': (1, 1), 'groups': 1, 'activation': 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None}
{'name': 'leaky_re_lu_551', 'trainable': True, 'dtype': 'float32', 'alpha': 0.30000001192092896}
{'name': 'conv2d_435', 'trainable': True, 'dtype': 'float32', 'filters': 64, 'kernel_size': (3, 3), 'strides': (2, 2), 'padding': 'same', 'data_format': 'channels_last', 'dilation_rate': (1, 1), 'groups': 1, 'activation': 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None}
{'name': 'leaky_re_lu_552', 'trainable': True, 'dtype': 'float32', 'alpha': 0.30000001192092896}
{'name': 'flatten_107', 'trainable': True, 'dtype': 'float32', 'data_format': 'channels_last'}

{'name': 'kmeans', 'trainable': True, 'dtype': 'float32', 'units': 288, 'activation': 'softmax', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None}
3DCPK dicriminator AC-CGAN GRU Model parameters:
{'batch_input_shape': (None, 3, 123), 'dtype': 'float32', 'sparse': False, 'ragged': False, 'name': 'input_197'}
{'batch_input_shape': (None, 3, 123), 'dtype': 'float32', 'sparse': False, 'ragged': False, 'name': 'input_198'}
{'name': 'reshape_121', 'trainable': True, 'dtype': 'float32', 'target_shape': (3, 123, 1)}
{'name': 'reshape_122', 'trainable': True, 'dtype': 'float32', 'target_shape': (3, 123, 1)}
{'name': 'concatenate_51', 'trainable': True, 'dtype': 'float32', 'axis': 2}
{'name': 'conv2d_436', 'trainable': True, 'dtype': 'float32', 'filters': 64, 'kernel_size': (3, 3), 'strides': (2, 2), 'padding': 'same', 'data_format': 'channels_last', 'dilation_rate': (1, 1), 'groups': 1, 'activation': 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None}
{'name': 'conv2d_437', 'trainable': True, 'dtype': 'float32', 'filters': 64, 'kernel_size': (3, 3), 'strides': (2, 2), 'padding': 'same', 'data_format': 'channels_last', 'dilation_rate': (1, 1), 'groups': 1, 'activation': 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None}
{'name': 'leaky_re_lu_553', 'trainable': True, 'dtype': 'float32', 'alpha': 0.30000001192092896}
{'name': 'conv2d_438', 'trainable': True, 'dtype': 'float32', 'filters': 64, 'kernel_size': (3, 3), 'strides': (2, 2), 'padding': 'same', 'data_format': 'channels_last', 'dilation_rate': (1, 1), 'groups': 1, 'activation': 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None}
{'name': 'leaky_re_lu_554', 'trainable': True, 'dtype': 'float32', 'alpha': 0.30000001192092896}
{'name': 'conv2d_439', 'trainable': True, 'dtype': 'float32', 'filters': 64, 'kernel_size': (3, 3), 'strides': (2, 2), 'padding': 'same', 'data_format': 'channels_last', 'dilation_rate': (1, 1), 'groups': 1, 'activation': 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None}
{'name': 'leaky_re_lu_555', 'trainable': True, 'dtype': 'float32', 'alpha': 0.30000001192092896}
{'name': 'conv2d_440', 'trainable': True, 'dtype': 'float32', 'filters': 64, 'kernel_size': (3, 3), 'strides': (2, 2), 'padding': 'same', 'data_format': 'channels_last', 'dilation_rate': (1, 1), 'groups': 1, 'activation': 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None}
{'name': 'leaky_re_lu_556', 'trainable': True, 'dtype': 'float32', 'alpha': 0.30000001192092896}
{'name': 'conv2d_441', 'trainable': True, 'dtype': 'float32', 'filters': 64, 'kernel_size': (3, 3), 'strides': (2, 2), 'padding': 'same', 'data_format': 'channels_last', 'dilation_rate': (1, 1), 'groups': 1, 'activation': 'linear', 'use_bias': True, 'kernel_initializer': {'class_name': 'GlorotUniform', 'config': {'seed': None}}, 'bias_initializer': {'class_name': 'Zeros', 'config': {}}, 'kernel_regularizer': None, 'bias_regularizer': None, 'activity_regularizer': None, 'kernel_constraint': None, 'bias_constraint': None}
{'name': 'leaky_re_lu_557', 'trainable': True, 'dtype': 'float32', 'alpha': 0.30000001192092896}
{'name': 'flatten_108', 'trainable': True, 'dtype': 'float32', 'data_format': 'channels_last'}

Anexo 5

Resultados de las pruebas de ablación con la base de datos CMU completa

Modelo	3DPCK	Euclidiana	FID	MMD	K-S
Datos reales de referencia	0.00	0.00	0.0000	0.9490	0.0000
AC CGAN GRU completo	80.18	143.05	0.8284	0.0031	0.1041
Modelo sin K-means / PCK / Etiquetas	66.20	191.91	1.2583	0.0053	0.0967
Modelo sin etiquetas únicamente	12.33	689.64	36.5965	0.2461	0.6248
Modelo con K-Means únicamente	71.15	163.81	0.9309	0.0033	0.0873
Modelo con 3DPCK únicamente	69.32	199.91	1.1166	0.0046	0.0939
AC CGAN GRU Ensamble	**87.29**	**103.69**	**0.4703**	**0.0025**	**0.0708**
AC CGAN GRU 11	83.76	130.75	**0.5319**	0.0028	0.0798
AC CGAN GRU 12	**83.92**	**124.45**	0.5417	**0.0026**	**0.0675**
AC CGAN GRU 13	80.18	143.05	0.8284	0.0031	0.1041
AC CGAN GRU 14	79.51	145.01	0.7801	0.0032	0.0901
AC CGAN GRU 15	81.52	141.08	0.7225	0.0041	0.0820