APRENDIZAJE DE MOVIMIENTOS EN UN ROBOT HUMANOIDE POR MEDIO DE IMITACIÓN

ANDREA CATALINA REY RAMÍREZ
ALISON GISSELL RUIZ RUIZ

UNIVERSIDAD SANTO TOMÁS
INGENIERÍA ELECTRÓNICA
BOGOTÁ D.C
2020

APRENDIZAJE DE MOVIMIENTOS EN UN ROBOT HUMANOIDE POR MEDIO DE IMITACIÓN

ANDREA CATALINA REY RAMÍREZ
ALISON GISSELL RUIZ RUIZ

Proyecto de grado presentado como requisito para optar al título de
INGENIERO ELECTRÓNICO

UNIVERSIDAD SANTO TOMÁS
INGENIERIA ELECTRÓNICA
BOGOTÁ D.C
2020

UNIVERSIDAD SANTO TOMÁS

Ingeniería Electrónica

Este proyecto de grado fue aceptado para la carrera de Ingeniero(a) Electrónico(a) en Junio 2019

Director: Edgar Camilo Camacho Poveda
Magíster en Ingeniería Electrónica
Universidad Nacional
Bogotá, Colombia

Co-director: Carolina Higuera Arias
Magíster en Ingeniería Electrónica y de Computadores
Universidad de los Andes
Bogotá, Colombia

Jurados: Dario Alejandro Segura Torres
Magíster en Ciencias de las comunicaciones y la información
Universidad Distrital
Bogotá, Colombia

José Guillermo Guarnizo Marín
Doctor en Automática, Robótica e Informática Industrial
Universidad Politécnica de Valencia
Valencia, España

Sustentación proyecto de grado: ____ de Junio de 2020, BOGOTÁ D.C

Andrea Catalina Rey Ramírez _____

Alison Gissell Ruiz Ruiz _____

NOTA DE ACEPTACIÓN

El trabajo de grado titulado ***APRENDIZAJE DE MOVIMIENTOS EN UN ROBOT HUMANOIDE POR MEDIO DE IMITACIÓN****, realizado por las estudiantes* ***Andrea Catalina Rey Ramírez y Alison Gissell Ruiz Ruiz****, cumple con los requisitos exigidos por la* ***UNIVERSIDAD SANTO TOMÁS*** *para optar al título de* ***INGENIERO ELECTRÓNICO.***

Dario Alejandro Segura _____

José Guillermo Guarnizo _____

Edgar Camilo Camacho _____

Carolina Higuera Arias _____

DEDICATORIA

Dedicamos este trabajo de grado a nuestros padres, quienes hicieron todo lo posible para que culmináramos de manera exitosa nuestros estudios; por medio de su apoyo y motivación, fueron una base importante para nuestra formación integral y académica.

AGRADECIMIENTOS

Agradecemos a nuestro director de proyecto de grado, el Ingeniero Edgar Camilo Camacho y a nuestra codirectora la Ingeniera Carolina Higuera Arias, quienes nos colaboraron con el desarrollo del Dataset y fueron las personas que con su conocimiento y experiencia en la rama de Inteligencia Artificial, nos brindaron las herramientas necesarias para el desarrollo del trabajo, ayudándonos a entender y analizar el funcionamiento de una red neuronal y una SVM; agradecemos a nuestros padres quienes fueron el apoyo incondicional durante nuestra etapa académica; agradecemos a nuestros compañeros Yeison Suárez y José López quienes contribuyeron a la grabación del dataset y en temas relacionados con el uso del simulador; finalmente agradecemos a todos los docentes que estuvieron involucrados en nuestro aprendizaje, pues nos guiaron en el camino del conocimiento.

Índice general

Resumen **14**

Introducción **16**

1. Planteamiento del problema **18**

2. Justificación **20**

3. Objetivos **22**

3.1. Objetivo General 22

3.2. Objetivos Específicos 22

4. Antecedentes **24**

4.1. Conceptos básicos de aprendizaje de máquina 24

4.2. Algoritmos de aprendizaje de máquina aplicados a robótica 25

4.3. Robot humanoide 29

5. Marco Teórico **30**

5.1. Aprendizaje Automático 30

5.2. Aprendizaje Supervisado 31

5.3. Regresión lineal 32

5.3.1. Optimización y Descenso de Gradiente 33

5.4. Regresión logística 34

5.4.1. Función de costo 36

5.5. Redes Neuronales 37

5.5.1. Redes Neuronales Convolucionales 39
5.5.2. Redes Neuronales Recurrentes 41
5.5.3. *Long Short-Term Memory (LSTM)* 44
5.6. *Support Vector Machine (SVM)* 46
5.6.1. Función de Costo 47
5.6.2. Kernel 48
5.6.3. Límite de decisión o superficie de decisión 49
5.7. Métricas en Aprendizaje de Máquina 49
5.7.1. Matriz de confusión 49
5.7.2. Recall - Sensibilidad 50
5.7.3. Precisión 50
5.7.4. Accuracy - Exactitud 51
5.7.5. Puntuación F1 51
5.8. Estimación de la postura humana (*Pose estimation*) 51
5.9. Interpolación 53

6. Desarrollo Metodológico **54**

7. Construcción Robot Humanoide (Poppy Torso) **56**
7.1. Impresión de piezas 56
7.2. Ensamblaje de motores 56

8. Creación de *Dataset* **59**
8.1. *Dataset* de movimientos humanos 59
8.1.1. Elección de movimientos a implementar 59
8.1.2. Software para la adquisición de datos 60
8.1.3. Configuración del entorno de grabación 61
8.1.4. Identificación y extracción de información 61
8.1.5. Normalización y organización del *dataset* 63
8.2. *Dataset* de movimientos robóticos 63

9. Clasificación de movimientos **65**
9.1. Red Neuronal Recurrente - LSTM Many to one 65
9.1.1. Evaluación cuantitativa 67

9.2. Máquina de vectores de soporte (SVM) 69
9.2.1. SVM 70
9.2.2. Evaluación cuantitativa 73

10. Generación de trayectorias **75**
10.1. Red Neuronal Recurrente - LSTM One to Many 75
10.1.1. Evaluación cuantitativa 78

11. Integración del sistema **82**

12. Análisis de resultados **90**
12.1. Algoritmos de clasificación 90
12.1.1. Comparación del desempeño de los algoritmos de clasificación 91
12.2. Algoritmo de generación de trayectorias 93

Conclusiones **95**

Trabajos futuros **98**

Productos del proyecto **99**

Glosario **101**

A. Construcción Poppy **103**

B. Entrenamientos Many to one **104**

C. Entrenamientos SVM **107**

D. Entrenamientos One to many **110**

E. Descripción Software **113**

Índice de cuadros

5.1. Matriz de confusión 50

7.1. Tabla Aspectos técnicos - Poppy Torso 58

9.1. Tabla Arquitectura RNN many to one 66
9.2. Tabla Parámetros Barridos 66
9.3. Tabla Parámetros Barridos 70
9.4. Tabla Arquitectura RNN many to one- dataset 71
9.5. Tabla Parámetros Barridos 71

10.1. Tabla de ángulo por motor 76
10.2. Tabla Parámetros Barridos 76

12.1. Tabla Comparación Algoritmos 92
12.2. Especificaciones del computador 93

A.1. Tabla Impresión partes Poppy 103

B.1. Tabla Entrenamientos Many to One-Dataset 2 dimensiones 106

C.1. Tabla Entrenamientos SVM Dataset 2 dimensiones 109

D.1. Tabla Entrenamientos One to Many 112

E.1. Autores del registro software 113

Índice de figuras

4.1. PoppyTorso [15] 29

5.1. Aprendizaje Supervisado[26] 31
5.2. Regresión lineal 33
5.3. Regresión Logística 35
5.4. Función Sigmoidea o función logística 36
5.5. Modelo biológico neurona [27] 37
5.6. Arquitectura de una red neuronal 39
5.7. Ejemplo de convolución 40
5.8. Arquitectura de una red convolucional[28] 41
5.9. Arquitectura de una red neuronal recurrente 41
5.10. Tipos de Redes Neuronales Recurrentes [30] 43
5.11. Estructura de una LSTM [30] 44
5.12. Puertas LSTM [30] 46
5.13. Margen de separación SVM[35] 47
5.14. Límite de decisión 49
5.15. Salida Pose Estimation 52
5.16. Arquitectura Pose Estimation [34] 52
5.17. Interpolación 53

7.1. Poppy Torso Construido 57
7.2. Poppy Articulaciones 58

8.1. GUI empleada para creación de dataset 60
8.2. Puntos extraídos de la máquina de pose 62

8.3. Puntos extraídos de un movimiento empleando *Pose estimation* 62
8.4. Poppy Torso simulado en RVIZ 64

9.1. Precisión y pérdida Many to one 67
9.2. Matriz de confusión Many to one 68
9.3. Posición Manos RNN 69
9.4. SVM 69
9.5. Curva aprendizaje con validación cruzada 72
9.6. Matriz de confusión SVM 73
9.7. Posición Manos SVM 74

10.1. One to many pérdida por movimiento 77
10.2. Trayectoria de motores para Sides-Center 78
10.3. Error en el tiempo para movimiento Sides-Center 79
10.4. Error de posición final de los motores 79
10.5. Posición de los motores para el movimiento Down 80
10.6. Movimiento ejecutado por Poppy 80

11.1. Interfaz tiempo Real 82
11.2. Interfaz tiempo Real Partes 84
11.3. Servicios de ROS 84
11.4. Servicios ROS no encontrados 85
11.5. Interfaz ejecutando movimiento 86
11.6. Interfaz sin algoritmo seleccionado 86
11.7. Interfaz sin Persona 87
11.8. Aplicación y Simulación(Down-Center) 88
11.9. Aplicación y Simulación(Down-Up) 89

Resumen

El proyecto de grado titulado *Aprendizaje de movimientos en un robot humanoide por medio de imitación* implementa algoritmos de aprendizaje supervisado, proporcionando a un robot la capacidad de identificar movimientos realizado en las extremidades superiores por una persona y replicarlos. Se evalúa, además, el desempeño cuantitativo de las arquitecturas implementadas.

Para desarrollar el aprendizaje de movimientos se selecciona una plataforma robótica llamada Poppy Torso, correspondiente a un proyecto de código abierto. Se construye este robot por medio de impresión 3D de las piezas y su posterior ensamblaje. Dicho desarrollo contiene además la descripción virtual del robot, lo cual permite realizar simulación en herramientas como *Rviz*.

El desarrollo del proyecto se divide en dos etapas: la primera de ellas consiste en lograr la clasificación de los movimientos humanos. Por esta razón, se crea un *dataset* de videos RGB-D (RGB más profundidad), que contiene movimientos de las extremidades superiores, realizados por un grupo de personas con características diferentes. Empleando un modelo extractor de poses, se extrae la ubicación espacial de las articulaciones, que sirve como entrada a una red neuronal recurrente (*RNN-many to one*) y una SVM *(Support Vector Machine)*. Se evalúa el desempeño de estos algoritmos por medio de matrices de confusión.

La segunda etapa consiste en un problema de regresión que se basa en replicar la ejecución del movimiento en el robot. Debido a esto, se crea un *dataset* que contiene las trayectorias de ángulos que toma cada motor,

para que el robot desplace sus articulaciones de un punto aleatorio a uno definido. Para el aprendizaje supervisado se implementa una red neuronal recurrente (*RNN-one to many*), capaz de predecir las trayectorias de cada una de las articulaciones del robot para llegar a un punto objetivo.

Los algoritmos de clasificación se evalúan por medio de matrices de confusión, donde se obtienen precisiones de 97 % en el caso de la SVM y 92 % para la RNN. Por otro lado, el algoritmo de regresión se evalúa con error de posición final, donde se obtienen errores entre $9{,}32 \times 10^{-8}$ y $4{,}21 \times 10^{-3}$ radianes respecto a la posición objetivo. Por el valor de estas métricas y el comportamiento de los algoritmos al predecir, se puede concluir que el aprendizaje en los algoritmos de clasificación y regresión se desarrolla adecuadamente, cumpliendo con los objetivos propuestos.

Introducción

El programa de Ingeniería Electrónica de la Universidad Santo Tomás, se propone formar ingenieros electrónicos integrales con sólida fundamentación científica, tecnológica y humanística, que responda a las necesidades de la sociedad en un mundo globalizado, con innovación y creatividad bajo los principios de la ética.

Este proyecto de grado pretende aportar en áreas de investigación sobre herramientas tecnológicas actuales como la Robótica y la Inteligencia Artificial, a partir del uso de algoritmos de aprendizaje supervisado como redes neuronales recurrentes (RNN) y máquinas de vectores de soporte (SVM), para lograr el aprendizaje de movimientos en las extremidades superiores de un robot humanoide, por medio de ejemplos ejecutados por personas.

Para lograr este objetivo, es necesario realizar la detección del movimiento que ejecuta la persona, esto corresponde a un problema de clasificación, donde se busca agrupar los movimientos a través de etiquetas, teniendo en cuenta sus características. Esta primera etapa se desarrolla haciendo uso de algoritmos supervisados, específicamente redes neuronales y máquinas de vectores de soporte, con el fin de comparar el comportamiento de dichos métodos. Por este motivo, se crea un *dataset* de videos RGB-D con movimientos humanos, que sirven como entrada a los algoritmos LSTM *Many to One* y SVM, teniendo como tarea la clasificación de los movimientos ejecutados por la persona. La precisión de estos algoritmos de clasificación se mide cuantitativamente por medio de matrices de confusión.

Posteriormente, se debe realizar la ejecución en el robot del movimiento detectado. Esta tarea corresponde a un problema de regresión, ya que busca generar la secuencia que debe realizar cada uno de los motores. Esta predicción surge a partir de un entrenamiento previo, donde el algoritmo encuentra una tendencia de datos. Con este objetivo, se crea un segundo *dataset*, con trayectorias que genera el robot en sus articulaciones al moverse de un punto a otro, para servir como entrada al algoritmo LSTM *One to Many*. Dicho modelo tiene como objetivo generar la secuencia de ángulos que deben seguir las articulaciones, desde cualquier posición inicial, para que el robot ejecute el movimiento.

De esta forma, se logra por medio de algoritmos de aprendizaje de máquina que el robot aprenda a partir de demostraciones, identificando de forma autónoma los movimientos de extremidades superiores realizados por el humano y generando trayectorias que reproduzcan el movimiento, teniendo en cuenta que los grados de libertad que tiene la plataforma robótica empleada no van a coincidir con los de una persona.

Además, se desarrolla una interfaz gráfica que le permite al usuario observar la clasificación de los algoritmos en tiempo real a través de una cámara. Permitiendo que al realizar alguno de los movimientos para los cuales fueron entrenados, se presente la detección y se ejecute dicho movimiento por medio de la herramienta de visualización *Rviz*, presentando la trayectoria que genera el robot para llegar a la posición.

Los resultados de esta investigación pueden ser extrapolados para su futura aplicación al apoyo en terapias físicas, evitando el desplazamiento de pacientes que requieren rehabilitación de la capacidad motora, producto de discapacidad o lesiones musculares que dificulten su traslado hacia un centro médico. El robot puede aprender la rutina de movimientos específica que sugiera el fisioterapeuta y ejecutarla desde el lugar de residencia del paciente.

Capítulo 1

Planteamiento del problema

Según el Ministerio de Salud, en Colombia ha incrementado la calidad de vida y ha disminuido la mortalidad, ocasionando que las personas tengan un tiempo de vida más alto. Por esta razón, existe un aumento en el envejecimiento demográfico, donde la tasa de población mayor a 60 años tiene un crecimiento superior al de la población total. Al aumentar su edad, las personas se ven limitadas en sus movimientos, creando dependencia de otra persona, ocasionándoles problemas de baja autoestima y depresión [3].

En algunos casos se presenta para esta población la necesidad de asistir a terapias físicas para adquirir rehabilitación de lesiones musculares, ya sean éstas producto de enfermedades degenerativas o eventos traumáticos (por ejemplo, implantación de una prótesis en una extremidad superior)[4]. El aumento de tiempo de vida de la población provoca un incremento en la cantidad de pacientes que requieren fisioterapia, afectando de forma negativa la atención médica, provocando un tiempo de espera bastante alto para adquirir la restauración de la capacidad motora mediante la terapia física.

Un robot humanoide podría ser quien lleve a cabo la fisioterapia, ya sea teleoperándolo o programando explícitamente el movimiento. Sin embargo, el profesional de la salud no necesariamente está capacitado para operar este tipo de dispositivos. Por lo tanto, podría ser más factible que el fisiatra

instruya al robot por medio de demostraciones de los movimientos que deben realizar las personas de la tercera edad durante la terapia. Para lograr esto, primero se necesita resolver el problema de ingeniería, que consiste en lograr que el robot humanoide aprenda a imitar movimientos por medio de las demostraciones de los mismos, los cuales son ejecutados por personas.

Se debe identificar inicialmente el movimiento que ejecuta la persona. Para esto surge la problemática de definir la metodología para adquirir los datos, cómo se manejará la extracción de información útil, qué arquitecturas de aprendizaje supervisado pueden ser empleadas para lograr este fin y de qué manera se puede evaluar cuantitativamente el desempeño de los algoritmos planteados, con el fin de conocer cuál estructura favorece más a las necesidades del proyecto.

Para realizar la ejecución del movimiento en el robot, se hace necesario que sea capaz de llegar a un punto final sin importar la posición inicial, evitando de esta forma predefinir infinitos movimientos, logrando que generalice y cree trayectorias que alcancen la posición deseada. Por este motivo se hace tácita la búsqueda de una arquitectura de aprendizaje supervisado capaz de generar las trayectorias que deben seguir los motores para ejecutar el movimiento sin importar la posición de partida del robot.

Capítulo 2

Justificación

A través de los avances tecnológicos y robóticos, y teniendo en cuenta la hoja de ruta de la robótica europea y americana, se percibe que uno de los ejes de acción de la robótica es brindar asistencia a la tercera edad [5]. La relación entre la cantidad de fisiatras con respecto al número de personas que requieren terapias físicas se convierte en un impedimento para que todos reciban correcta atención, en especial para los adultos mayores, que necesitan ser atendidos de forma efectiva (basado en la ley 1251 de 2008, artículo 6 [6]), pues el tiempo de recuperación del paciente depende de la cantidad de espacio práctico dedicado a la terapia física. Dicho esto, se considera pertinente realizar una investigación temprana, que permita a futuro generar un prototipo de un robot humanoide equipado con inteligencia artificial, capaz de aprender por medio de imitación.

En los estudios de lactantes y animales, la capacidad de imitar generalmente se concluye a partir de la mayor tendencia del sujeto a ejecutar un comportamiento previamente demostrado [8]. Para que un robot humanoide alcance la naturalidad de la interacción, que es uno de los principios de la robótica social, un buen enfoque es dotar al robot de formas de aprendizaje similares a las de las personas [7], como podría ser el aprendizaje por observación e imitación. Actualmente, en el área del aprendizaje de máquina, se cuenta con una serie de algoritmos y

estrategias que permiten a los sistemas aprender a través de datos que surgen de la observación de su entorno.

El robot empleado es Poppy Torso, una plataforma de código abierto que incluye esquemas para su construcción, y descripción virtual para su simulación. Para la implementación del robot, se realiza la impresión 3D de las piezas, y posteriormente se ensambla con los motores adecuados, logrando el movimiento para cada articulación. La construcción del robot se realiza bajo la asesoría del grupo de investigación GED, en el marco del proyecto FODEIN titulado *"Aprendizaje supervisado para imitación de movimientos de un robot humanoide en el marco de la alianza SINFONIA"* planteado para el año 2019. Mediante las herramientas desarrolladas, como los son el robot y el *dataset*, estudiantes y docentes de la facultad pueden desarrollar proyectos enfocados a robótica social, enriqueciendo las herramientas del laboratorio de Robótica, de la Facultad de Ingeniería Electrónica, y de la Universidad Santo Tomás.

El robot adquirirá la capacidad de imitar movimientos de las extremidades superiores al obtener demostraciones por medio de un *dataset* conformado por vídeos RGB-D, de movimientos de las extremidades superiores. La implementación de algoritmos de aprendizaje supervisado, los cuales le permitan al robot humanoide una obtención clara de los movimientos realizados por el usuario y una ejecución del mismo, la ejecución del movimiento se puede realizar por medio de simulación o en la plataforma física real.

Por otro lado, los algoritmos implementados sirven de insumo para futuras investigaciones, que tengan relación con problemas que se puedan solucionar por medio del uso de aprendizaje supervisado, ya que las estructuras de clasificación y regresión implementadas pueden ser utilizadas para el desarrollo de diferentes proyectos. Principalmente, aporta a la facultad de Ingeniería Electrónica en el tema de investigación sobre la imitación de movimientos humanos en un robot.

Capítulo 3

Objetivos

3.1. Objetivo General

Seleccionar el algoritmo de aprendizaje supervisado que presente mejor desempeño cualitativo y cuantitativo a partir de la implementación de redes neuronales y SVMs, para que un robot humanoide imite movimientos básicos de las extremidades superiores de los seres humanos.

3.2. Objetivos Específicos

- Realizar la impresión en 3D y el ensamblaje del robot Poppy torso.
- Construir una base de datos de videos RGB-D que contenga movimientos de extremidades superiores, los cuales puedan ser utilizados como experto para el robot.
- Identificar las características del *dataset* necesarias para que el robot pueda replicar el movimiento y obtenerlas por medio de algoritmos de visión artificial.
- Implementar algoritmos de aprendizaje supervisado, como redes neuronales y SVMs, para la imitación de movimientos.
- Analizar los resultados de los algoritmos de aprendizaje de forma

cuantitativa entregados por la simulación del robot Poppy Torso, por medio de una matriz de confusión.

- Realizar la comparación del desempeño de los algoritmos a partir de los resultados cuantitativos presentados.

Capítulo 4

Antecedentes

A continuación se presentan investigaciones previamente realizadas en el ámbito de ingeniería, que tienen relación los objetivos propuestos en este proyecto de grado. Para contextualizar, inicialmente se definen algunos conceptos básicos de aprendizaje de máquina que se emplearán a lo largo de este capítulo. Luego, se muestran diferentes puntos de vista desde los cuales se han abordado soluciones a movimientos en robots. Finalmente se describe la plataforma robótica empleada en el proyecto de grado.

4.1. Conceptos básicos de aprendizaje de máquina

Una red neuronal consta de muchos procesadores interconectados entre ellos, llamados neuronas, donde cada neurona produce una secuencia de activaciones de valor real. Como se explica en [1], las neuronas de entrada se activan a través de sensores que perciben el entorno, otras neuronas se activan por medio de conexiones ponderadas de neuronas previamente activas. El aprendizaje consiste en la asignación de los pesos que multiplican a las entradas, lo cual permite que la red neuronal presente el comportamiento deseado.

Por otro lado, una máquina de soporte de vectores SVM está inicialmente diseñada para clasificación binaria por medio de un criterio

de margen maximizado. Este tipo de algoritmo contiene una herramienta, llamada kernel, que logra realizar clasificación de una forma no lineal. Existen varios tipos de kernel que se pueden implementar según el tipo de problema a abordar, ya que tienen la capacidad de aumentar la dimensión, ayudando a buscar que los datos sean linealmente separables, como el kernel polinomial, RBF y Sigmoide. El algoritmo SVM construye un hiperplano de separación con el margen máximo, donde se emplean los vectores de soporte para la clasificación y se pueden eliminar muchas muestras mayoritarias lejos del límite de decisión [2], siendo capaz de solucionar problemas multiclases.

4.2. Algoritmos de aprendizaje de máquina aplicados a robótica

En torno a la historia, existe bastante documentación sobre temas relacionados con inteligencia en los robots, donde uno de los primeros robots humanoides que desarrollaron inteligencia se muestra en [16], en el cual, gracias a Honda y su creación de un robot humanoide en el año 1996 llamado P2, se logró formar en un robot humanoide la capacidad de subir escaleras, manipular objetos simples y caminar en una ruta planeada de forma autónoma si el ambiente es conocido o con la posibilidad de ser teleoperado inalámbricamente si el ambiente es desconocido. El humanoide posee cuatro videocámaras para el proceso de realizar la visión del *target* o de la posición estimada y así aprender sobre su ubicación en el espacio y ejecutar movimientos para llegar a la ubicación deseada. Sin embargo, se consideró poco eficiente debido a la necesidad, bajo algunas condiciones, de realizar teleoperación.

En [8] Schaal plantea las similitudes y diferencias entre las formas de aprendizaje de imitación biológicas y computacionales. Se explica el funcionamiento de la imitación en el cerebro humano, donde existe una entrada visual y una salida motora manejada en diferentes áreas del cerebro y luego se realiza una aproximación de dicho comportamiento a

través de un modelo computacional que se puede implementar por medio de redes neuronales o lógica difusa. El estado de *target* es alguno de los parámetros naturales que posee un movimiento primitivo y las transiciones entre el estado actual al estado de *target* son la firma espacio-temporal a los cuales se recurren como candidatos, para lograr la correspondencia entre diferentes modos de información sensorial y así lograr la ejecución de los movimientos.

En [9] se presenta una propuesta donde se emplea un algoritmo de regularización adaptativa de políticas, siendo capaz de converger en espacios dinámicos que afecten el movimiento. En [10] se aplica una red neuronal recurrente basada en un framework de codificación predictiva e interferencia activa, que logra detectar el movimiento de la persona y actualiza los datos de los motores para lograr el seguimiento de las articulaciones. En [11] se emplea un método de aprendizaje de movimientos que usa una combinación de las fortalezas de una red neuronal recurrente LSTM (Long short-term memory) y sistemas dinámicos, permitiendo el ajuste de las trayectorias ante perturbaciones externas que puedan afectar el movimiento.

En [12] se realiza un framework que emplea similitud de imagen futura para predecir el movimiento que realizará la persona un momento después, se utiliza aprendizaje no supervisado, por medio de un autoencoder convolucional condicionado a acciones para replicar los movimientos en el robot. Por otro lado, en [13] se emplea una red neuronal perceptrón multicapa (conocida como MLP, por sus siglas en inglés) para evaluar la estabilidad en la ejecución del movimiento en el robot; se emplea un kinect para lectura de las articulaciones humanas y se transforma esta información por medio de la ecuación kinemática inversa a ángulos del robot.

El aprendizaje a partir de demostraciones (*Learning from Demonstration - LfD*) se ha catalogado como un enfoque prometedor para obtener buenas políticas, partiendo del conocimiento de expertos en la tarea de aprendizaje [17]. Si el experto es un ser humano, se garantiza

que el robot no replica errores del experto al aprender de las demostraciones, trasladando el problema a un cambio de dominio del espacio de estado de la persona al del robot [18].

En [19] Hussein propone una solución para el problema de aprendizaje por demostración *(LfD)* únicamente con aprendizaje supervisado. Los autores proponen construir un *dataset* a través de las demostraciones del experto para entrenar una red neuronal convolucional profunda que capture las características que permiten imitar el comportamiento deseado. La política aprendida es refinada por medio de aprendizaje activo para generalizar situaciones que no se hayan presentado en las demostraciones del experto. Cuando el agente requiere al experto en situaciones donde tiene baja confianza, este le provee acciones óptimas que le permiten continuar explorando nuevos estados.

En [20] se presenta otro estudio de aprendizaje de robots por medio de demostración, donde se define una técnica para alcanzar el aprendizaje por medio de una función de mapeo que es usada para llevar al robot del estado de observación al estado de actuar. También emplean un modelo dinámico del sistema que se usa para interactuar con el entorno. Esta técnica de aprendizaje se limita a aprender las demostraciones del maestro, no hay posibilidad de que el robot cree nuevas secuencias de movimiento así estas secuencias mejoren el rendimiento para alcanzar el *target*. El *target* de este tipo de algoritmo es reproducir una secuencia de movimientos que inicialmente es desconocido, para lo cual debe generalizar sobre el conjunto de ejemplos disponibles.

Kaelbling en [21] establece diferentes métodos para abordar el problema del aprendizaje supervisado, como lo son métodos de redes neuronales, lógica difusa, CMAC (*Cerebellar Model Articulation Controller*) y métodos locales basados en la memoria. Sin embargo, los problemas de replicar movimientos son afrontados frecuentemente a partir de redes neuronales con el enfoque de reproducir secuencias dinámicas, como ocurre en el estudio desarrollado por Maass en [22].

Zhang en [23] presenta un sistema de aprendizaje profundo por

imitación, para la manipulación de tareas complejas a partir de teleoperación por realidad virtual. En esencia es un algoritmo de clonado basado en una red neuronal profunda. Levine en [24] propone utilizar una red neuronal convolucional (CNN por sus siglas en inglés) para aprender la política de control, mapeando directamente las acciones desde la imagen raw. La CNN fue entrenada usando aprendizaje por refuerzo, a través del método de simple trayectoria céntrica.

En [25] se realiza aprendizaje por medio de imitación empleando vídeos de humanos realizando tareas con varios objetos, buscando extraer los aspectos de la actividad del ser humano que son los más relevantes para elegir acciones. El algoritmo presenta la capacidad de aprender de una sola demostración utilizando el conocimiento previo creado a partir de muchas demostraciones de otras tareas. El robot debe inferir la meta del movimiento y determinar la acción adecuada, para lo cual emplean el método MAML (*Model-Agnostic Meta-Learning*). El proyecto de grado difiere de este estudio en que las demostraciones por medio de vídeo se realizan en formato RGB-D para evaluar la profundidad de los movimientos y en que el *target* es que el robot aprenda movimientos, no debe inferir que parte del movimiento es importante y elegir acciones que cumplan con la meta que se obtuvo en la demostración, sino ser capaz de aprender movimientos y replicarlos cuando se requiera, lo cual quiere decir que se realizará aprendizaje supervisado y no por refuerzo, por ello no se empleará el método MAML sino otros acordes al *target* como lo son redes neuronales y SVMs.

En [30] se realiza una corporación del comportamiento presentado por una red neuronal convencional y una red neuronal recurrente LSTM, para un problema de reconocimiento de actividad humana, aquí se observa que resultados muestran la LSTM supera a las redes convolucionales en el conjunto prueba con un rango de diferencia entre 4 % y 9 %, este resultado nos demuestra que la arquitectura LSTM, sirve para solucionar problemas secuenciales y presenta similitudes con el proyecto de grado, puesto que se empleara para detección de movimientos humanos.

4.3. Robot humanoide

Se define como robot humanoide, "Robot que adquiere por su forma o por sus propiedades la capacidad para interactuar con humanos de forma que se asemeje en su comportamiento o en su aspecto al de un humano" [14].

Bajo la visión de que los robots son herramientas poderosas para aprender y ser creativos, los autores de origen Francés de la plataforma *Poppy Project*, desarrollaron autómatas interactivos impresos en 3D. Este desarrollo se encuentra disponible para fines educativos, artísticos y científicos. La plataforma promueve el código abierto al compartir hardware por medio de los planos en 3D de las partes que componen los robots y software por medio de la descripción virtual del robot, que permite realizar simulación en herramientas de visualización como *Rviz*. Existen tres tipos de robots Poppy: *Poppy humanoid*, *Poppy Torso* y *Ergo Jr* [15]. Para el proyecto de grado se implementa el robot *Poppy Torso*, que se observa en la figura 4.1

Figura 4.1. PoppyTorso [15]

Capítulo 5

Marco Teórico

A continuación se presentan los soportes contextuales y de carácter teórico, que se emplean en el desarrollo de este proyecto. Inicialmente, se describe en qué consiste el aprendizaje automático y supervisado. Luego, se describen los tipos de aprendizaje supervisado estudiados. Finalmente, se abordan herramientas como *Pose Estimation* e interpolación.

5.1. Aprendizaje Automático

El aprendizaje automático, es un subcampo de la ciencia de la computación que busca resolver problemas a partir de ejemplos. Ya que a diferencia de los programas informáticos típicos, posee técnicas que permiten iterativamente crear un aprendizaje por medio de datos, donde el algoritmo extrae sus características y crea un modelo que corresponde a los valores de algunos parámetros, que permiten predecir un cierto resultado. Los tres principales tipos de algoritmos de aprendizaje automático son: *aprendizaje supervisado*, *aprendizaje no supervisado* y *aprendizaje por refuerzo*. En este proyecto se plantea una solución por medio de aprendizaje supervisado.

5.2. Aprendizaje Supervisado

El aprendizaje supervisado crea un modelo de entrenamiento sobre datos históricos, al conocer el conjunto de características de la variable de entrada *(x)* y la variable de salida *(y)*. Este tipo de aprendizaje es empleado para la solución de dos tipos de problemas. Cuando se desea conocer un valor continuo, se llama problema de *regresión*, por otro lado, cuando se desean predecir valores discretos, previamente etiquetados como valores categóricos, se denomina problema de *clasificación*.

Los parámetros que intervienen en un aprendizaje supervisado son:

- n: Número de características de la muestra.
- m: Número de muestras de entrenamiento.
- (x, y): Muestra de entrenamiento.

El algoritmo de aprendizaje toma los datos del conjunto de entrenamiento y construye una función $h(x)$, llamada hipótesis, que tiene como fin estimar el valor de y. En la figura 5.1 se observa la estructura de esta función.

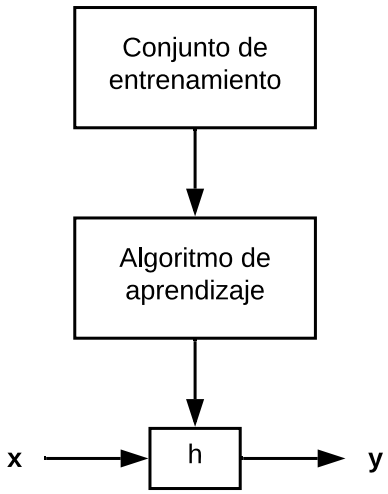


Figura 5.1. Aprendizaje Supervisado[26]

5.3. Regresión lineal

Algoritmo de aprendizaje supervisado, cuyo objetivo es establecer un modelo para la relación entre variables dependientes y variables independientes, obteniendo una variable cuantitativa. La regresión lineal es clasificada de acuerdo a la cantidad de variables independientes que tenga el problema, de acuerdo a esto los tipos son:

- *Regresión lineal simple:* Regresión compuesta por una variable independiente y una variable dependiente.
- *Regresión lineal múltiple:* Regresión compuesta por varias variables independientes y una variable dependiente.

La representación de la regresión lineal múltiple se encuentra en la ecuación 5.1.

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_n x_n \tag{5.1}$$

En la figura 5.2 se observa un ejemplo de una regresión lineal simple, donde los puntos negros representan el conjunto de características x con relación a la salida y, la linea roja representa un modelo entrenado y se conoce como línea de regresión, es representada con una ecuación lineal $h_\theta(x) = \theta_0 + \theta_1 x$, aquí se puede observar que θ_0 no multiplica ningún parámetro, esto sucede por que θ_0 es el sesgo del algoritmo, el sesgo es la representación matemática de errores al momento de la toma de los datos y $h_\theta(x)$, es la hipótesis del algoritmo, al tomar un valor de x, con respecto a los parámetros .

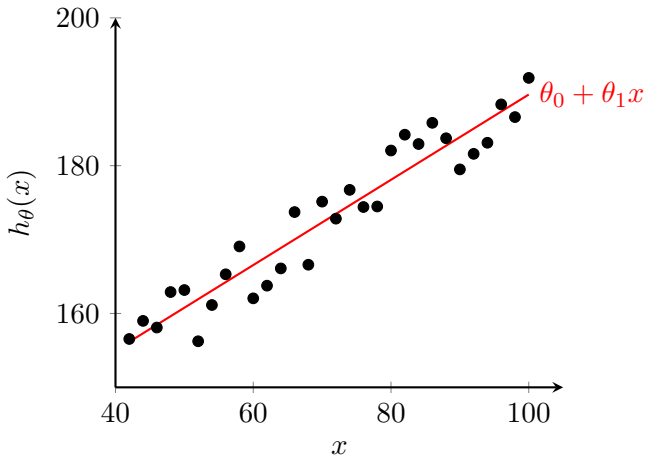


Figura 5.2. Regresión lineal

5.3.1. Optimización y Descenso de Gradiente

La función de costo representada con la ecuación 5.2, es aquella que determina el error entre el valor estimado y el valor real, con el fin de optimizar los parámetros del algoritmo de aprendizaje, la función de costo se halla sobre todas las muestras como se observa en la ecuación 5.2.

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h\theta(x^{(i)}) - y^{(i)})^2 \tag{5.2}$$

Se busca minimizar $J(\theta_0, \theta_1)$, por medio de un algoritmo de optimización como el *gradiente descendente* presentado en la ecuación 5.3.

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial j} J(\theta_0, \theta_1) \tag{5.3}$$

Se debe tener en cuenta que:

- $:=$ representa asignación, es decir sobre escribir θ_j en cada iteración hasta que converja.
- α es el *Coeficiente de aprendizaje*, el cual controla el tamaño del paso de gradiente el cual siempre es positivo.

- Se deben actualizar simultáneamente θ_0 y θ_1.
- Cuando ya se encontró el mínimo, ya sea local o global, los parámetros no cambian, por lo que al realizar otro paso del algoritmo, no se verán variaciones en el resultado.
- A medida que se acerca a un mínimo, el descenso de gradiente dará pasos mas pequeños, lo que hace mas pequeña la derivada, es por esto que no sera necesario disminuir α con el paso del tiempo.

El descenso de gradiente se puede representar de acuerdo a los posibles valores que tome j, los cuales son *0* y *1*, en las ecuaciones 5.4 y 5.5 se puede representar el descenso de gradiente para una regresión lineal simple.

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i) \tag{5.4}$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} ((h_\theta(x_i) - y_i)x_i) \tag{5.5}$$

5.4. Regresión logística

Tipo de regresión, utilizado para predecir el valor de una variable la cual puede adoptar diferentes categorías, esta predicción se realiza por medio de las variables independientes o predictorias; otra de sus aplicaciones es el modelado de la probabilidad de un evento el cuál este ocurriendo como consecuencia de otros factores, estas probabilidades se modelan a través de la función logística 5.6. La representación gráfica de la regresión logística se puede observar en la figura 5.3.

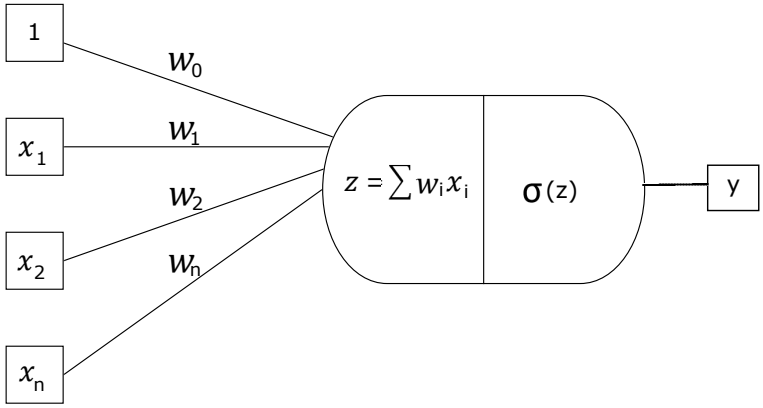


Figura 5.3. Regresión Logística

$$g(z) = \frac{1}{1 + e^{-z}} \tag{5.6}$$

Los valores de x_i corresponden las variables independientes del problema, los valores de w_i representan la influencia de cada una de las entradas para la predicción del valor deseado y, esto se puede ver representado matemáticamente en la ecuación 5.7.

$$y = \sigma(\sum w_0 x_0 + w_1 x_1 + ... + w_n x_n) \tag{5.7}$$

Es importante tener en cuenta que $y \in \{0, 1\}$, por lo que es importante que la hipótesis $h_\theta(x)$, pueda satisfacer $0 \leq h_\theta(x) \leq 1$, lo cual se realiza al introducir $\theta^\tau x$ en la función logística, por lo que la hipótesis toma la forma de la ecuación 5.8.

$$h_\theta(x) = \frac{1}{1 + e^{\theta^\tau - x}} \tag{5.8}$$

Esta nueva hipótesis puede estimar la probabilidad de $y = 1$ con un valor de x a la entrada como se puede observar en la figura 5.4 y teniendo en cuenta la ecuación 5.9 que representa la estimación de la probabilidad cuando $y = 0$ ó $y = 1$.

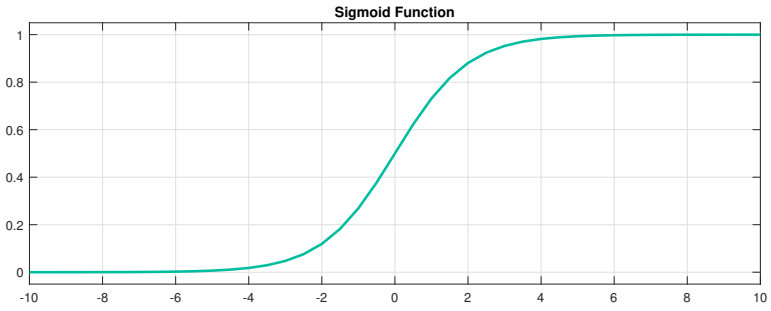


Figura 5.4. Función Sigmoidea o función logística

$$P(y = 0|x; \theta) + P(y = 1|x; \theta) = 1 \tag{5.9}$$

5.4.1. Función de costo

Es aquella función que determina el error entre el valor estimado y el valor real, con el fin de optimizar los parámetros del algoritmo de aprendizaje, este toma el valor de 1 ó 0 como se observa en la ecuación 5.10.

$$Cost(h_\theta(x), y) = \begin{cases} -log(1 - h_\theta(x)) & y = 0 \\ -log(h_\theta(x)) & y = 1 \end{cases} \tag{5.10}$$

La ecuación 5.10 se puede representar como una sola ecuación 5.11, finalmente para obtener la función de costo total es necesario encontrar el error en cada una de las iteraciones y en todas las muestras, por lo que surge la ecuación 5.12.

$$Cost(h_\theta(x), y) = -ylog(h_\theta(x)) - ((1 - y)log(1 - h_\theta(x)) \tag{5.11}$$

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} Cost(h\theta(x^{(i)}), y^{(i)}) \tag{5.12}$$

5.5. Redes Neuronales

Las redes neuronales artificiales están basadas en el comportamiento de las redes biológicas, donde el sistema nervioso está compuesto por una red de células individuales llamadas neuronas, que se encuentran ampliamente interconectadas, transfiriendo señales eléctricas entre sí, en la figura 5.5 se observan las partes de una neurona biológica, indicando de qué manera se interconecta con la neurona previa y la siguiente. A nivel matemático, una neurona se puede representar por medio de la ecuación de regresión logística 5.7, teniendo en cuenta que posee señales de entrada, pesos característicos, una única salida y una función de activación, que en este caso específico corresponde a una sigmoide.

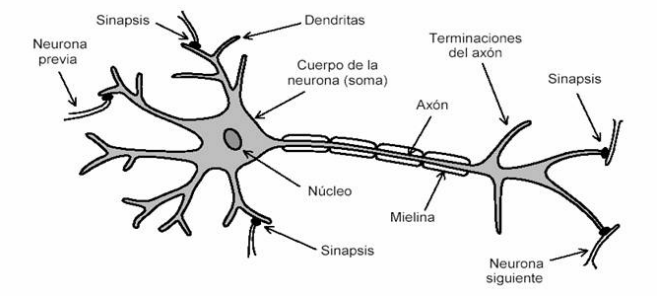


Figura 5.5. Modelo biológico neurona [27]

Como se puede observar en la figura 5.5, la neurona biológica se encuentra compuesta por diferentes partes las cuales son:

- *El cuerpo celular ó soma:* Aquí es donde se encuentra el material genético de la neurona, es la parte donde ocurren todos los procesos metabólicos de la neurona incluyendo la condensación de las moléculas necesarias para la correcta transmisión de señales eléctricas.

- *Dendritas:* Son las encargadas de captar los neurotransmisores producidos por la neurona más cercana y enviar la información química al cuerpo de la neurona para hacer que esta se active eléctricamente, estas son prolongaciones que nacen del cuerpo o

soma y que conforman una especie de ramas que recubren todo el centro de la neurona.

- *Núcleo:* Parte la cual controla la expresión del material genético, regulando todo lo que sucede en la neurona.

- *Axón:* Es el encargado de conducir el impulso eléctrico hasta los botones sinápticos.

- *Sinapsis:* Es la parte donde se liberan los neurotransmisores para informar a la siguiente neurona.

- *Mielina:* Es aquella sustancia compuesta de proteínas y grasas que rodea el axón de las neuronas, esta es imprescindible para permitir que el impulso eléctrico viaje a través de este a la velocidad correcta.

El sistema global del proceso llevado a cabo por la red neuronal consiste en las neuronas artificiales organizadas en capas, las interfaces de entrada y las de salida, según los principios descritos en su estructura. Una red neuronal consta de 3 partes:

- *Capa de entrada:* Compuesta por neuronas que reciben directamente datos o señales procedentes del entorno (capa roja en la figura 5.6).

- *Capa oculta:* No tiene conexión directa con el entorno, puede ser precedida por otras capas ocultas, o bien, por la capa de entrada. La cantidad de capas ocultas, varía de acuerdo a la complejidad deseada y a la no linealidad del modelo que se desea crear (capa azul en la figura 5.6).

- *Capa de salida:* Compuesta por un numero determinado de neuronas. Se emplea función de activación para los casos en los que el problema corresponde a clasificación. La capa de salida generalmente se toma para representar las puntuaciones de clase (capa amarilla en la figura 5.6).

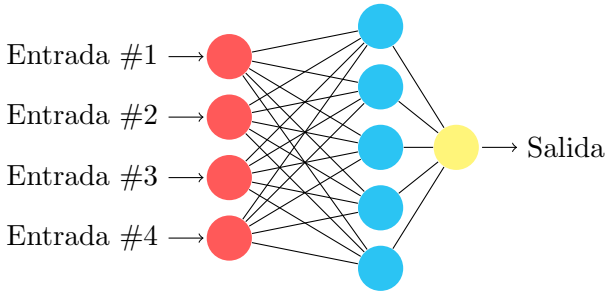


Figura 5.6. Arquitectura de una red neuronal

5.5.1. Redes Neuronales Convolucionales

Una red neuronal convolucional es un tipo de red neuronal artificial basada en aprendizaje supervisado que procesa sus capas imitando al corte visual del ojo humano para identificar distintas características en las entradas. Este tipo de redes neuronales se caracteriza por admitir solo imágenes en sus entradas, por lo que se usan para el filtrado de imágenes por medio de mascaras.

La convolución consiste en tomar grupos de píxeles cercanos de la imagen de entrada e ir operando matemáticamente un producto entre matrices, donde se emplean las neuronas de entrada y una matriz llamada kernel. En la figura 5.7 se presenta un ejemplo sobre la forma en la que se realiza una convolución. El kernel recorre todas las neuronas de entrada (capa de partida) de izquierda-derecha, de arriba-abajo y genera una nueva matriz de salida (capa convolucionada), que conforma la capa de neuronas ocultas de la red.

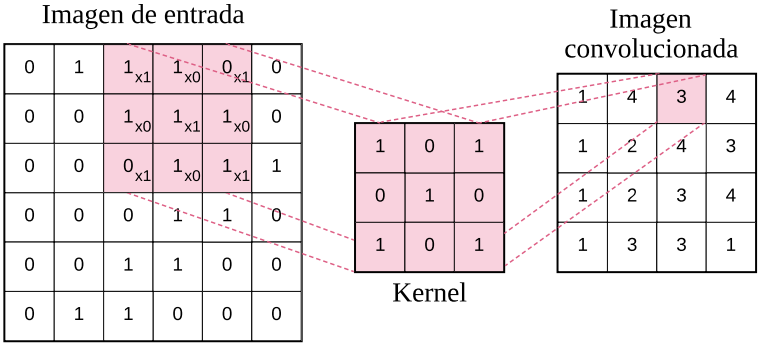


Figura 5.7. Ejemplo de convolución

Para llegar a los valores de la capa convolucionada se debe definir inicialmente el tamaño del kernel, en este caso 3x3. Se realiza el producto entre todas las secciones de la capa de partida, divididas en tamaños 3x3 y el kernel. El resultado de cada producto arroja una nueva matriz con las mismas dimensiones. Para crear la capa convolucionada, se toma la suma de números 1 de estas matrices resultantes. El producto resaltado en la figura 5.7 da como resultado una matriz diagonal de tamaño 3x3. La suma de estos tres dígitos corresponde al número 3 que se ubica en la capa convolucionada. Este tipo de capas corresponden a mapas de detección de características de la imagen y tienen jerarquía, ya que las capas iniciales detectan lineas, luego curvas y formas cada vez más elaboradas.

Las neuronas de esta arquitectura se encuentran dispuestas en 3 dimensiones: *ancho, alto y profundidad*. En la figura 5.8 se presenta la arquitectura de red, donde se tiene una capa de entrada, dos capas ocultas y una de salida. Se observa, además, la tridimensionalidad que genera la arquitectura internamente, donde las capas ocultas de la red procesan cada una de las dimensiones de la imagen (píxeles en eje x, píxeles en eje y, RGB). La red neuronal convolucional se encuentra en la capacidad de reducir la imagen completa a un solo vector de puntajes de clase, ya que se obtiene en la capa de salida un vector de dimensiones (1,1,número de clases).

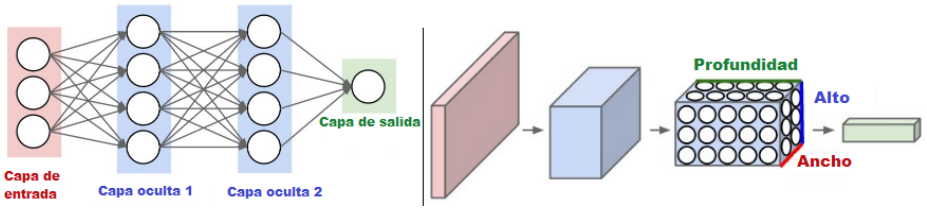


Figura 5.8. Arquitectura de una red convolucional[28]

5.5.2. Redes Neuronales Recurrentes

Las redes neuronales recurrentes poseen bucles de retroalimentación como se puede observar en la figura 5.9, los cuales les permiten almacenar en memoria la salida de la neurona en un tiempo anterior y emplear esta misma como entrada de la celda en el presente, lo cual permite trabajar por medio de series temporales.

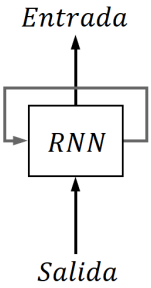


Figura 5.9. Arquitectura de una red neuronal recurrente

Las series temporales son una colección de observaciones de una variable, recogida secuencialmente en el tiempo, estas muestras no son independientes entre si, es decir que dependen del estado anterior, por lo que se emplean para predecir el siguiente valor [29]. Las funciones empleadas en las redes neuronales recurrentes son :

$$\sigma(x) = \frac{1}{(1 + e^{-x})} \tag{5.13}$$

$$tanh(x) = 2\sigma(2x) - 1 \tag{5.14}$$

$$f(x) = max(0, x) \tag{5.15}$$

La red neuronal recurrente tiene una función de activación (5.16), encargada de estimular la siguiente capa, generalmente se emplean ReLU (5.15) y Tanh (5.14). En la ecuación (5.16) la variable W_{ax}, representa el vector de pesos de la entrada, también se encuentra W_{aa} que representa el vector de pesos de la activación y finalmente b_a, que representa la tendencia de la activación.

$$a^{<t>} = g(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a) \tag{5.16}$$

Por otra parte se emplea una función de la salida (5.17), esta solo interviene en la ultima capa de la red y suele utilizar la función sigmoide (5.13) para la aproximación de la precisión de la red. En esta ecuación la variable W_{ya} representa el vector de pesos de la salida, por otro lado, b_y representa la tendencia que ha tenido la salida.

$$\hat{y}< t > = g(W_{ya}a^{<t>} + b_y) \tag{5.17}$$

Para calcular la pérdida en las redes neuronales recurrentes, es necesario calcularla para cada paso de tiempo, lo cual permite determinar si la red predice correctamente. Para esto se emplea la entropía cruzada como función de pérdida, en un paso de tiempo se puede obtener con la siguiente función 5.18:

$$L_t = y_t log(y^t) \tag{5.18}$$

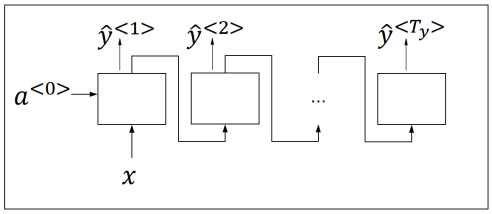
Donde:

- y_t es la salida de real del problema.
- y^t es la salida que la red predice en un instante de tiempo.

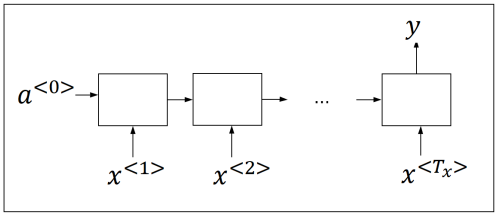
La pérdida total de la red es una suma de la pérdida en todos los instantes de tiempo y se representa con la ecuación 5.19.

$$L = \sum_{j=0}^{t-1} L_j \tag{5.19}$$

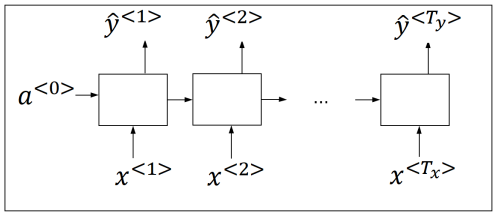
Las redes neuronales recurrentes, tienen diferentes tipos de arquitecturas, como se observa en la figura 5.10, donde se evidencia para todos los casos, la retroalimentación de los estados anteriores para llegar a una salida.



(a) Uno a muchos



(b) Muchos a uno



(c) Muchos a muchos

Figura 5.10. Tipos de Redes Neuronales Recurrentes [30]

Los tipos de redes neuronales recurrentes se encuentran compuestas de diferentes formas, por lo que sus usos varían, como se observa a continuación:

- *Uno a muchos 5.10(a):* Compuesta por una entrada y múltiples salidas, es empleada en problemas de predicciones de valores durante periodos de tiempo.

- *Muchos a uno 5.10(b):* Compuesta por muchas entradas y una única salida, es principalmente utilizada en problemas de análisis de texto, donde al tener un conjunto de palabras se extrae la idea del texto.

- *Muchos a muchos 5.10(c):* Compuesta por múltiples entradas y salidas, se usa normalmente en problemas de traducción.

5.5.3. *Long Short-Term Memory (LSTM)*

Son un tipo *especial* de redes neuronales recurrentes, que tienen como característica principal recordar información durante largos periodos de tiempo. Se encuentran compuestas por una celda y tres reguladores o puertas *(puerta de actualización, puerta de salida y puerta memoria)* como se observa en la figura 5.11 .

La diferencia de un LSTM sobre un RNN es la celda de memoria, que permite mantener información por largos periodos de tiempo a través de la obtención del estado de la capa en un tiempo anterior, logrando retro alimentar el estado presente de acuerdo al anterior.

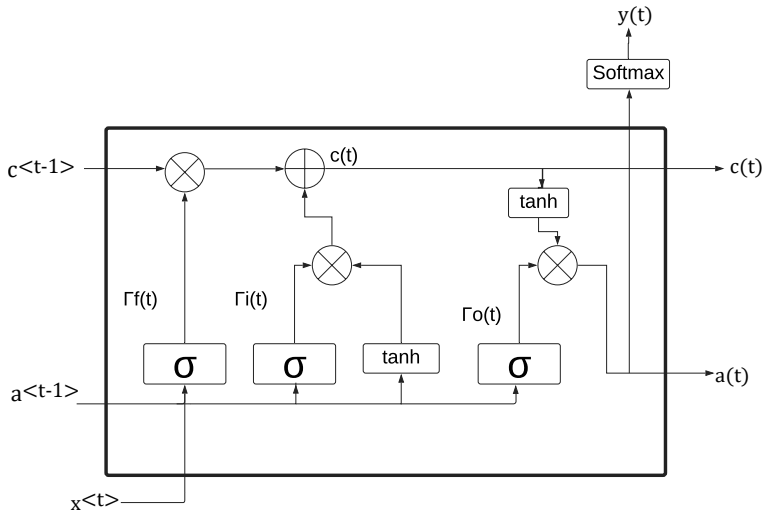


Figura 5.11. Estructura de una LSTM [30]

Este comportamiento en las LSTM, permite identificar qué tan importante es olvidar la información pasada o recordar la información

actual; todo esto se realiza en la capa oculta, por medio de puertas que permiten estas funcionalidades. Las ecuaciones que representan cada uno de los estados de una LSTM son:

- El estado de la celda es calculado por medio de la ecuación 5.20 y su representación gráfica es la figura 5.12(a)

$$c^{<t>} = (\sigma_u * \tilde{c}^{<t>}) + (\sigma_f * c^{<t-1>}) \tag{5.20}$$

- σ_f es la *puerta de olvido*, la cual tiene un valor entre 0 y 1. Es 0 para olvidar completamente el estado de la celda anterior, o 1 para mantenerlo. En esta puerta se evalúa una operación lineal seguida de una activación sigmoidea en la concatenación del valor anterior de la celda y la entrada actual, con W_f como pesos y b_f como sesgo, como se observa en la ecuación 5.21, la representación gráfica de esta celda, se encuentra en la figura 5.12(b)

$$\sigma_f = \sigma(W_f[a^{<t-1>}, x^{<t>}] + b_f) \tag{5.21}$$

- $\tilde{c}^{<t>}$ es el *Estado candidato de la celda*, según su peso $_c$ y su sesgo b_c:

$$\tilde{c}^{<t>} = \tanh(W_c[a^{<t-1>}, x^t] + b_c) \tag{5.22}$$

- σ_u es *Puerta de actualización*, tiene un valor entre 0 y 1, y representa la cantidad del estado candidato de la celda que se transferirá al estado de celda actual:

$$\sigma_u = \sigma(W_u[a^{<t-1>}, x^{<t>}] + b_u) \tag{5.23}$$

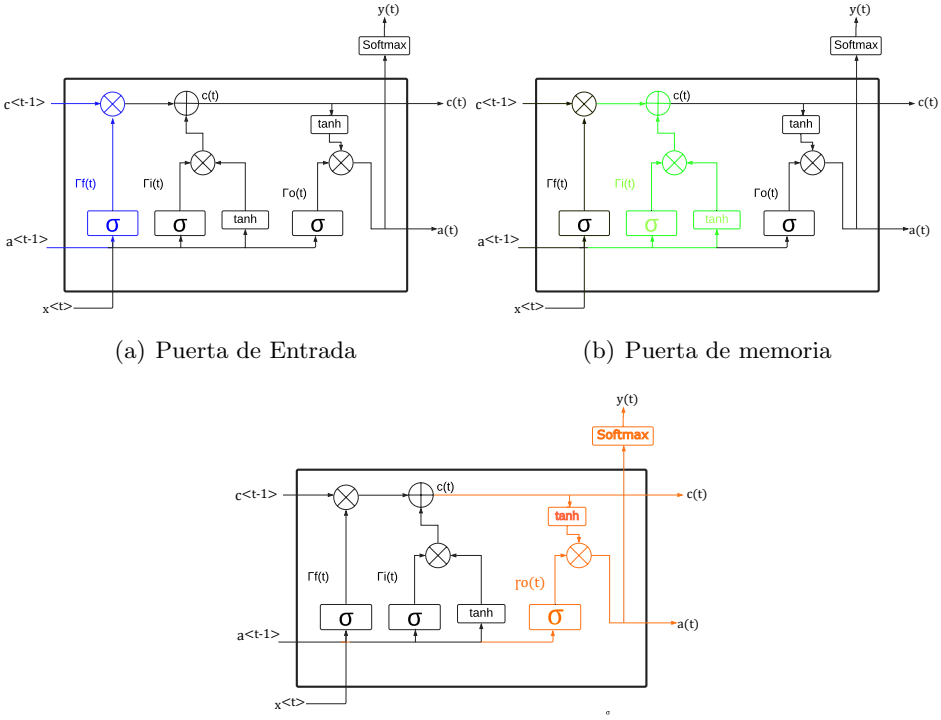
- Salida de la celda

$$a^{<t>} = \sigma_o * \tanh(c^{<t>})y^{<t>} \tag{5.24}$$

- Puerta de salida, su representación gráfica se observa en la figura 5.12(c)

$$\sigma_o = \sigma(W_o[a^{<t-1}, x^{<t>}] + b_o) \tag{5.25}$$

En la figura 5.12 se puede observar la representación gráfica de las ecuaciones anteriores:



(a) Puerta de Entrada

(b) Puerta de memoria

(c) Puerta de salida

Figura 5.12. Puertas LSTM [30]

5.6. *Support Vector Machine (SVM)*

Support Vector Machine es un algoritmo de aprendizaje supervisado empleado para la clasificación binaria; para ello construye un hiper-plano de separación con el margen máximo, como se observa en la figura 5.13, donde se emplean los vectores de soporte para la clasificación y se pueden eliminar aquellas muestras que se encuentren lejos del límite de decisión [2][32].

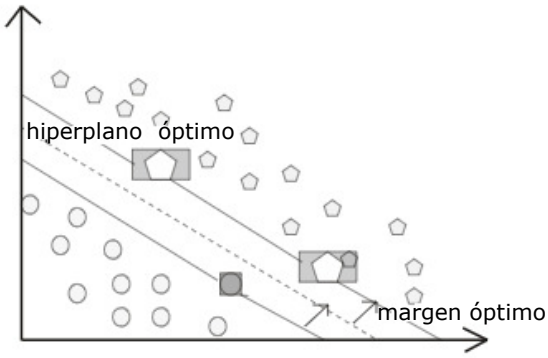


Figura 5.13. Margen de separación SVM[35]

Las SVM poseen dos esquemas de conjunto representativos: uno-versus-todos *(1 vs R)* y uno-versus-uno *(1 vs 1)*; por medio de los cuales se solucionan problemas multiclases, esto se lleva a cabo a través de la descomposición del problema en un grupo de problemas binarios predefinidos con el uso de la matriz ECOC (Error Correcting Output Codes).

5.6.1. Función de Costo

Es aquella función que determina el error entre el valor estimado y el valor real, esto con el fin de optimizar los parámetros del algoritmo de aprendizaje. En el caso de una SVM, la función de costo corresponde a la presentada en la ecuación 5.26

$$J(\theta) = C\sum_{i=1}^{m}[y^{(i)}Cost_1(\theta^\tau x^{(i)}) + (1-y^{(i)})Cost_0(\theta^\tau x^{(i)})] + \frac{1}{2}\sum_{j=1}^{n}\theta j^2 \quad (5.26)$$

Donde el $Cost_0$ y el $Cost_1$, representan el costo de cada uno de los valores que puede tomar la clasificación es decir 0 ó 1, teniendo en cuenta que es una clasificación binaria.

5.6.2. Kernel

El Kernel es aquella herramienta dentro de la SVM la cual obtiene una solución lineal, dentro del espacio de características que se busca clasificar, lo cual se convierte en una solución no lineal, en el espacio de entrada; el kernel es empleado cuando el problema de clasificación:

- Tiene más de dos variables predictoras.
- Posee curvas no lineales de separación.
- Es un caso donde el *dataset* no puede ser completamente separado.
- Es un problema de clasificación en más de dos categorías

Existen diferentes tipos de kernel, entre ellos se encuentran:

- **Lineal** representado en la ecuación 5.27.

$$< x, x' > \tag{5.27}$$

- **Polinomial** se encuentra representado en la ecuación 5.28, donde d es el grado del polinomio y r esta definido por el primer coeficiente.

$$(\gamma < x, x' > +r)^d \tag{5.28}$$

- **RBF(Función de base radial)** representado en la ecuación 5.29, donde γ debe ser mayor que 0.

$$exp(-\gamma||x - x'||^2) \tag{5.29}$$

- **Sigmoide** representado por la ecuación 5.30, donde r esta definido por el primer coeficiente.

$$(\tanh(\gamma < x, x' > +r)) \tag{5.30}$$

5.6.3. Límite de decisión o superficie de decisión

Es aquella hiper superficie que divide el espacio vectorial en 2 conjuntos como se puede observar en la figura 5.14, uno para cada clase que se vea involucrada, cuando el problema a solucionar por la SVM no es linealmente separable, es necesario emplear un kernel para realizar esta conversión, esto se realiza aumentando el numero de dimensiones, pasando así de una hiper superficie pequeña a un hiperplano con dimensiones mas grandes.

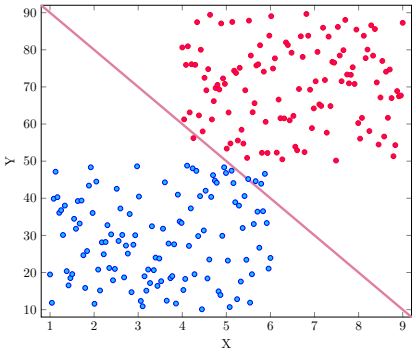


Figura 5.14. Límite de decisión

5.7. Métricas en Aprendizaje de Máquina

Las métricas para el aprendizaje supervisado son aquellas que permiten evaluar el desempeño del algoritmo de acuerdo a unas características especificas, algunas de estas métricas son matriz de confusión, recall, precisión, exactitud, puntuación F1, entre otras.

5.7.1. Matriz de confusión

Una matriz de confusión es aquella herramienta que ayuda a evaluar el desempeño que presenta un algoritmo de clasificación; la matriz de confusión funciona por medio de un conteo de aciertos y errores de cada una de las clases que se está clasificando y tiene la forma que se observa en el cuadro 5.1.

Clasificador			
		Predicción	
		Negativos	Positivos
Actual	Negativos	TP	FN
	Positivos	FP	TN

Cuadro 5.1. Matriz de confusión

Las variables que tienen relación con la matriz de confusión se explican a continuación:

- **TP:** Número de predicciones *correctas* de que un caso es *negativo*.
- **FN:** Número de veces en que la predicción es *positiva* cuando realmente el valor tendría que ser *negativo*.
- **FP:** Número de veces en que la predicción es *negativa* cuando realmente el valor tendría que ser *positivo*.
- **TN:** Número de predicciones *correctas* de que un caso es *positivo*.

5.7.2. Recall - Sensibilidad

El recall muestra la cantidad de verdaderos positivos que el modelo ha clasificado en función del número total de valores positivos, como se observa en la ecuación 5.31.

$$FPrate = \frac{TP}{TP + FN} \tag{5.31}$$

5.7.3. Precisión

La precisión, es la proporción de casos predichos positivos que fueron correctos 5.32.

$$Precision = \frac{TP}{FP + TP} \tag{5.32}$$

5.7.4. Accuracy - Exactitud

El accuracy indica el número de elementos clasificados correctamente en comparación con el número total de artículos, para esta métrica las clases deben estar equilibradas en cantidad de elementos, como se observa en la ecuación 5.33.

$$Exactitud = \frac{TP + TN}{TP + FN + FP + TN} \tag{5.33}$$

5.7.5. Puntuación F1

Esta métrica es la combinación de las métricas de precisión y sensibilidad. La mejor puntuación F1 es igual a 1 y la peor a 0, como se observa en la ecuación 5.34.

$$F1 = \frac{Precision * Recall}{Precision + Recall} \tag{5.34}$$

5.8. Estimación de la postura humana (*Pose estimation*)

El *pose estimation* es una herramienta que permite determinar la posición de la postura humana a partir de la identificación de articulaciones en imágenes o vídeos por medio de mapas de confianza 2D en puntos clave que son: cabeza, cuello, hombros, codos, muñecas, caderas, rodillas y tobillos. Posterior a los mapas de confianza se crea un conjunto de campos vectoriales de afinidades de partes que codifican el grado de asociación entre partes, por ejemplo, la asociación entre el cuello y un hombro, entre un hombro y un codo, etc, como se observa en la figura 5.15.

Figura 5.15. Salida Pose Estimation

El pose estimation funciona por medio de una máquina pose convolucional *CMP* [34], entrenada anteriormente en *Caffe Deep Learning Framework* con el *dataset* de posiciones humanas MPII; la CMP es la encargada de brindarnos un mapa de calor por cada punto de interés, luego de esto se debe seleccionar el valor máximo en cada uno de estos mapas, teniendo estos valores máximos se procede a asociarlos para identificar cuales pertenecen a la misma persona, la arquitectura de la CPM se puede observar en la figura 5.16.

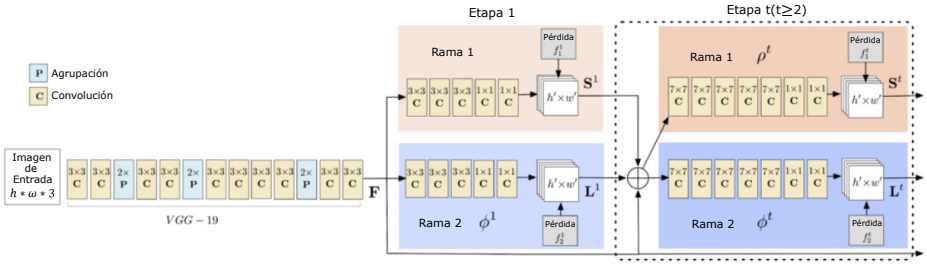


Figura 5.16. Arquitectura Pose Estimation [34]

5.9. Interpolación

Disponiendo de pares de datos $(xk, yk)k = 1, 2...n$. Se desea conocer el valor de y para un valor cualquiera de x en el intervalo $x1$ a xn. X está en el intervalo $(xk, xk + 1)$ como se muestra en la figura 5.17. Se traza la recta que pasa por los puntos (xk, yk) y $(xk + 1, yk + 1)$.

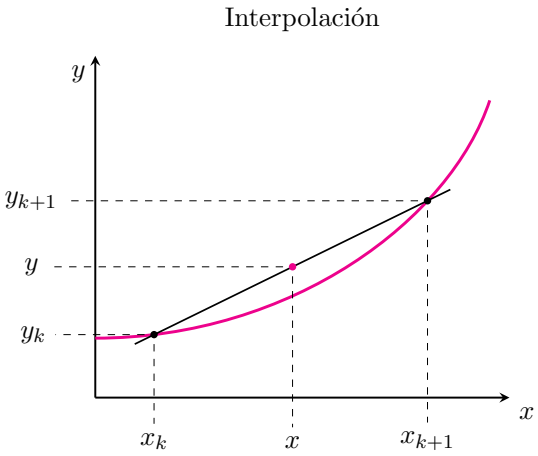


Figura 5.17. Interpolación

La representación matemática de la figura 5.17,esta expresada como:

$$y = \left(\frac{(x - x_k)\, y_{k+1} - (x - x_{k+1})\, y_k}{x_{k+1} - x_k} \right) \tag{5.35}$$

Capítulo 6

Desarrollo Metodológico

A continuación se presenta la metodología empleada para el desarrollo del proyecto, donde se aborda el problema a través de 5 etapas, cada una de ellas es necesaria y aporta para el cumplimiento de los objetivos planteados anteriormente. Las etapas son:

- *Investigación*: Se estudia del estado del arte y los algoritmos de aprendizaje supervisado que pueden ser aplicados a la imitación de movimientos en un robot humanoide. Se aprende a utilizar la herramienta RealSense D435, necesaria para la construcción del *dataset* de movimientos humanos.
- *Creación de herramientas necesarias*: Se construye el robot Poppy Torso. Se crea un *dataset* de videos RGB-D, compuesto por 2400 muestras de movimientos de las extremidades superiores, ejecutados por 6 personas de diferente contextura física. Se elabora un *dataset* de movimientos robóticos, donde se obtienen diferentes trayectorias que genera el robot al moverse de un punto aleatorio a uno final.
- *Implementación*: Se implementan los algoritmos de aprendizaje supervisado, correspondientes a clasificación y regresión, donde inicialmente se desarrolla un modelo para funciones sinusoidales, permitiendo observar el comportamiento frente a problemas secuenciales. Al obtenerse un resultado satisfactorio con estas

funciones, se implementan estas arquitecturas directamente con los *dataset*.

- *Evaluación*: Se evalúan cuantitativamente los modelos. Para los algoritmos de clasificación se emplea la matriz de confusión como la métrica de evaluación, observando la precisión de los algoritmos frente a cada uno de los movimientos clasificados. Para el algoritmo de regresión se utiliza el error de la posición final de la trayectoria frente a la posición deseada.

Capítulo 7

Construcción Robot Humanoide (Poppy Torso)

Poppy project, es un proyecto de código abierto realizado en Francia por Matthieu Lapeyre, por medio del cual se busca crear interdisciplinariedad y participación por parte de la comunidad para fomentar desarrollos en robótica y darle múltiples aplicaciones. El proyecto Poppy consta de 3 robots, cada uno de estos permite la realización de diferentes tareas que se deseen resolver. El presente proyecto de grado fue implementado con Poppy Torso, aprovechando su característica de poseer extremidades superiores.

7.1. Impresión de piezas

Para obtener como resultado final a Poppy Torso es necesario imprimir las piezas que componen su estructura, las cuales se encuentras disponibles libremente, y se pueden observar en la tabla A.1.

7.2. Ensamblaje de motores

Para el ensamblaje del robot humanoide se emplean motores Dynamixel MX Series, que contienen control PID y capacidad de movimiento de 360 grados contando con 4096 pasos, siendo así adecuados

para lograr movimientos complejos y naturales en el robot. Estos motores permiten una programación completa, permiten leer la posición actual del motor, la velocidad, la temperatura interna, el torque y la tensión de alimentación, además de la interacción con el usuario por medio de alarmas que se activan cuando el estado de la temperatura interna, torque (carga), tensión de alimentación, entre otras se ven afectadas.

Figura 7.1. Poppy Torso Construido

Para realizar la construcción de Poppy Torso, es necesario utilizar 13 motores Dynamixel, los cuales le permiten al robot humanoide realizar movimientos en: cabeza, cuello, hombros, codos y cintura. El resultado final de la construcción de este robot humanoide se observa en la figura 7.1.

Para el uso de Poppy Torso, es necesario tener en cuenta los aspectos técnicos y físicos, tales como cantidad de articulaciones, nombre de las mismas, como programar el robot, método de comunicación del mismo, entre otras, estas se pueden ver en la tabla 7.1

Aspectos técnicos	
Altura	$87cm$
Cantidad de piezas a imprimir	22
Número de articulaciones	13
Nombre de articulaciones	Ver figura 7.2
Tipo de comunicación	USB
Simulación	V-Rep o visualizador web 3D

Cuadro 7.1. Tabla Aspectos técnicos - Poppy Torso

En la figura 7.2 , se pueden observar las articulaciones necesarias para que el robot tenga movimiento. Los nombres presentados en la figura están compuestos por *lado_articulación_dirección del movimiento*, entendiendo que i es izquierda, d es derecha, y permite movimientos verticales, x movimientos horizontales y z movimientos de hacia adelante y atrás.

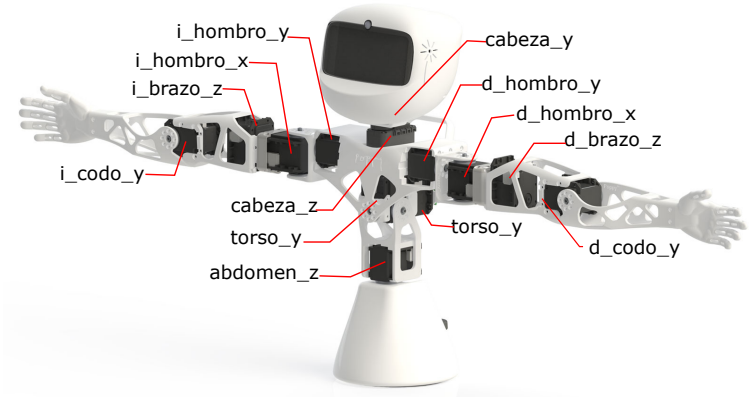


Figura 7.2. Poppy Articulaciones

Capítulo 8

Creación de *Dataset*

Para el desarrollo del aprendizaje de movimientos en un robot humanoide por medio de imitación se hace necesaria la creación de los conjuntos de datos a partir de los cuales se genera el entrenamiento de los algoritmos de aprendizaje. En la sección 8.1 se encuentra la descripción de la creación y procesamiento del *dataset* de movimientos humanos, donde se grabaron videos en RGB-D cuyo contenido consiste en el desplazamiento de las extremidades superiores de una posición a otra en humanos. Ya que el proyecto hace necesario el cambio de dominio del espacio de la persona al del robot, en la sección 8.2 se encuentra la descripción del *dataset* que especifica los ángulos en los que el robot desplaza sus grados de libertad para llegar a una posición objetivo.

8.1. *Dataset* de movimientos humanos

8.1.1. Elección de movimientos a implementar

Se seleccionan los siguientes puntos finales: brazos extendidos hacia arriba, abajo, al frente, a los lados y recogidos hacia el pecho (centro). El *dataset* consta de 20 movimientos, conformados por dos de los cinco puntos finales asignados como punto de partida y punto final, por ejemplo: de arriba hacia abajo, del frente hacia el centro, de abajo hacia arriba. En el *dataset* creado se nombran estos movimientos en inglés, de

la siguiente manera: Up-Down, Front-Center, Down-Up. Estos movimientos se seleccionan proyectando que sean útiles para realizar el aprendizaje de movimientos en el robot humanoide, no existe criterio médico para la selección de los mismos.

8.1.2. Software para la adquisición de datos

Para la grabación de los videos se emplea una cámara Intel RealSense D435, herramienta que permite obtener imágenes en RGB y su profundidad por medio de un proyector infrarrojo. Para obtener control de los videos a la hora de grabarlos, se realiza un software en python, empleando el framework QT para la realización de una interfaz gráfica. Esta GUI se muestra en la figura 8.1, contando con las siguientes secciones: 1. *Parámetros de captura*, donde los FPS y la resolución puede ser establecida; las secciones 2, 3 y 4 corresponden a *información del dataset*, permite la visualización y manipulación de la información de las personas, los movimientos y la ruta de almacenamiento de los videos; sección 5, *Visualización*, donde se observan las imágenes capturadas por la cámara (en RGB y profundidad) y la sección 6, *control de grabación*, donde la grabación inicia y se detiene.

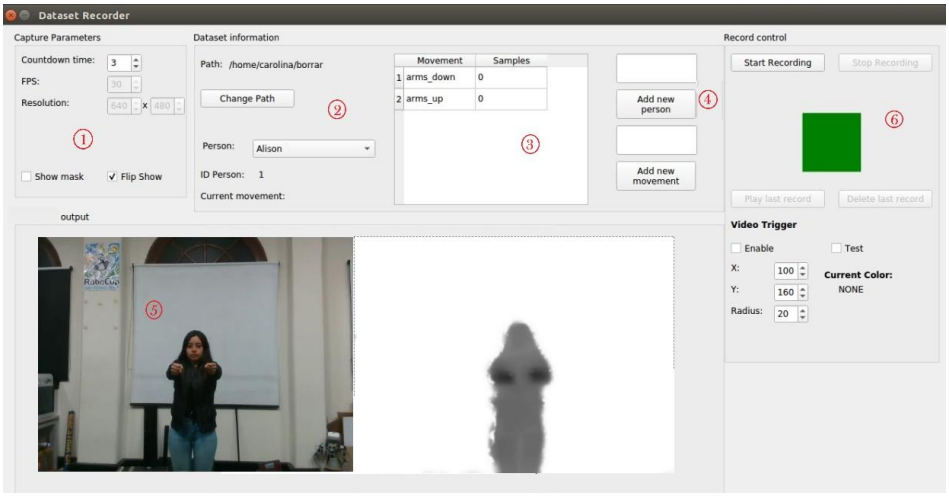


Figura 8.1. GUI empleada para creación de dataset

8.1.3. Configuración del entorno de grabación

Inicialmente se realiza la captura de los videos, se cuenta con la participación de 6 personas, donde cada una de ellas realiza 20 repeticiones de cada movimiento, estos videos se capturaron con la cámara RealSense a 2 metros de distancia de la persona que ejecuta los movimientos y a 1.2 metros de altura, las condiciones de luz corresponden a las condiciones promedio que se obtienen en ambientes de oficina, la cámara se conecta a una computadora que guarda los videos para cada movimiento, la configuración de la cámara corresponde a 30 FPS y 640x480 píxeles de resolución.

8.1.4. Identificación y extracción de información

Se busca extraer una secuencia que contenga la información sobre el cambio de posición de las extremidades superiores a través del tiempo en los videos, por esto, el primer reto es encontrar solución al problema de reconocimiento del movimiento del cuerpo a partir de la estimación de la pose, para esto se encuentra de gran utilidad el uso de modelos que desarrollan esta tarea, llamados Máquinas de pose.

Pose es el conjunto de puntos espaciales donde las coordenadas x, y son detectadas en puntos claves del cuerpo a partir de redes convolucionales, que generan mapas de calor de las partes del cuerpo desde imágenes RGB y luego realizan un post procesamiento para extraer los correspondientes puntos, para este proyecto de grado se emplea una red convolucional entrenada con el *dataset* de postura humana MPII. Esta red hace posible obtener la posición x y y de los puntos que se observan en la figura 8.2.

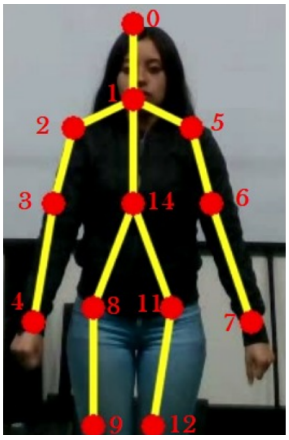


Figura 8.2. Puntos extraídos de la máquina de pose

Se consideran de utilidad para que el robot aprenda movimientos en las extremidades superiores los puntos del 2 al 7, correspondientes a los puntos de los hombros, codos y muñecas. Estas coordenadas se calculan de acuerdo a la distancia del pecho (punto 14), para garantizar que la altura de la persona y su posición dentro del rango de la cámara no afecte en el *dataset*.

En la figura 8.3 se observan los puntos extraídos por el *pose estimation*, durante la ejecución de un movimiento, logrando identificar el movimiento realizado, por lo tanto se emplean estos puntos como la información útil que servirá como entrada a los algoritmos de clasificación.

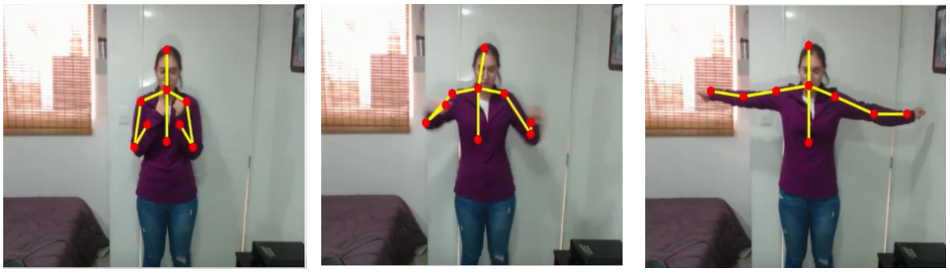


Figura 8.3. Puntos extraídos de un movimiento empleando *Pose estimation*

8.1.5. Normalización y organización del *dataset*

Los videos del *dataset* cuentan inicialmente con tiempos de duración diferentes entre sí, por este motivo se realiza interpolación en frames y en el tiempo para obtener un *dataset* con la misma cantidad de datos entre muestras y evitar pérdida de información, se interpola para obtener 66 frames por vídeo con la información x,y de los 6 puntos de interés, quedando finalmente en arreglos de 66x12, estos arreglos se organizan según movimiento en un diccionario conformado por 20 claves que corresponden a los movimientos seleccionados, cada clave contiene un arreglo de (120,66,12), 120 porque se tienen 20 videos grabados por 6 personas con apariencia física diferente en un ambiente controlado, logrando un total de 2400 muestras en el *dataset*; el 66 corresponde a la cantidad de frames que componen los videos y el 12 a los puntos que conforman el pose.

8.2. *Dataset* de movimientos robóticos

Con el fin de generar un aprendizaje en el robot para llegar a un punto objetivo, se plantea entrenar una red neuronal cuya entrada esté en términos del espacio del robot, es decir, los ángulos en los que el robot desplaza sus grados de libertad para llegar a una posición objetivo.

El *dataset* se captura por medio de la herramienta de visualización *RViz*, que se observa en la figura 8.4, donde el robot ejecuta los movimientos bajo un programa en Python que emplea una función en ROS llamada *Plan Movement*, que permite realizar un movimiento desde un punto inicial a uno final, siendo posible establecer estos puntos de forma aleatoria o explícitamente programados, retornando una trayectoria que consiste en una tupla que contiene la sucesión de ángulos en radianes que toma lugar cada uno de los motores para trasladarse de un punto a otro.

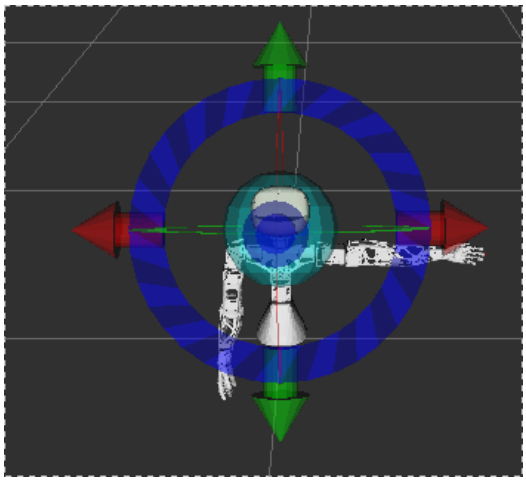


Figura 8.4. Poppy Torso simulado en RVIZ

Para crear el *dataset* de movimientos robóticos se emplea esta función, iniciando el movimiento en una posición aleatoria y finalizando en una posición objetivo. Estos puntos son interpolados a un valor de n ya que al ser aleatorio el punto inicial las longitudes de las tuplas varían. Una vez interpolado, se guarda cada secuencia en un diccionario donde las claves son los nombres de los movimientos objetivo: arriba, centro, lados, frente y abajo, dentro de cada clave hay un arreglo de tamaño (614,20,4), donde 614 corresponde al número de muestras tomadas, 20 al numero n al que fueron interpolados los datos de la trayectoria y 4 corresponde a los motores que contiene un brazo.

Capítulo 9

Clasificación de movimientos

A continuación se presentan los algoritmos empleados para la clasificación de movimientos del dataset y su comportamiento frente al mismo; las arquitecturas elegidas son una red neuronal recurrente *many to one* y una máquina de vectores de soporte. Para observar la conducta de los algoritmos frente a problemas de clasificación secuenciales, como lo son los movimientos, se evalúan por medio de matrices de confusión.

9.1. Red Neuronal Recurrente - LSTM Many to one

La entrada de la red esta representada por la secuencia de coordenadas cartesianas(*posición en* $\boldsymbol{x}$ *y posición en* $\boldsymbol{y}$) que interfieren en la ejecución de un movimiento, esta secuencia tiene en cuenta las articulaciones que interfieren en la acción; mientras que la salida esta compuesta por las etiquetas que se encuentran en el *dataset*, para obtener las coordenadas de las articulaciones, se realiza un procesamiento por medio del *pose estimation*, obteniendo la dimensión de la entrada de la red, la salida al ser un problema de clasificación tiene una dimensión de 1, esto se puede observar en la tabla 9.1.

Arquitectura	
PARÁMETRO	VALOR
Lote(b)	$Valor$
Pasos de tiempo(ts)	66
Número de entradas(p)	18
Número de posibles salidas(c)	20
Dimensión x	(b, ts, p)
Dimensión y	(b, c)

Cuadro 9.1. Tabla Arquitectura RNN many to one

Al tener todos los puntos, se separan los mismos en dos conjuntos, el primero es de entrenamiento (70 % del total), con este la red realiza la clasificación de los movimientos; el otro conjunto es el de prueba (30 % del total), con el cual se evalúa la precisión de la red. Al tener la arquitectura de la red, se realiza un barrido de parámetros el cual se puede evidenciar en la tabla 9.2

Parámetros barridos	
PARÁMETRO	RANGO
Numero de neuronas	128 - 512
Tasa de aprendizaje	0,0001 - 0,01
Pasos de entrenamiento	250 - 600
Tamaño del lote	10 - 50

Cuadro 9.2. Tabla Parámetros Barridos

A través del barrido de parámetros presentados en la tabla 9.2, se obtienen los 7 entrenamientos consignados en los la tabla B.1, de estos se observa que el mejor comportamiento fue el del compuesto *por 256*

neuronas, una tasa de aprendizaje de 0.001 y un lote de 15 muestras, con este entrenamiento se gráfico la precisión y la pérdida que presenta el modelo con el conjunto de prueba, obteniendo una precisión de 93,2 % y una pérdida que desciende hasta alcanzar un valor cercano a 0, lo cual se puede ver en la figura 9.1, también se puede observar que el algoritmo converge sobre la iteración 250 , la pérdida empleada en el modelo es la entropía cruzada, pues es la recomendada en problemas de clasificación de múltiples clases.

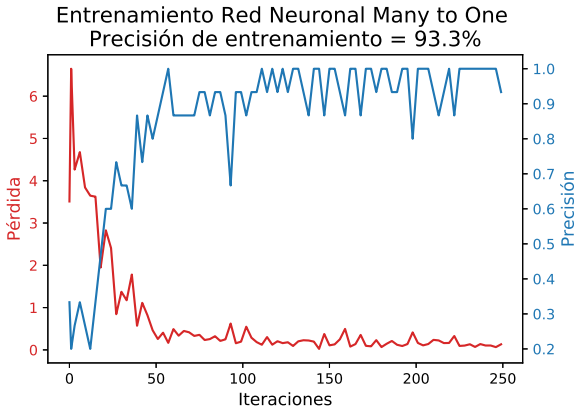


Figura 9.1. Precisión y pérdida Many to one

9.1.1. Evaluación cuantitativa

La evaluación cuantitativa del algoritmo se realiza al entrenamiento con las características presentadas anteriormente, esta evaluación se realiza a través de una matriz de confusión la cual se crea a partir del *dataset* de prueba, por medio de esta se observa la relación de las clases a etiquetar y las clases etiquetadas por el modelo, esto se puede observar en la figura 9.2.

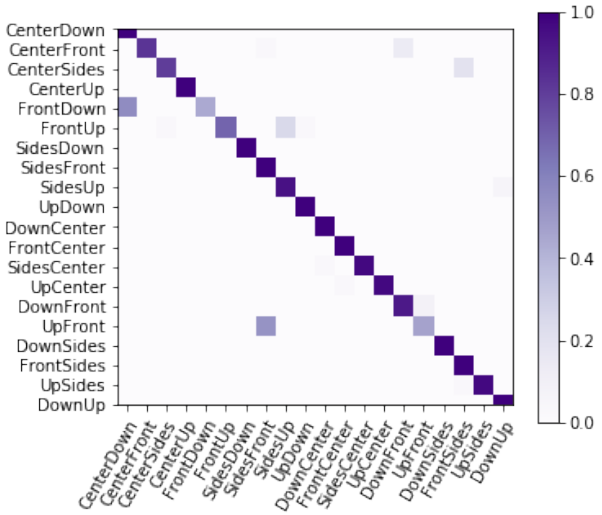


Figura 9.2. Matriz de confusión Many to one

En la figura 9.2, se puede observar que la red *many to one*, presenta una precisión total del 91 %, contando con una precisión superior al 80 %, en dieciséis de los veinte movimientos clasificados. Uno de los movimientos que suele hacer una predicción incorrecta *Center* figura 9.3(a), en esta predicción se falla dando como resultado el movimiento *Front* figura 9.3(b), y viceversa, esto ocurre por que al realizar estos movimientos las articulaciones se encuentran en posiciones similares, específicamente la mano se encuentra espacialmente en las mismas coordenadas para la cámara, como se puede observar en la figura 9.3.

(a) Centro (b) Frente

Figura 9.3. Posición Manos RNN

9.2. Máquina de vectores de soporte (SVM)

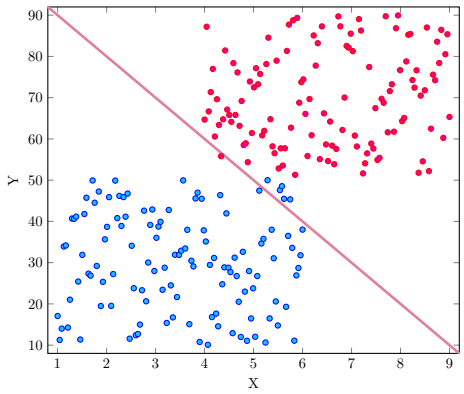


Figura 9.4. SVM

Para la implementación es necesario tener en cuenta que las SVM se suelen entrenar, por medio de dos pasos:

1. Transformar los datos de entrada en un espacio de características altamente dimensionales, especificando el kernel.

2. Resolver un problema de optimización cuadrática que se ajuste a un hiperplano óptimo para clasificar el numero de clases.

Para el proyecto se empleo la librería **sklearn**, la cual permite variar parámetros en la creación de la SVM,como se puede observar en la tabla 9.3.

Parámetros skalearn	
PARÁMETRO	DEFINICION
Parámetro de Regularización (C)	Regulariza la estimación
Kernel	Tipo de núcleo, empleado para aumentar la dimensión del hiperplano
Grado Polinomial(d)	Se emplea para el kernel polinomial
Tamaño del lote	Valor

Cuadro 9.3. Tabla Parámetros Barridos

9.2.1. SVM

La entrada de la SVM esta representada por la secuencia de coordenadas cartesianas(*posición en* $\boldsymbol{x}$ *y posición en* $\boldsymbol{y}$) que interfieren en la ejecución de un movimiento, esta secuencia tiene en cuenta las articulaciones que interfieren en el desplazamiento de un punto a otro; mientras que la salida son las etiquetas que se encuentran en el *dataset*, para obtener las coordenadas de las articulaciones, se realizó un procesamiento por medio del *pose estimation*, obteniendo la dimensión de la entrada de la red, la salida al ser un problema de clasificación tiene una dimensión de 1, esto se puede observar en la tabla 9.4.

Arquitectura	
PARÁMETRO	VALOR
Lote(b)	$Valor$
Pasos de tiempo(ts)	66
Número de entradas(p)	18
Número de posibles salidas(c)	20
Dimensión x	$(p * ts, b)$
Dimensión y	(b)

Cuadro 9.4. Tabla Arquitectura RNN many to one- dataset

Al tener el todos los puntos, se separan los mismos en dos conjuntos, el primero es de entremetimiento (70 % del total), con este la SVM realizara la clasificación de los movimientos; el conjunto otro es el de prueba (30 % del total), con el cual se evalúa la precisión de la SVM, frente a un conjunto de muestras no conocido. Al tener la arquitectura de la SVM, se realizó un barrido de parámetros el cual se puede evidenciar en la tabla 9.5

Parámetros barridos	
PARÁMETRO	RANGO
Kernel	*Linear - Poly - RBF*
Tasa de aprendizaje	0,01 - 0,1
C	0,5 - 1
d	3 - 6

Cuadro 9.5. Tabla Parámetros Barridos

A través del barrido de parámetros presentados en la tabla 9.5, se obtienen los 5 entrenamientos consignados en los la tabla C.1, los cuales

poseen algunas de las posibles combinaciones de los parámetros ya mencionados, de estos se observa que el mejor comportamiento fue el del *tamaño del lote de 250, un kernel polinomial, con un grado de polinomio de 0.5, una tasa de aprendizaje de 0.1 una regularización de 0.5*, con este entrenamiento se gráfico la validación cruzada y el valor de la precisión durante el entrenamiento. La validación cruzada consta en dividir el *dataset* en dos subconjuntos, el primero sera el conjunto de prueba y el otro sera el conjunto de validación, el conjunto de prueba se va a dividir en pequeños subconjuntos con los cuales se realizara el entrenamiento, luego se indica como se comporta el algoritmo de acuerdo al conjunto de validación. En la figura 9.5, se puede observar la precisión durante el entrenamiento y la validación cruzada.

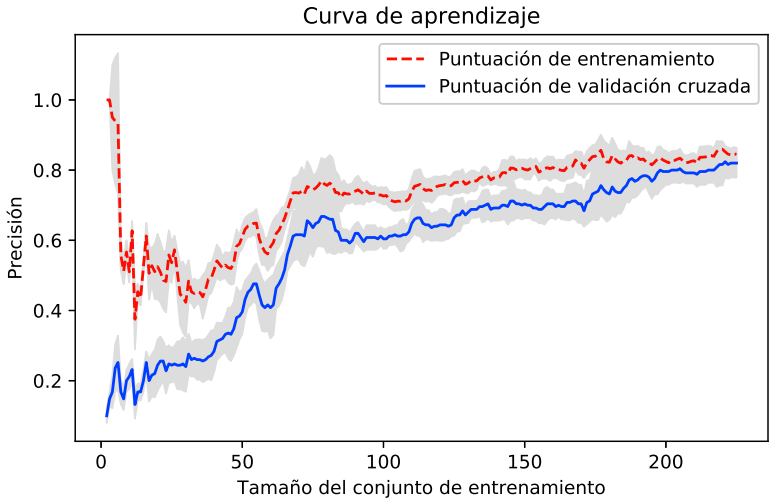


Figura 9.5. Curva aprendizaje con validación cruzada

Al obtener los resultados deseados se procede a realizar pruebas con los diferentes kernel y variando los valores de parámetros como *gamma*,C y el *tamaño del lote*, lo cual se puede observar en la tabla C.1, de la tabla C.1, se puede observar que el kernel *lineal* y el kernel *polinomial*, dan mejores resultados; el kernel lineal brinda buenos resultados con tamaño del lote relativamente pequeños, lo que representa que el conjunto de entrenamiento

para este kernel no necesariamente tiene que ser grande; para el kernel polinomial, se puede observar que tiene buen comportamiento al tener un grado menor o igual a 6, y de la misma forma que el kernel lineal el conjunto de entrenamiento puede ser pequeño.

9.2.2. Evaluación cuantitativa

La evaluación cuantitativa de la máquina de soporte de vectores,se encuentra compuesta por una matriz de confusión para tener mayor claridad de los movimientos clasificados y su precisión 9.6, aplicado al entrenamiento mencionado anteriormente, contando con una precisión de 91,94 %.

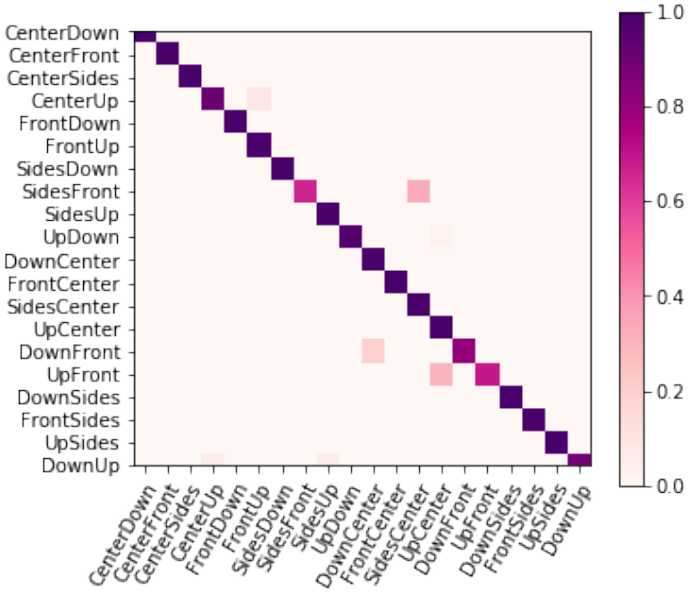
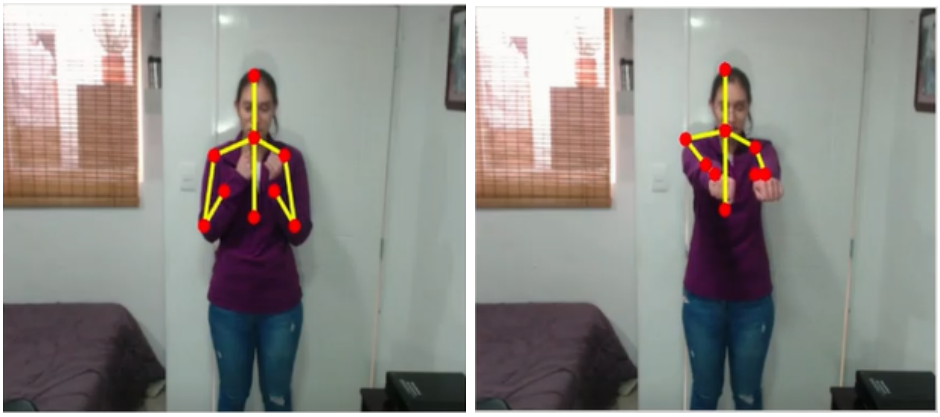


Figura 9.6. Matriz de confusión SVM

En la figura 9.6, se puede observar que la máquina de soporte vectorial, cuenta con una precisión total del 89 % superior al 80 %, en quince de los veinte movimientos clasificados. Los movimientos que suele confundir el algoritmo son *Front* (figura 9.7(b)) y *Center*(figura 9.7(a)), esto ocurre por que al realizar estos movimientos las articulaciones se encuentran en

posiciones similares, específicamente la mano se encuentra espacialmente en las mismas coordenadas, para la cámara, como se puede observar en la figura 9.7.



(a) Centro (b) Frente

Figura 9.7. Posición Manos SVM

Capítulo 10

Generación de trayectorias

A continuación se presenta la explicación del algoritmo empleado para la generación de trayectorias, que posibilita la ejecución de movimientos en las extremidades superiores del robot Poppy Torso. Para evaluar su desempeño a nivel cuantitativo se calcula el error de posición por medio de distancia euclidiana.

10.1. Red Neuronal Recurrente - LSTM One to Many

La entrada de esta red corresponde a trayectorias que siguen los motores (en ángulos) para que un brazo desde un punto aleatorio de partida, llegue a uno final. Se tienen cinco posiciones objetivo teniendo en cuenta el *dataset* de movimientos humanos, son: brazos hacia arriba, abajo, a los lados, al centro y al frente. Se realiza un entrenamiento por movimiento objetivo, donde a partir de la posición actual del robot (vector de dimensiones $1x4$, ya que un brazo contiene 4 motores), la red genera una secuencia final (de dimensiones $20x4$), que describe la variación de los motores en el tiempo.

Los datos de entrenamiento se parten en 70 % para entrenar y 30 % para realizar pruebas. Esta red neuronal se implementa sobre el framework de Tensorflow Keras. Al tratarse de regresión, se evalúa únicamente el error, la precisión así como las matrices de confusión no sirven en este caso para

evaluar a la red.

Los ángulos registrados en la tabla 10.1, corresponden a los valores que deben tomar los motores codo_y, brazo_z, hombro_y, hombro_x respectivamente para ejecutar los movimientos descritos. Estas articulaciones se pueden observar en la figura 7.2 del capítulo *Construcción Robot Humanoide (Poppy Torso)*.

POSICIÓN	ÁNGULO POR MOTOR (rad)			
Center	−1,57,	1,39,	−0,61,	2,09
Up	1,57,	1,22,	0,	0
Down	−1,57,	1,57,	0,	0
Sides	0,	0,	0,	0
Front	0,	1,57,	0,	0

Cuadro 10.1. Tabla de ángulo por motor

Con el fin de conocer con qué valores de parámetros se obtienen los mejores resultados, se realiza un barrido de los parámetros de entrenamiento, con los cuales se obtiene para algunos casos que las predicciones corresponden a la trayectoria esperada. En la tabla 10.2 se presentan los parámetros barridos y los rangos de valores que tomaron.

PARÁMETRO	RANGO
Tamaño del lote	12 - 28
Pasos de entrenamiento	500 - 1000
Número de neuronas	512 - 1024

Cuadro 10.2. Tabla Parámetros Barridos

En el anexo D.1 se observan las pérdidas de entrenamiento y validación, donde se obtiene menor pérdida de entrenamiento cuando los parámetros tienen los siguientes valores: tamaño de lote de 28, 1024 neuronas y 500 pasos de entrenamiento. Con estos valores de parámetros se realiza la figura 10.1 que contiene la pérdida del modelo a lo largo del entrenamiento. Se emplea como métrica el error cuadrático medio.

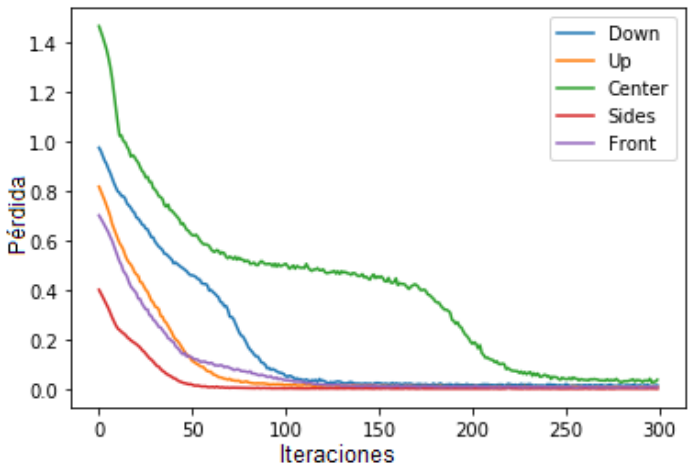


Figura 10.1. One to many pérdida por movimiento

En la figura 10.1, se observa que la pérdida de todos los entrenamientos converge, logrando llegar a valores cercanos a 0. A partir de la información de esta figura y lo registrado en la tabla 10.1, se puede inferir que existe una relación entre la velocidad con la que convergen los algoritmos y la complejidad del movimiento en términos de los ángulos que toman los motores. Se observa que el entrenamiento para el movimiento *Sides* converge al rededor de las 50 iteraciones y que el valor en radianes de los ángulos es 0 para todos los motores. Por otro lado, cercano a las 100 iteraciones convergen los entrenamientos de los movimientos *Down*, *Up* y *Front*, donde uno o dos motores obtienen valor diferente de 0.

Finalmente, el movimiento mas complejo según el número de iteraciones que requiere para converger es *Center*, en el que todos los

motores toman un valor diferente de 0 y requiere 5 veces las iteraciones que requiere el movimiento *Sides* para converger, esto quiere decir que la arquitectura aprende con mayor facilidad ante datos simples, sin embargo, tiene la capacidad de generar también trayectorias que traduzcan a movimientos más complejos en el robot.

En la figura 10.2 se presenta la trayectoria que genera la red entrenada y ejecuta el robot en cada uno de los motores cuando se da la instrucción de realizar el movimiento Center teniendo como partida la posición Sides.

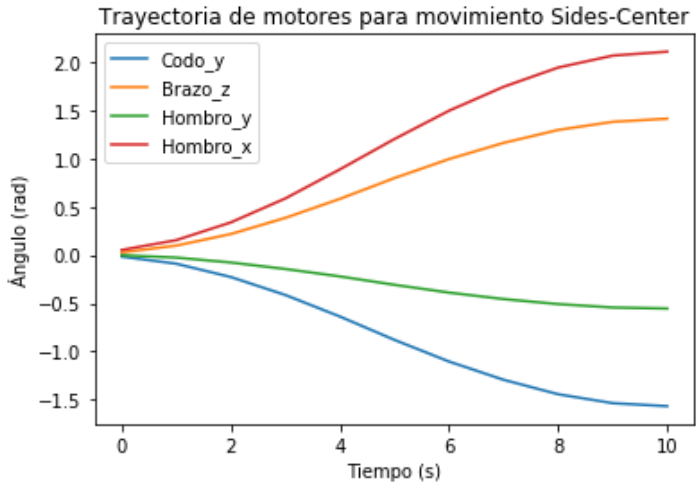


Figura 10.2. Trayectoria de motores para Sides-Center

10.1.1. Evaluación cuantitativa

Se calcula el error empleando la ecuación de la distancia euclidiana de los valores finales (en radianes) que toma cada motor en la predicción dada por los entrenamientos con respecto al valor real que debe tomar cada uno de los motores, con el fin de observar cuál es la distancia de cada motor con respecto al punto objetivo. En la figura 10.3 se muestra cómo disminuye a través del tiempo el error de posición al ejecutarse el movimiento Center, teniendo como partida la posición Sides, donde aproxima el valor a 0 para la totalidad de los motores.

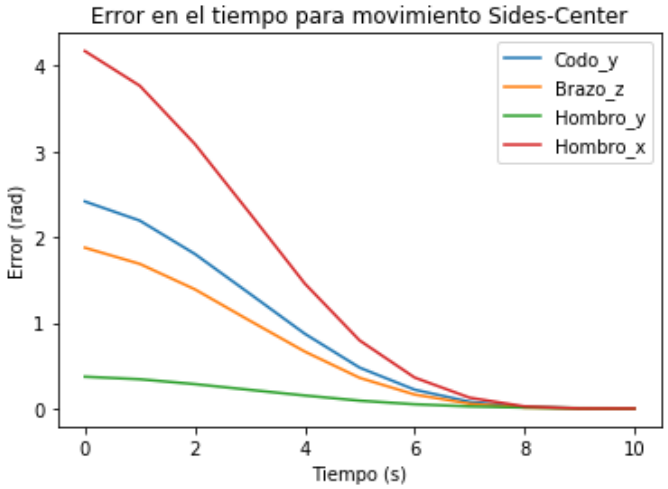


Figura 10.3. Error en el tiempo para movimiento Sides-Center

El error de posición obtenido en cada motor por movimiento se encuentra en la figura 10.4, donde se puede observar que el error máximo se presenta en el motor brazo_z, movimiento Down, siendo 0.025 radianes, sin embargo, los otros motores presentan error muy cercano a cero para este movimiento. El promedio total de los errores que se muestran en la figura 10.4 es 0.035 radianes que corresponde a 2 grados deg.

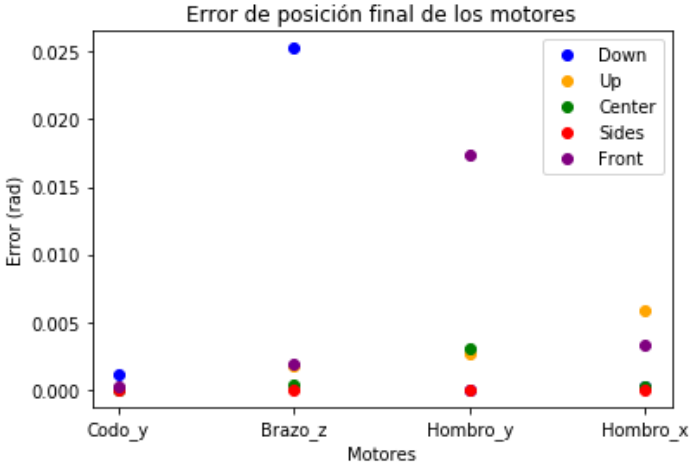


Figura 10.4. Error de posición final de los motores

En la figura 10.5 se presenta la posición obtenida con la red y la posición objetivo (presentada en tabla 10.1) que debe tomar cada motor para realizar el movimiento Down. A pesar de ser el que genera mayor error de posición, se puede observar que el comportamiento de los motores al finalizar la ejecución del movimiento es muy similar al deseado, por lo que se deduce que el robot alcanza visiblemente las posiciones objetivo empleando los modelos entrenados con la arquitectura seleccionada.

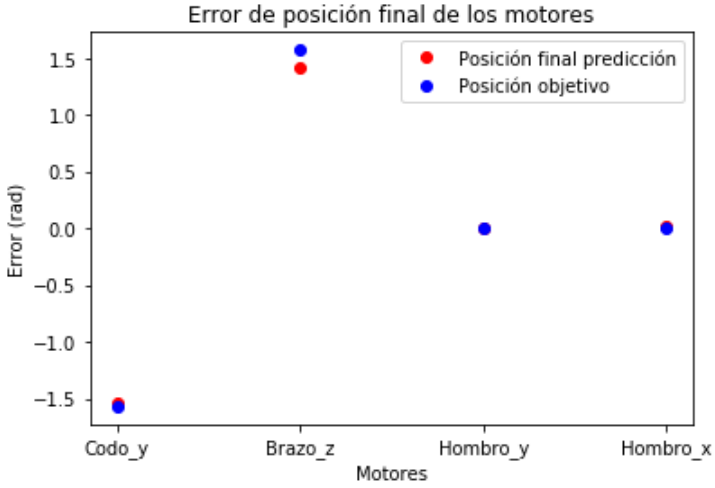


Figura 10.5. Posición de los motores para el movimiento Down

En la figura 10.6 se observa una trayectoria que genera el modelo entrenado en el robot Poppy Torso al recibir la orden de ejecutar el movimiento Up-Down.

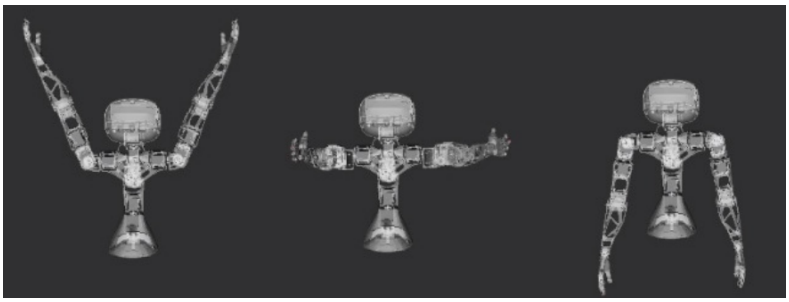


Figura 10.6. Movimiento ejecutado por Poppy

En el siguiente link se observa un video en el que se generan 5 trayectorias, mostrando el funcionamiento obtenido con las redes entrenadas: `https://youtu.be/Uf49NgZBufI`.

Capítulo 11

Integración del sistema

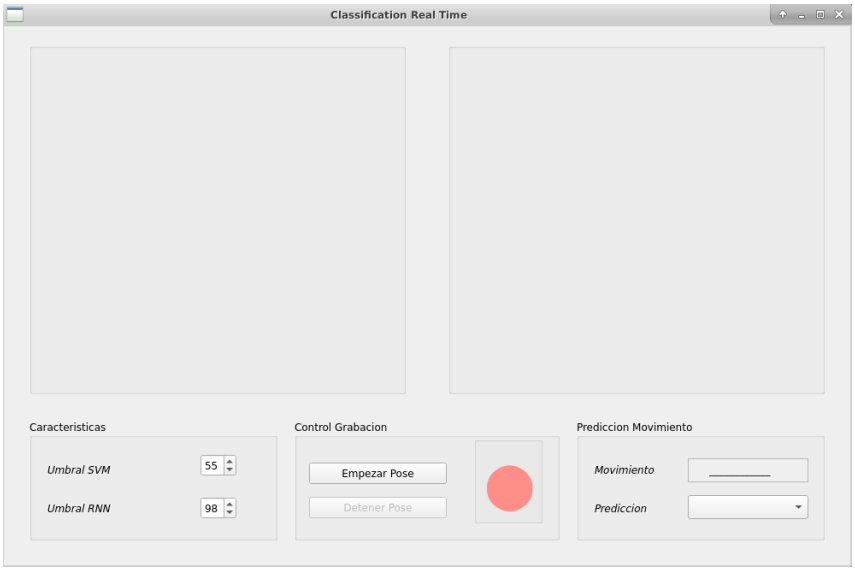


Figura 11.1. Interfaz tiempo Real

Se realizó una aplicación como se observa en la figura 11.1, la cual permite al usuario realizar movimientos y verlos reflejados en la herramienta de visualización **RVIZ**. La interfaz, es aquella aplicación con la cual interactúa el usuario, esta interfaz se realizó con PyQt5 y es la que le permite al usuario decidir que algoritmo quiere que clasifique su movimiento, permite la visualización del movimiento, de la estimación de

la pose y finalmente le permite al usuario que interactué con los umbrales de interacción de cada uno de los algoritmos.
La interfaz se encuentra compuesta por los siguientes elementos:

1. Visualización del movimiento realizado por el usuario en tiempo real.
2. Visualización de la estimación de pose.
3. Controlador del umbral SVM .
4. Controlador del umbral RNN.
5. Botón para iniciar la visualización del movimiento en tiempo real y la estimación de pose.
6. Contador decreciente para iniciar la visualización del movimiento y la estimación de pose.
7. Botón para detener la visualización del movimiento en tiempo real y la estimación de la pose.
8. Predicción de movimiento.
9. Selector de algoritmo de aprendizaje
10. Indicador de cuando se está estableciendo los servicios de ROS, empleados por el simulador, este también indica cuando se esta ejecutando un movimiento en el simulador.

Esto se puede observar en la gráfica 11.2, algunos de los elementos mencionados en los ítems no se encuentran visibles en el momento de ejecutar la aplicación, pues es necesario cumplir con algunas condiciones para que estos sean visibles.

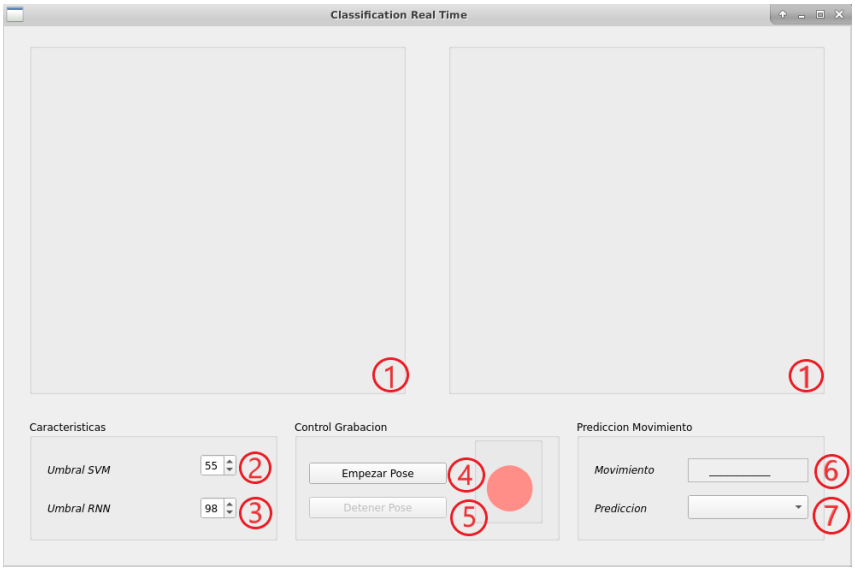


Figura 11.2. Interfaz tiempo Real Partes

Cuando la interfaz ha sido inicializada se mostrará el indicador que verifica que los servicios de ROS como se observa en la figura 11.3, si estos se encuentran se habilitarán las otras funcionalidades de la aplicación.

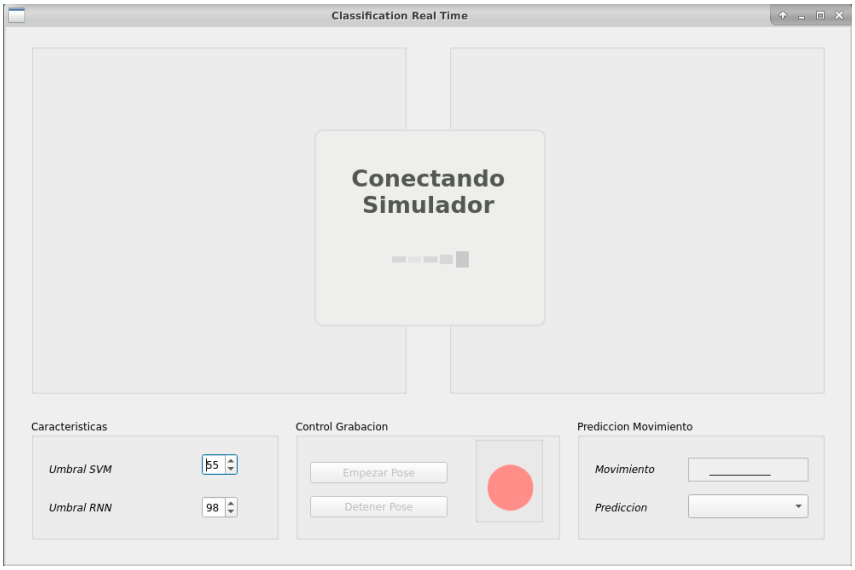


Figura 11.3. Servicios de ROS

Si la aplicación no encuentra los servicios de ROS, necesarios para la correcta ejecución del simulador se observará una alerta como en la figura 11.4, el usuario debe proceder a verificar los servicios.

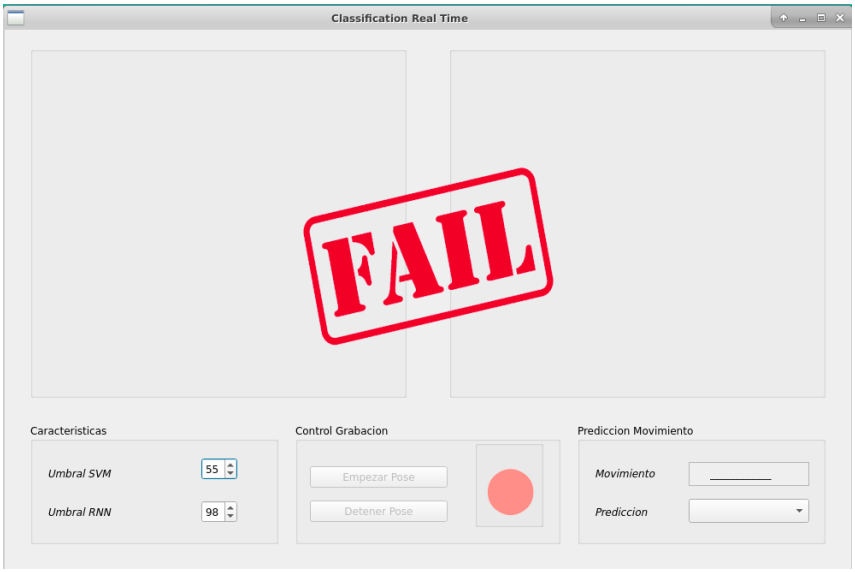


Figura 11.4. Servicios ROS no encontrados

Al inicializar los servicios en el simulador se observará que el robot mueve los brazos hacia abajo mediante un indicador que dice *Ejecutando movimiento* como se observa en la figura 11.5, cuando el robot termine el movimiento, se le permitirá al usuario empezar la interacción con las funcionalidades de la interfaz, es decir ya tendrá habilitado el botón de empezar la visualización y podrá elegir el algoritmo.

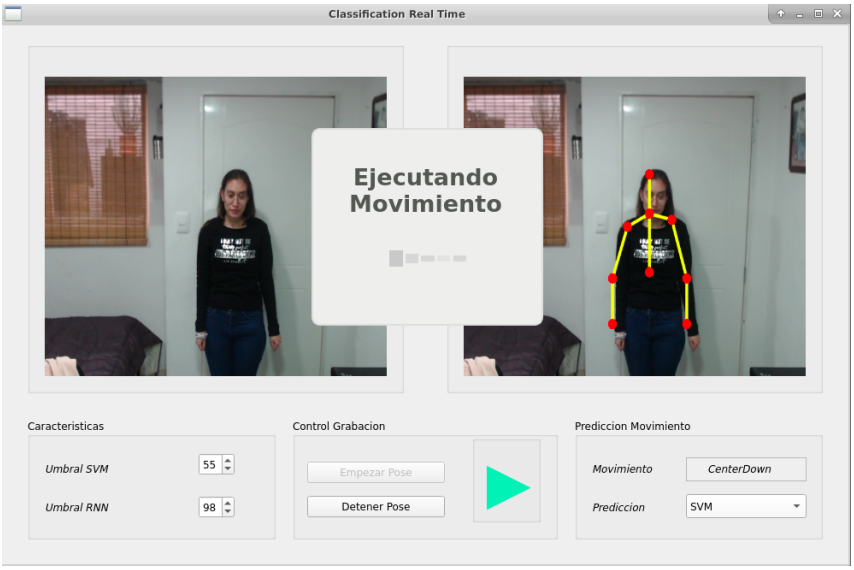


Figura 11.5. Interfaz ejecutando movimiento

En el selector de algoritmo de predicción, etiquetado como *Prediction*, asegurarse seleccionar una de las opciones disponbles, correspondientes a los dos métodos de clasificación disponibles: *SVM* y *LSTM*.

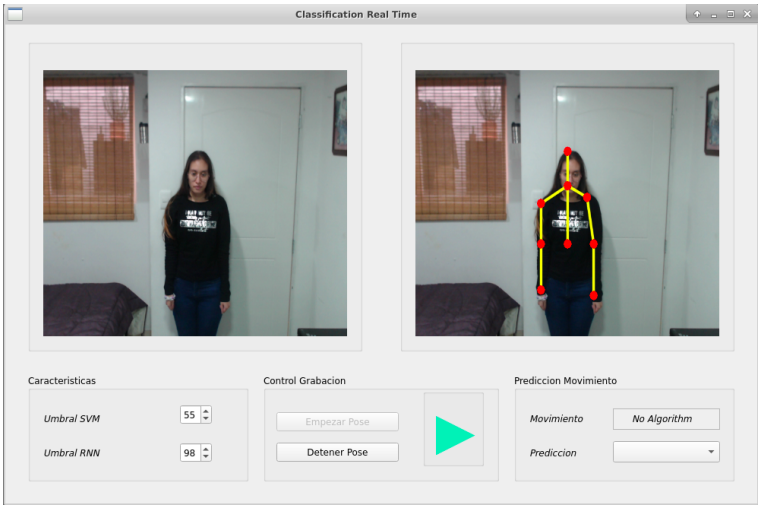


Figura 11.6. Interfaz sin algoritmo seleccionado

Si el usuario ya eligió el algoritmo, pero no cuenta con los puntos del

pose necesarios para realizar el movimiento o en su defecto no hay una persona; en el indicador de la predicción se verá *No person*, como se puede observar en la figura 11.7.

Los puntos necesarios para que la aplicación realice la predicción es de 9, esto implica los puntos de *muñecas, codos, hombros, tronco, cuello y cabeza*; es importante tener encuentra que al dibujar el pose en la aplicación, se omiten los puntos de las extremidades inferiores, esto se realiza por que el proyecto tiene en cuenta únicamente a las extremidades superiores, evitando interferencias en la predicción.

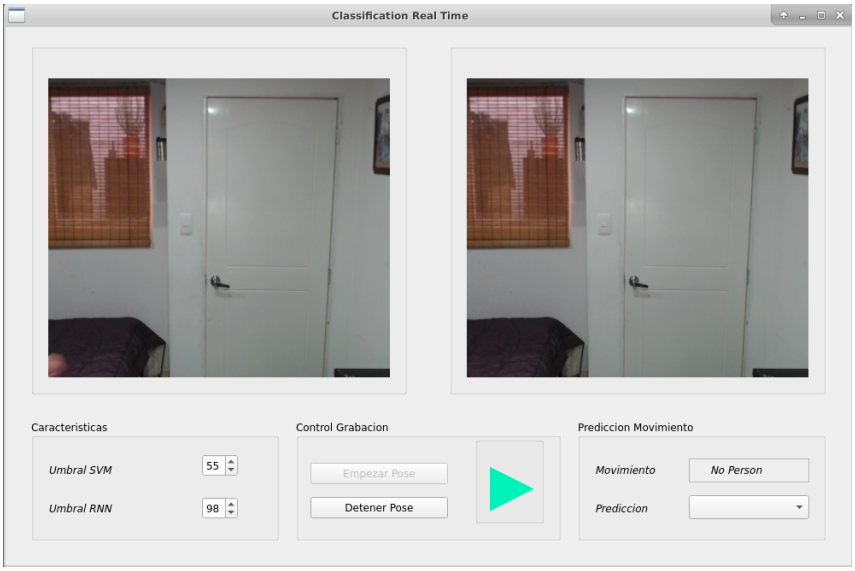


Figura 11.7. Interfaz sin Persona

Si el usuario cuenta con todos los puntos y ya eligió el algoritmo para la predicción, puede realizar el movimiento y en el indicador de predicción se podrá observar cual fue el movimiento que el algoritmo predijo.

Al tener una predicción de la SVM o la red *many to one*, que proviene de clasificar el movimiento que ha ejecutado la persona, se observa como trabajan en conjunto las dos predicciones, tanto de clasificación como regresión, ya que el robot Poppy torso, desde la herramienta de visualización *Rviz* ejecuta un movimiento, este movimiento ejecutado es el resultado de la predicción de la red *one to many* y consiste en ir a la

posición destino que predijo el algoritmo de clasificación, es decir si la predicción del algoritmo de clasificación fue de frente hacia abajo(Front-Down), el simulador ejecutará un movimiento desde la posición en donde se encuentra el robot hasta abajo (Down), mostrando la trayectoria que predice el algoritmo de regresión. En la figura 11.8 se puede observar la predicción del robot para llegar desde la posición centro hasta arriba (Down-Center).

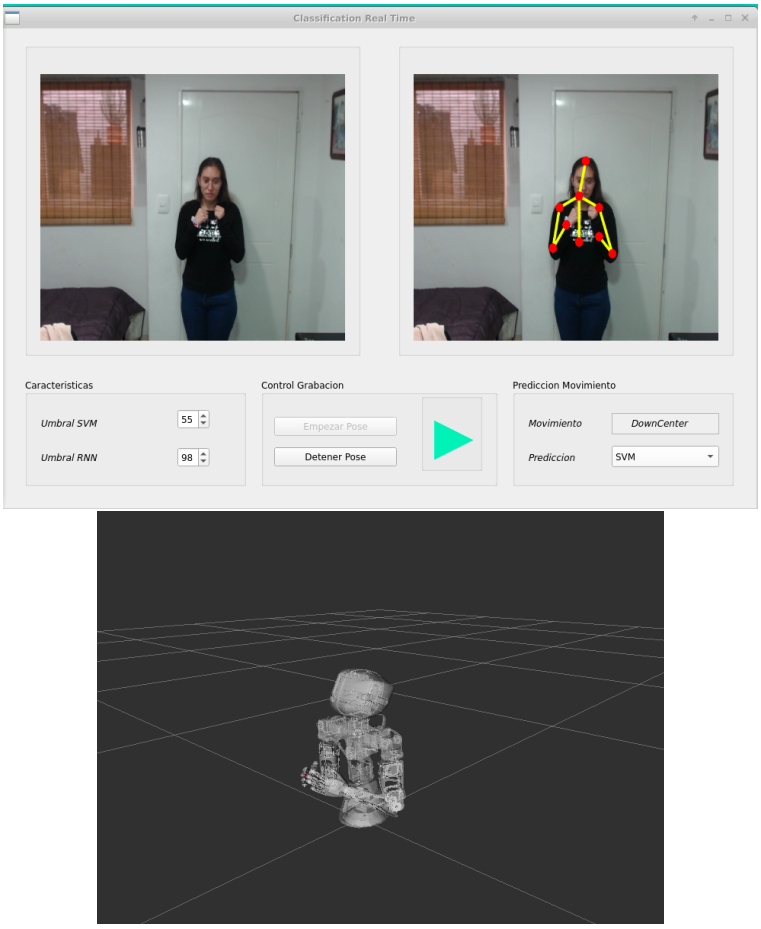


Figura 11.8. Aplicación y Simulación(Down-Center)

Otro movimiento Realizado fue Down-Up como se puede observar en la figura 11.9

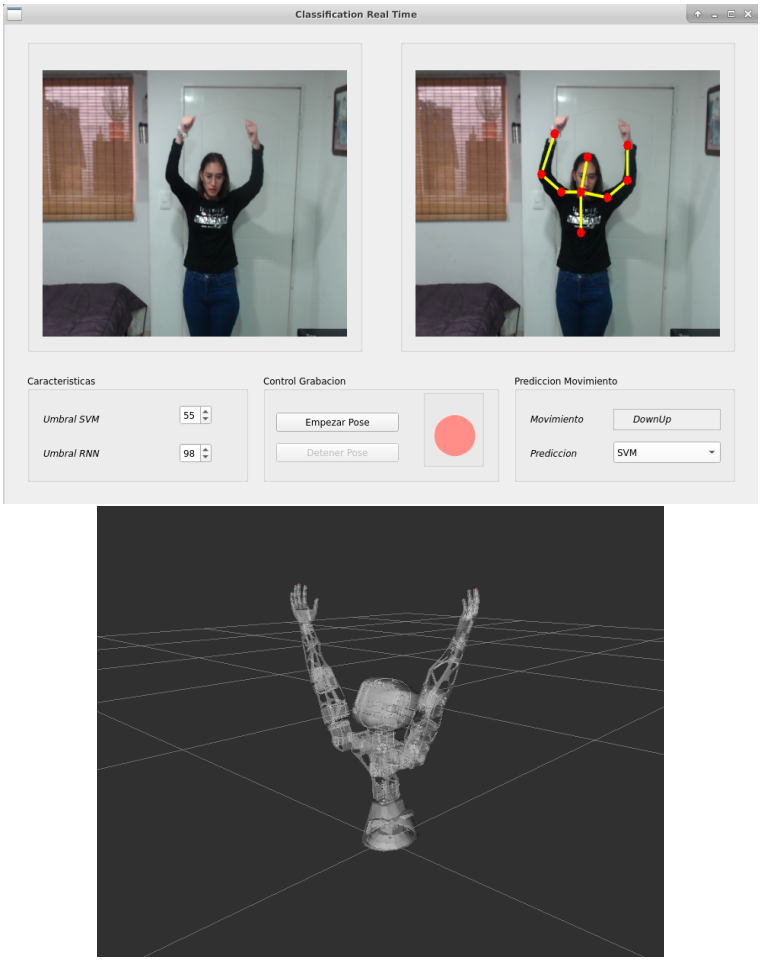


Figura 11.9. Aplicación y Simulación(Down-Up)

En el siguiente link se puede observar un video con el funcionamiento de la aplicación `https://www.youtube.com/watch?v=xFWh0xwRp_E&t=1s`.

Capítulo 12

Análisis de resultados

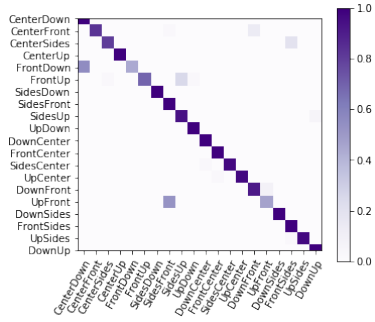
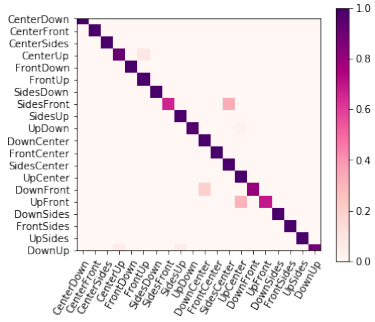
A continuación se presentan los análisis de los resultados obtenidos con la implementación de los algoritmos de clasificación y regresión, con el fin de observar el comportamiento de cada uno de los algoritmos, frente a las métricas de precisión y pérdida.

12.1. Algoritmos de clasificación

Con el fin de evaluar la precisión de los algoritmos de clasificación, se emplean matrices de confusión, que permiten observar las clases en el *eje y* y los movimientos que la red predice sobre el *eje x*. Los puntos en la diagonal principal representan el número de aciertos del algoritmo a la hora de clasificar, por otro lado, los puntos fuera de la diagonal principal, son aquellos que el algoritmo no clasifica correctamente. Es importante tener en cuenta la barra de color que indica la precisión, para cada uno de los movimientos clasificados en la matriz. La precisión de los algoritmos se mide de acuerdo a la ecuación 5.32, donde se tienen en cuenta las predicciones correctas y los falsos positivos.

12.1.1. Comparación del desempeño de los algoritmos de clasificación

Para la comparación de los algoritmos se desarrollo una tabla comparativa, de acuerdo al análisis realizado para cada uno de los algoritmos.

Comparación Algoritmos	
MANY TO ONE (RNN)	*SUPPORT VECTOR MACHINE*
	
La precisión total es de 91 %.	La precisión total es de 89 %.
El fondo de la figura presenta un tono morado claro, lo cual indica que al clasificar los movimientos, el algoritmo suele errar en la predicción aproximadamente un 2 % de las veces.	El fondo de la figura es de un tono rosa claro, indicando que al clasificar los movimientos, el algoritmo suele errar en la predicción aproximadamente un 10 % de las veces.
En la diagonal se puede observar que la mayoría de los movimientos se encuentran con una precisión del 100 % -76 % aproximadamente.	En la diagonal se puede observar que la mayoría de los movimientos se encuentran con una precisión del 100 % -62 % aproximadamente.

Se puede observar que 9 movimientos suelen confundirse con otros.	Se puede observar que 8 movimientos suelen confundirse con otros.
Uno de los movimientos confundidos tiene una precisión menor al 40 %.	Dos de los movimientos confundidos tiene una precisión menor al 55 %.
Se puede observar que los movimientos que suelen confundirse son Frente y Centro.	Se puede observar que los movimientos que suelen confundirse son Frente y Centro.

Cuadro 12.1. Tabla Comparación Algoritmos

De acuerdo a la tabla 12.1, se puede observar que la *Red Neuronal Recurrente - Many to one*, presenta buenos resultados, cuando es empleada para la predicción y/o clasificación de series de tiempo, por esto para la primera etapa del proyecto, la cual consta de la clasificación de movimientos, consigue los resultados esperados haciendo una correcta predicción de los movimientos.

Por otro lado la *máquina de soporte de vectores*, presenta resultados aceptables, cuando es empleada para la predicción y/o clasificación de series de tiempo, esta arquitectura falla más que la arquitectura anterior, debido a que el hiperplano de clasificación esta diseñado para menor cantidad de clases y además no se encuentra diseñada para realizar clasificaciones de series de tiempo, fue diseñada para clasificación sobre hiperplano.

Los tiempos de clasificación de cada algoritmo son:

- *Red neuronal recurrente - Many to One:* $600ms$
- *Máquina de soporte de vectores:* $400ms$

Los tiempos presentados por los algoritmos varían de acuerdo a las especificaciones del computador, en el que estos se estén ejecutando, en nuestro caso, el computador presenta las especificaciones consignadas en la tabla 12.2.

Especificaciones del computador	
COMPONENTE	REFERENCIA
Procesador	Intel Core7-8750H
GPU	NVIDIA GeForce GTX 1060 6GB VRAM GDDR5
Memoria RAM	8GB DDR4
Sistema Operativo	Ubuntu 19.10

Cuadro 12.2. Especificaciones del computador

12.2. Algoritmo de generación de trayectorias

Para este algoritmo se implementa el cálculo del error de posición final de cada uno de los motores (en radianes), empleando la ecuación de distancia euclidiana, donde se calcula la diferencia de los ángulos que debería tomar cada uno de los motores con respecto a su posición final. En la figura 10.4 que se presenta en el capítulo *Generación de trayectorias*, se registra el error obtenido para los cuatro motores que intervienen en el movimiento de un brazo para cada uno de los movimientos entrenados. Se observa que el mayor error se presenta en el motor brazo_z al ejecutarse el movimiento Down. Sin embargo, en la figura 10.5 que se presenta en el capítulo *Generación de trayectorias*, se observa que el error presentado no evita que la predicción que realiza la red se asemeje a la posición deseada.

En relación a los parámetros medidos, se obtiene una menor pérdida de entrenamiento cuando los parámetros barridos, que se muestran en el cuadro 10.2 del capítulo *Generación de trayectorias*, toman los valores de 28, 500 y 1024 respectivamente. Obteniendo pérdidas entre $9{,}841 \times 10^{-}4$ y $1,7 \times 10^{-}3$. Teniendo en cuenta estos bajos valores y las figuras 10.1 y 10.3 del capítulo *Generación de trayectorias*, donde se evidencia que la pérdida de entrenamiento y el error en el tiempo convergen a 0, se llega

a la conclusión que el tipo de arquitectura seleccionada y el valor de los parámetros con los que se llevan a cabo los entrenamientos son satisfactorios para cumplir el objetivo de generar trayectorias para ejecutar movimientos en un robot, ya que se encuentra en la capacidad a nivel general de ejecutar el movimiento solicitado, con un error de posición bajo, teniendo en cuenta que el error total, resultado de promediar la distancia euclidiana de todos los movimientos y motores, es de 0.035 radianes, que corresponde a 2 grados deg.

Conclusiones

Tener un *dataset* de movimientos de brazos con 2400 muestras, de 20 movimientos diferentes, grabados con 6 personas que tiene diferente apariencia física, permite a los algoritmos de clasificación arrojar predicciones del 91 % para la red neuronal recurrente *many to one* y del 89 % para la máquina de vectores de soporte.

Al tener un *dataset* de dos dimensiones,los algoritmos de clasificación tienden a confundir los movimientos *Front* y *Center*, esto ocurre por la posición espacial las muñecas y los codos, pues desde el punto de vista de la cámara, y la proyección bidimensional que ésta captura de la escena, estas articulaciones se encuentran en la misma ubicación, ocasionando que para la estimación de la pose los movimientos sean similares.

A partir de la implementación de una red neuronal con memoria a corto y largo plazo (LSTM) con el fin de reconocer movimientos desde puntos obtenidos a través de la estimación de pose en vídeos, se obtuvo una precisión del 93,2 % en los datos de prueba. Con este resultado se muestra que un modelo secuencial (redes neuronales con memoria) es apropiado para el reconocimiento de movimientos.

Para la implementación de algoritmos de clasificación (Red Many to one y SVM) es necesario que el *dataset* sea etiquetado, lo cual permite la agrupación de datos de acuerdo a las características que estos tienen en común.

Implementar una red neuronal con memoria a corto y largo plazo (LSTM) es adecuado a la hora de aprender patrones o secuencias, como se evidencia en el caso de la red *one to many*, donde se logró un

funcionamiento satisfactorio en los tres escenarios propuestos.

La máquina de estimación de postura empleada para la extracción de información del *dataset* RGB-D es muy útil pues permite a los algoritmos de clasificación tener como entrada la secuencia de posiciones de las articulaciones cuando se realiza un movimiento, esto se realiza por medio de una CMP que recolecta los puntos fotograma á fotograma durante la ejecución de un movimiento.

Para el aprendizaje de movimientos en un robot humanoide por medio de imitación se hizo necesario dividir el problema en la clasificación de movimientos y la regresión de secuencias que crea cada articulación del robot para ejecutar los movimientos.

Para una red LSTM de tipo *one to many* entrenada con el *dataset* del robot, se observa que con 512 neuronas el aprendizaje es insuficiente ya que al predecir las trayectorias en ocasiones genera movimientos inválidos, donde el robot genera posiciones imposibles para el cuerpo humano, por otro lado, con 1024 neuronas la predicción se hace de forma correcta. Funciona mejor con un *tamaño del lote* de 28 que de 12 y con 500 épocas da mejor desempeño que con 1000.

Emplear aprendizaje supervisado para el aprendizaje de movimientos es viable por medio de LSTM dados los resultados obtenidos.

Por medio de las matrices de confusión se evidencia que es más precisa la clasificación de los movimientos por medio de SVM, pues en la *many to one* se observa que la red confunde el movimiento center up por front up la mayoría de las veces.

El cálculo de error por distancia euclidiana en la posición final de los motores al predecir las trayectorias de los movimientos muestra que logra el *target* deseado ya que esta diferencia es insignificante en todos los movimientos.

Los entrenamientos de SVM tienen una mejor precisión al entrenar con un *tamaño del lote* mayor, gamma de 0.1 y C de 1, por otro lado el *many to one* funciona mejor con un *tasa de aprendizaje* de 0.001, *tamaño del lote* de 15, 256 neuronas y 500 épocas, funciona mejor con 256 neuronas que

con 128 o 512.

En la aplicación se puede observar que la SVM, es mas sensible a cambios, en especial cuando se trata de otra persona, esto se debe a que el límite de decisión restringe el límite entre las clases y al tener demasiadas clases, las muestras pueden pasar del hiperplano y ser clasificadas en otra clase.

Trabajos futuros

Este proyecto de grado puede ser continuado a través de desarrollos llevados a espacios de rehabilitación física, realizando un nuevo dataset, compuesto por movimientos que le permitan al paciente mejorar la lesión que tenga en sus extremidades superiores. Para la construcción de este dataset, se necesita el acompañamiento de un fisioterapeuta, que a partir de su criterio médico guíe los movimientos que deben ser incluidos en el dataset, en el cuál se puede contar además con la participación de adultos mayores. Se deben entrenar los algoritmos de clasificación y regresión para darle la capacidad al robot de replicar el movimiento y guiar la terapia física, mejorando la calidad de vida de los pacientes de edad avanzada mediante una atención personalizada y guiada completamente por el robot.

Se puede realizar estudios también donde se amplíen las partes del cuerpo involucradas, como extremidades inferiores, espalda y cadera, mejorando la prestación de servicios dentro de los centros de rehabilitación física, para este es necesario buscar un robot que posea los mismos grados de libertad que tiene una persona, como el robot Poppy Humanoid u otro semejante, permitiendo que la replica del movimiento tenga mayor similitud.

Para evaluar si el adulto mayor realiza la terapia física correctamente se puede realizar monitoreo a través de la aplicación en tiempo real, donde por medio del algoritmo de clasificación se identifique si el movimiento realizado por el adulto mayor corresponde al correcto.

Productos del proyecto

Como productos obtenidos del proyecto se presenta inicialmente el registro del software de clasificación *LSTM and SVM based Movement Recognition Software*, que se encuentra vinculado al proyecto FODEIN *Aprendizaje supervisado para imitación de movimientos de un robot humanoide en el marco de la alianza SINFONIA*. Este registro se encuentra en el anexo E.1, donde se encuentra la aplicación que permite identificar en tiempo real el movimiento realizado por una persona, a través de algoritmos de clasificación *Red neuronal Recurrente y Máquina de vectores de soporte*.

Un segundo producto corresponde a la presentación del artículo *"Vision based upper limbs movement recognition using LSTM neural network"* en la conferencia *AETA 2019 - Recent Advances in Electrical Engineering and Related Sciences: Theory and Application*, en noviembre del año 2019. Además está en proceso de publicación en la revista *Lecture Notes in Electrical Engineering*, de la editorial *Springer*, clasificada por *Scimago Journal* como Q3. El artículo presenta la teoría y la implementación de una red neuronal recurrente empleando la arquitectura LSTM many to one, esto con el fin de reconocer los movimientos realizados en los videos, los puntos que se tuvieron en cuenta para la ejecución de los movimientos y su identificación corresponden a la posición de seis partes del cuerpo: hombros, codos y muñecas, estas se extraen con una máquina de pose convolucional pre-entrenada, la red implementada consta de 128 celdas LSTM y presentó un 92,5 % de precisión en los datos de prueba y un tiempo de ejecución de alrededor de

6,64ms.

Otros de los productos resultantes del proyecto son los *dataset* construidos. El primero de ellos es el de movimientos humanos, el cual contiene 2400 videos RGB-D de desplazamientos de las extremidades superiores, que constan de la combinación de los siguientes puntos como partida y objetivo: *brazos extendidos hacia arriba*, *brazos extendidos hacia abajo*, *brazos extendidos hacia el frente*, *brazos extendidos hacia los lados* y *brazos recogidos hacia el pecho*, este *dataset* se construyó con 6 personas de diferentes contexturas físicas. El segundo *dataset* se construyó por medio de la herramienta de simulación *R-VIZ*, permitiendo obtener la trayectoria que realiza el robot desde un punto aleatorio hasta una posición final. Las posiciones finales corresponden a las anteriormente mencionadas en el *dataset* de movimientos humanos.

Adicional, se hace un aporte a las herramientas del laboratorio de Robótica de la Universidad Santo Tomás por medio del robot humanoide Poppy Torso, que al igual que los productos anteriores, puede ser de ayuda para futuros desarrollos de estudiantes y docentes que requieran el uso del robot humanoide, así como el estudio de redes neuronales recurrentes tanto para clasificación como predicción de trayectorias ó SVM para clasificación.

Glosario

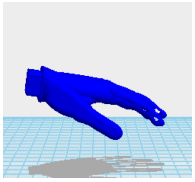
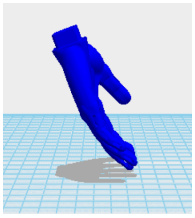
Se presenta un glosario, con las palabras con mayor relevancia dentro del proyecto.

- **Error Correcting Output Codes *(ECOC)*:** Conjunto de métodos diseñados para problemas de clasificación multiclase; este tiene como tarea la elección de una etiqueta; algunos de estos algoritmos son :el árbol de decisión, las bahías ingenuas y la red neuronal [33].
- **Distancia Euclidiana**: Distancia entre dos puntos de un espacio euclídeo, deducida empleando el teorema de Pitágoras.
- **Keras**: API de alto nivel de Tensorflow para construir y entrenar modelos de aprendizaje profundo. Se utiliza para la creación rápida de prototipos, la investigación de vanguardia (estado-del-arte) y en producción, con tres ventajas clave: Amigable con el usuario, lo cual lo hace fácil de entender, además de ser modular y configurable.
- **Máquina de vectores de soporte*(SMV)***: Algoritmo de aprendizaje supervisado para la clasificación binaria, por medio de un criterio de margen maximizado. Este es empleado para la solución de problemas multiclases[2][32].
- **Long short-term memory *(LSTM)***: Arquitectura artificial de red neuronal recurrente (RNN), la cual tiene conexiones de retroalimentación entre las celdas permitiendo el procesamiento de puntos de datos individuales (como imágenes) o secuencias completas de datos (como voz o vídeo).

- **MPII:** Dataset de 25000 imágenes con muestras de 4000 personas y 410 actividades humanas; este es empleado para la estimación y evaluación de una posición humana.
- **Red Neuronal**: Interconexión de procesadores, los cuales reciben el nombre de neuronas, cada una de estas neuronas, produce una secuencia de activación, la cual puede o no activar otra neurona que se encuentre conectada a esta, permitiendo así que al final se encuentre un valor real.[1]
- **Red Neuronal Recurrente *(RRN)***: Red neuronal que permite la predicción de una acción de acuerdo a un entorno, esto por medio de una arquitectura la cual posee de una memoria, en esta se va a almacenar la salida de la capa anterior, lo cual permite trabajar por medio de series temporales.
- **Robot Humanoide**: Robot que adquiere por su forma o por sus propiedades la capacidad para interactuar con humanos, asemejándose en su comportamiento o en su aspecto al de un humano.
- **Tensorflow**: Tensorflow es una plataforma end to end diseñada para realizar algoritmos de aprendizaje de máquina, es de fuente abierta, posee herramientas y librerías flexibles que permiten a desarrolladores construir aplicaciones de *Machine Learning*.
- **Vector Placeholder**: M 0arcador de posición, que ayuda a identificar a cuál grupo de funciones se hace referencia.

Apéndice A

Construcción Poppy

Vista partes en 3D robot humanoide	
PARTE	VISTA
Mano Horizontalmente	
Mano Verticalmente	

Cuadro A.1. Tabla Impresión partes Poppy

Apéndice B

Entrenamientos Many to one

Resultados Red neuronal recurrente (Many to one- Dataset 2 dimensiones)	
PARÁMETROS	RESULTADOS
Entrenamiento 1	
Tasa de Aprendizaje = 0,001 Tamaño del lote = 10 Neuronas = 128 Pasos de entrenamiento = 300	Precisión = 0,932
Entrenamiento 2	
Tasa de Aprendizaje = 0,0001 Tamaño del lote = 30 Neuronas = 512 Pasos de entrenamiento = 250	Precisión = 0,8889
Entrenamiento 3	
Tasa de Aprendizaje = 0,001 Tamaño del lote = 15 Neuronas = 256 Pasos de entrenamiento = 500	Precisión = 0,9792
Entrenamiento 4	
Tasa de Aprendizaje = 0,01 Tamaño del lote = 50 Neuronas = 128 Pasos de entrenamiento = 300	Precisión = 0,8708

Entrenamiento 5	
Tasa de Aprendizaje = 0,0001 Tamaño del lote = 20 Neuronas = 256 Pasos de entrenamiento = 600	Precisión = 0,9111

Cuadro B.1. Tabla Entrenamientos Many to One-Dataset 2 dimensiones

Apéndice C

Entrenamientos SVM

Resultados SVM Dataset	
PARÁMETROS	RESULTADOS
Entrenamiento 1	
Kernel = *Poly* Grado = 5 Tasa de aprendizaje = 0,01 C = 1 Tamaño del lote = 100	Precisión de Entrenamiento = 0,9375
Entrenamiento 2	
Kernel = *Linear* Tasa de aprendizaje = 0,1 C = 0,8 Tamaño del lote = 100	Precisión de Entrenamiento = 0,9681
Entrenamiento 3	
Kernel = *RBF* Tasa de aprendizaje = 0,1 C = 0,8 Tamaño del lote = 100	Precisión de Entrenamiento = 0,05

Entrenamiento 4	
Kernel = *Linear* Tasa de aprendizaje = 0,1 C = 1 Tamaño del lote = 30	Precisión de Entrenamiento = 0,881
Entrenamiento 5	
Kernel = *Poly* Grado = 6 Tasa de aprendizaje = 0,1 C = 0,5 Tamaño del lote = 250	Precisión de Entrenamiento = 0,9139

Cuadro C.1. Tabla Entrenamientos SVM Dataset 2 dimensiones

Apéndice D

Entrenamientos One to many

Resultados Red neuronal recurrente *One to Many*	
PARAMETROS	RESULTADOS
Entrenamiento 1	
Tamaño del lote = 12 Pasos de entrenamiento = 1000 Neuronas = 512 Movimiento = *Down*	Pérdida de entrenamiento = 0,0036 Pérdida de validación = $4{,}4681x10^{-4}$
Entrenamiento 2	
Tamaño del lote = 28 Pasos de entrenamiento = 500 Neuronas = 1024 Movimiento = *Down*	Pérdida de entrenamiento = 0,0021 Pérdida de validación = $2{,}6521x10^{-3}$

Entrenamiento 3	
Tamaño del lote = 12 Pasos de entrenamiento = 1000 Neuronas = 512 Movimiento = $Front$	Pérdida de entrenamiento = 0,0020 Pérdida de validación = $2{,}5984x10^{-4}$
Entrenamiento 4	
Tamaño del lote = 28 Pasos de entrenamiento = 500 Neuronas = 1024 Movimiento = $Front$	Pérdida de entrenamiento = 0,0017 Pérdida de validación = $6{,}0773x10^{-4}$
Entrenamiento 5	
Tamaño del lote = 12 Pasos de entrenamiento = 1000 Neuronas = 512 Movimiento = Up	Pérdida de entrenamiento = 0,0025 Pérdida de validación = $4{,}8514x10^{-4}$
Entrenamiento 6	
Tamaño del lote = 28 Pasos de entrenamiento = 500 Neuronas = 1024 Movimiento = Up	Pérdida de entrenamiento = 0,0015 Pérdida de validación = $5{,}9784x10^{-4}$
Entrenamiento 7	
Tamaño del lote = 12 Pasos de entrenamiento = 1000 Neuronas = 512 Movimiento = $Sides$	Pérdida de entrenamiento = $5{,}4955x10^{-4}$ Pérdida de validación = $3{,}0287x10^{-4}$

Entrenamiento 8	
Tamaño del lote = 28 Pasos de entrenamiento = 500 Neuronas = 1024 Movimiento = *Sides*	Pérdida de entrenamiento = $9{,}841x10^{-4}$ Pérdida de validación = $2{,}5185x10^{-4}$
Entrenamiento 9	
Batch size = 12 Pasos de entrenamiento = 1000 Neuronas = 512 Movimiento = *Center*	Pérdida de entrenamiento = 0,0066 Pérdida de validación = $7{,}4146x10^{-4}$
Entrenamiento 10	
Tamaño del lote = 28 Pasos de entrenamiento = 500 Neuronas = 1024 Movimiento = *Center*	Pérdida de entrenamiento = 0,0034 Pérdida de validación = 0,0011

Cuadro D.1. Tabla Entrenamientos One to Many

Apéndice E

Descripción Software

Título de la Obra: LSTM and SVM based Movement Recognition Software (Software de Reconocimiento de Movimientos basado en LSTM y SVM)

Autores	
Nombre	Documento de identidad
Edgar Camilo Camacho Poveda	CC 1 054 680 989
Andrea Catalina Rey Ramírez	CC 1 013 678 687
Alison Gissell Ruiz Ruiz	CC 1 012 454 646
Carolina Higuera Arias	CC 1 049 630 929

Cuadro E.1. Autores del registro software

Patrocinado por: Universidad Santo Tomás - Nit: 860.012.357-6

Lenguaje de Programación: Python 3

Descripción General: Este programa de computador permite determinar el movimiento realizado por las extremidades superiores de una persona frente a una cámara conectada al equipo, a través de algoritmos de aprendizaje de máquina (Long Short-Term Memory Neural

Networks y Support Vector Machines), pre entrenados para detectar las transiciones entre las siguientes posiciones de los brazos (20 en total):

- Extendidos hacia arriba
- Extendidos hacia abajo
- Extendidos hacia el frente
- Extendidos hacia los lados
- Recogidos contra el pecho

La información correspondiente a la detección realizada, puede ser enviada a través de tuberías o memoria compartida a otros procesos, para así utilizarla para tomar la acción necesaria.

El software dispone además de una interfaz gráfica para seleccionar el método de clasificación a seleccionar, así como umbrales de detección. Así mismo, controla el inicio y la finalización de la detección, y visualiza la transición detectada.

Requisitos de operación:

- GPU Nvidia Compatible con Cuda 10.0 o superior.
- Ubuntu 16.04 o superior
- Python 3.6 o superior.
- Dependencias listadas en el archivo environment.yml

Ejecución: Para la instalación de las dependencias y la ejecución del programa, siga las instrucciones descritas en el archivo README.md (README.pdf para visualizar imágenes).

Bibliografía

[1] J.Schmidhuber, "Deep learning in neural networks: An overview,"Neural Networks, vol. 26, pp. 1-6, 2012.

[2] Y. Ma and G. Guo, "Multi-Class Support Vector Machine,"Springer,pp.23-48, 2014.

[3] (Jun,). Envejecimiento demográfico. Colombia 1951-2020 dinámica demográfica y estructruras poblacionales. Available: https://www.minsalud.gov.co/sites/rid/Lists/BibliotecaDigital/RIDE/DE/PS/Envejecimiento-demografico-Colombia-1951-2020.pdf&sa=D&ust=1535994535894000&usg= AFQjCNFFif1BUW1zdkBzyFc1uRSn4XQxZA.

[4] Velarde, P and Perugachi, Erika and Vintimilla, B and Romero, Dennis, "Seguimiento y Análisis del Movimiento de las Extremidades Superiores Aplicado a la Rehabilitación Física de un Paciente Usando Técnicas de Visión Artificial,Revista Tecnológica ESPOL – RTE, Vol. 28, N. 1, 1-7, Agosto 2015.

[5] Robotics 2020 Multi-Annual Roadmap ,SPARC,2 Junio 2015.

[6] CONGRESO DE LA REPÚBLICA(2018,Junio 11), LEY 1251 DE 2008

[7] A. Ghazali, J. Ham, E. Barakova and P. Markopoulos, "The influence of social cues in persuasive social robots on psychological reactance and compliance", Computers in Human Behavior, vol. 87, pp. 58-65, 2018. Available: 10.1016/j.chb.2018.05.016.

[8] S. Schaal, Ïs imitation learning the route to humanoid robots?", 1999. Available: 10.1016/s1364-6613(99)01327-3.

[9] M. Morales, L. Tapia, G. Sanchez Ante and S. Hutchinson, Algorithmic foundations of robotics XIII. Cham: Springer, 2020.

[10] W. Ohata and J. Tani, "Investigation of multimodal and agential interactions in human-robot imitation, based on frameworks of predictive coding and active inference", 2020. arXiv preprint arXiv:2002.01632, 2

[11] J. Yu, H. Yao, G. Zuo and S. An, "LSTM Learn Policy from Dynamical System of Demonstration Motions for Robot Imitation Learning", 2019 IEEE 9th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER), 2019. Available: 10.1109/cyber46603.2019.9066716.

[12] A. Wu, A. Piergiovanni and M. Ryoo, "Model-Based Robot Imitation with Future Image Similarity", International Journal of Computer Vision, vol. 128, no. 5, pp. 1360-1374, 2019. Available: 10.1007/s11263-019-01238-5.

[13] S. Wen, Y. Liu, L. Zheng, F. Sun and B. Fang, "Pose Analysis of Humanoid Robot Imitation Process Based on Improved MLP Network", 2019 WRC Symposium on Advanced Robotics and Automation (WRC SARA), 2019. Available: 10.1109/wrc-sara.2019.8931970..

[14] M. Barrio Andrés, M. Froomkin and A. Aransay Alejandre, Derecho de los robots. Madrid: Wolters Kluwer, 2018.

[15] Poppy Project. Available: https://www.poppy-project.org/en/.

[16] K. Hirai, M. Hirose, Y. Haikawa and T. Takenaka, "The development of Honda humanoid robot", Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146). Available: 10.1109/robot.1998.677288.

[17] N. Ratliff, J. A. Bagnell, and S. S. Srinivasa, “Imitation learning for locomotion and manipulation,” in 2007 7th IEEE-RAS International Conference on Humanoid Robots. IEEE, 2007, pp. 392–397.

[18] P. Sermanet, C. Lynch, J. Hsu and S. Levine, ”Time-Contrastive Networks: Self-Supervised Learning from Multi-view Observation”, 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2017. Available: 10.1109/cvprw.2017.69.

[19] A. Hussein, E. Elyan, M. Gaber and C. Jayne,Deep imitation learning for 3D navigation tasks,Neural Comput —& Applic, vol. 29, pp. 389-404,Apr. 2018.

[20] B.D. Argall, S. Chernova, M. Veloso and B. Browning, A survey of robot learning from demonstration, Robotics and Autonomous Systems, vol. 57, pp. 469-483, 2009.

[21] L.P. Kaelbling, M. Littman and A. Moore, Reinforcement Learning: A Survey, Journal of ArtiCial Intelligence Research 4, pp. 237-285

[22] W.Maass, T.Natschläger and H.Markram, Real-Time Computing Without Stable States: A New Framework for Neural Computation Based on Perturbations, Neural Computation, vol. 14, pp. 2531-2560, Nov 1,.2002.

[23] T. Zhang, Z. Mccarthy, O. Jow, D. Lee, X. Chen, K. Goldberg and P. Abbeel,”Deep Imitation Learning for Complex Manipulation Tasks from Virtual Reality Teleoperation”., 2016 IEEE International Conference on Robotics and Automation (ICRA). Oct 12 2017.

[24] S. Levine, C. Finn, T. Darrell and P. Abbeel,End-to-End Training of Deep Visuomotor Policies, Journal of Machine Learning Research 17, pp. 1-40, Apr 2,. 2015.

[25] T. Yu, C. Finn, A. Xie, S. Dasari, T. Zhang, P. Abbeel and S. Levine,One-Shot Imitation from Observing Humans via Domain-

Adaptive Meta-Learning, University of California, Berkeley, Feb 5,. 2018.

[26] S. Gallant, Neural network learning and expert systems. Cambridge, Mass: The MIT, 1995.

[27] V. Sharma , S. Rai and A. Dev, A Comprehensive Study of Artificial Neural Networks,International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 2, 2012.

[28] N. Sharma, V. Jain and A. Mishra, .ªn Analysis Of Convolutional Neural Networks For Image Classification", Procedia Computer Science, vol. 132, pp. 377-384, 2018. Available: 10.1016/j.procs.2018.05.198.

[29] W. Ko, B. Tseng and H. Lee, Recurrent Neural Network based language modeling with controllable external Memory,"2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), New Orleans, LA, 2017, pp. 5705-5709, doi: 10.1109/ICASSP.2017.7953249.

[30] Yun Liu, Wendong Xiao, Han-Chieh Chao and Pony Chu, "Deep Convolutional and LSTM Recurrent Neural Networks for Multimodal Wearable Activity Recognition,"2016 Advances on Data Transmission and Analysis for Wearable Sensors Systems, do:10.3390/s16010115.

[31] S. Ravichandiran, Hands-On Deep Learning Algorithms with Python. Packt Publishing, 2019.

[32] V. Cherkassky and F. Mulier, Learning from data. Hoboken (N.J.): John Wiley, 2007.

[33] A. Passerini, M. Pontil and P. Frasconi, "New results on error correcting output codes of kernel machines,ïn IEEE Transactions on Neural Networks, vol. 15, no. 1, pp. 45-54, Jan. 2004.

[34] Zhe Cao and Gines Hidalgo and Tomas Simon and Shih-En Wei and Yaser Sheikh,Open Pose: realtime multi-person 2D pose estimation using Part Affinity Fields,arXiv preprint arXiv:1812.08008,2018

[35] Cortes, C., Vapnik, V. Support-vector networks. Mach Learn 20, 273–297 (1995). https://doi.org/10.1007/BF00994018