



Visión Computacional Aplicado a Drones para Actividades Recreativas

Sistemas Basados en Redes Neuronales

**Sánchez Patiño Natalia
Rosas Otero Mario**

Profesor: Dr. David Tinoco Varela

Licenciatura en Tecnología

Facultad de Estudios Superiores Cuautitlán
Universidad Nacional Autónoma de México

07 - 02 - 2021

-Proyecto para Sistemas Basados en Redes Neuronales: Aplicación de
modelos de reconocimiento a un dron como método de interacción.-

Resumen

Visión Computacional Aplicado a Drones para Actividades Recreativas En este proyecto se realiza la construcción de un sistema capaz de integrar visión computacional con el manejo, control e interacción con un dron, de forma que se tenga un control semi-autónomo y de interacción sin contacto físico con el hardware, esto principalmente con fines recreativos y educativos que permitan conocer el funcionamiento de los movimientos de un dron y la forma en que se pueden utilizar las señales e imágenes mandas por este al sistema interpretándolas y realizando acciones acordes en consecuencia. El correcto funcionamiento de las versiones construidas requiere de elementos tanto de hardware como de software, e involucra mantener distintos archivos, de ejecución de código, y bases de datos, así como instalación de distintas bibliotecas y dependencias. Dichos archivos en conjunto permiten tener la información necesaria para interactuar con el usuario con el objetivo de facilitar la comunicación y precisión de movimientos. Con esto realizamos la propuesta a una pequeña escala de procesar las imágenes obtenidas por el dron durante su vuelo, utilizando distintas herramientas de código abierto para poder integrar un programa funcional, permitiendo implementar la comunicación entre las distintas partes que integran al sistema. Esto de manera que pueda utilizarse de forma lúdica y cotidiana cuando se desee poder experimentar con este tipo de dron, en nuestro caso un DJI Tello. Todo esto es posible ejecutarlo de forma off-line. Dentro de las posibles extensiones del proyecto se plantea el funcionamiento del sistema dentro de una tarjeta de desarrollo como Raspberry Pi, a fin de dar portabilidad al sistema e integrarlo a otros dispositivos, con posibilidad de extensión de características. Es posible obtener el código total del proyecto en:
<https://github.com/NM-Labs/DronFollowMe>

Índice

1	Introducción	1
1.1	Motivación	1
1.2	Objetivos	2
1.3	Solución Propuesta	3
2	Marco teórico	4
2.1	Redes neuronales	4
2.1.1	Redes Neuronales Convolucionales	6
2.2	Visión Computacional	7
2.3	Drones	9
3	Elementos Utilizados y Proceso de Construcción	10
3.1	Hardware	10
3.2	Software	11
3.3	Instalaciones	12
3.4	Funcionamiento del Sistema	12
3.4.1	Parte A: Clasificación de letras	13
3.4.2	Parte B: Escritura en el Aire	16
3.4.3	Parte C: OpenPose	16
3.4.4	Parte D: Drone	19
4	Resultados obtenidos	22
4.1	Uso de redes neuronales	22
4.2	Implementación de OpenPose	23
4.3	Sistema de control del Drone	23

4.4	Practicidad de las bibliotecas y softwares requeridos	24
4.5	Uso integrado de hardware y software	24
5	Conclusiones	27
5.1	Posibles extensiones del proyecto	28
5.2	Comentarios Finales	30
	Referencias	31

Introducción

El proyecto desarrollado plantea la construcción de un sistema con dos modos de funcionamiento, el primero capaz de reconocer letras escritas en el aire, frente a la cámara montada en un drone programable, y actuar en consecuencia conforme a rutas específicas para cada letra, entre otros movimientos. Dichas características sirven para incrementar la interacción que existe entre el usuario y el quadracóptero. La segunda modalidad consiste en el uso de un modelo pre-entrenado para reconocimiento de pose humana, con el cual se plantea la posibilidad de dar indicaciones de control mediante posturas corporales. Esto puede implicar el poder desarrollar más actividades haciendo uso de este tipo de herramientas, de acuerdo también a los deseos e intereses del usuario. Y con ello ser capaz de realizar diferentes series de movimientos, sin necesidad de dirigirlo mediante un control remoto, de una forma mas natural.

1.1 | Motivación

Dentro del ámbito escolar, comercial e industrial, el término Inteligencia Artificial se ha vuelto una constante, relacionándose en la mayoría de las ocasiones con tecnología de punta. Sin embargo también han surgido posturas que consideran potencialmente peligroso el desarrollo de esta área, debido a la gestión de conceptos como la posibilidad de alcanzar la singularidad tecnológica. Este punto de singularidad se identifica como el momento donde los robots, y las computadoras serán capaces de desarrollar mejoras a si mismos, y con ello establecer dominio sobre la humanidad, sobrepasando la capacidad intelectual de sus creadores originales. Sin embargo, los preceptos bajo los que aun trabaja el área, están muy alejados de alcanzar dicho punto, principalmente basan su funcionamiento en herramientas matemáticas, para aproximar soluciones a la descripción numérica de los problemas a los que se trata de dar solución aplicando conceptos

de Inteligencia Artificial. Una de las herramientas dentro de la Inteligencia Artificial, que se ha popularizado en los últimos años son las Redes Neuronales, y sus distintas variantes, esto debido a su gran capacidad de aproximación, y alta efectividad en el campo práctico. Una variante de las redes neuronales, son las Redes Neuronales Convoluciones, un diseño adaptado a un tipo de información en específico: las imágenes. Debido a ello, este tipo de redes son ampliamente utilizadas en el campo de visión computacional, para resolver problemas de principalmente de clasificación o de generación. Es por ello que en este trabajo nos hemos enfocado un poco más a su estudio mediante la puesta en práctica de la misma, y mostrando algunas de las propuestas a las que se han aplicado este tipo de redes. Además, de encontrar puntos de convergencia entre elementos de la cotidianidad y el uso de este tipo de tecnologías, resultado del desarrollo de este trabajo se pretende desarrollar experimentación vinculando herramientas de inteligencia artificial con dispositivos de hardware: vehículos aéreos de uso recreativo.

1.2 | Objetivos

■ Objetivo general:

Construir de forma práctica un sistema experimental de reconocimiento y clasificación utilizando técnicas y conceptos vistos en clase, como lo son las redes neuronales y extendiendo dicho concepto hacia sus variantes aplicadas a la visión computacional. Logrando que el sistema sea capaz de llevar un correcto funcionamiento dentro del hardware, en este caso un drone, mediante indicaciones de control dadas mediante el procesamiento de información realizado por técnicas de Inteligencia Artificial, obteniendo así herramientas interactivas entre usuario y sistema.

■ Objetivos Particulares:

- Realizar una revisión teórica acerca del concepto de redes neuronales y sus variantes.
- Buscar la aplicabilidad de las redes neuronales como estructura de clasificación de datos visuales, y llevar el resultado hacia una acción de control.
- Elaborar un diseño teórico y práctico para llevar a cabo la implementación del sistema mediante el uso del lenguaje de programación Python.
- Complementar y desarrollar habilidades de programación en el lenguaje Python como herramienta para la creación y ejecución del sistema.

- Verificar que el sistema cuente con un funcionamiento práctico, así como identificar e implementar el sistema utilizando herramientas abiertas a todo público.
- Comprobar de manera aplicada, conceptos vistos en clase y conjuntándolo con conceptos encontrados durante la búsqueda de información relacionada a temas de la clase.
- Buscar nuevos métodos de interacción con dispositivos de hardware ejemplificando métodos que pueden ser utilizados en situaciones riesgosas o peligrosas para un ser humano.

1.3 | Solución Propuesta

La propuesta de implementación de este proyecto se decidió realizarla en el lenguaje de programación Python, dadas la herramientas ya existentes dentro del mismo, para poder establecer un canal de comunicación con el drone DJI Tello, debido a que existen bibliotecas que permiten programar comandos de control para este drone en específico. Además, de que tiene bibliotecas con implementaciones estables para desarrollo de Deep Learning y existe la posibilidad mediante el uso de una API de obtener datos de un modelo entrenado para la estimación de pose humana. Se propone, realizar un entrenamiento utilizando redes neuronales convolucionales para realizar el reconocimiento de trazos hechos por el usuario en el aire con un objeto que funciona como identificador de color, tratando de clasificar dichos trazos como una letra, y con ello establecer una señal de movimiento al drone. El principal escenario planteado consiste en la ejecución e interacción mediante movimientos con el drone volando frente al usuario. Pero también se contempla poder interactuar únicamente mediante el uso de una computadora para realizar dicho reconocimiento. Durante la búsqueda se seleccionaron herramientas de código abierto lo cual las hace accesibles para cualquier persona, si bien, estas son de libre acceso, teniendo la posibilidad de inspirar hacia otros proyectos utilizando este tipo de técnicas. Sin embargo, este tipo de modelos computacionales aún implican requerimientos de capacidad computacional un poco altos, lo que hace necesario el tener un dispositivo con capacidad de computo suficiente, es decir una computadora como intermediario para realizar el procesamiento de la información. Más adelante se esperaría poder contar con herramientas para el mismo propósito con requerimientos computacionales más flexibles y que permitan realizar el procesamiento de la información en los propios dispositivos móviles donde se captura la información, haciendo el proceso más sencillo y sin necesidad de dispositivos intermediarios.

Marco teórico

2.1 | Redes neuronales

Las redes neuronales son uno de los paradigmas de programación más hermosos jamás inventados. En el enfoque convencional de la programación, le decimos a la computadora qué hacer, dividiendo los grandes problemas en muchas tareas pequeñas y definidas con precisión que la computadora puede realizar fácilmente. Por el contrario, con una red neuronal no le decimos a la computadora cómo resolver nuestro problema. En cambio, aprende de los datos de observación y descubre su propia solución al problema en cuestión [Nielsen \(2015\)](#).

El reciente resurgimiento de las redes neuronales, la revolución del aprendizaje profundo, es cortesía de la industria de los juegos de computadora. Las imágenes complejas y el ritmo rápido de los videojuegos actuales requieren un hardware que pueda mantenerse al día, y el resultado ha sido la unidad de procesamiento de gráficos (GPU), que incluye miles de núcleos de procesamiento relativamente simples en un solo chip. Los investigadores no tardaron en darse cuenta de que la arquitectura de una GPU es notablemente parecida a la de una red neuronal [Hardesty \(2017\)](#).

Estas redes neuronales son redes de múltiples capas de neuronas que se usan para clasificar cosas, hacer predicciones, máscaras, etc. A continuación (figura 2.1) se muestra el diagrama de una red neuronal simple con cinco entradas, 5 salidas y dos capas ocultas de neuronas.

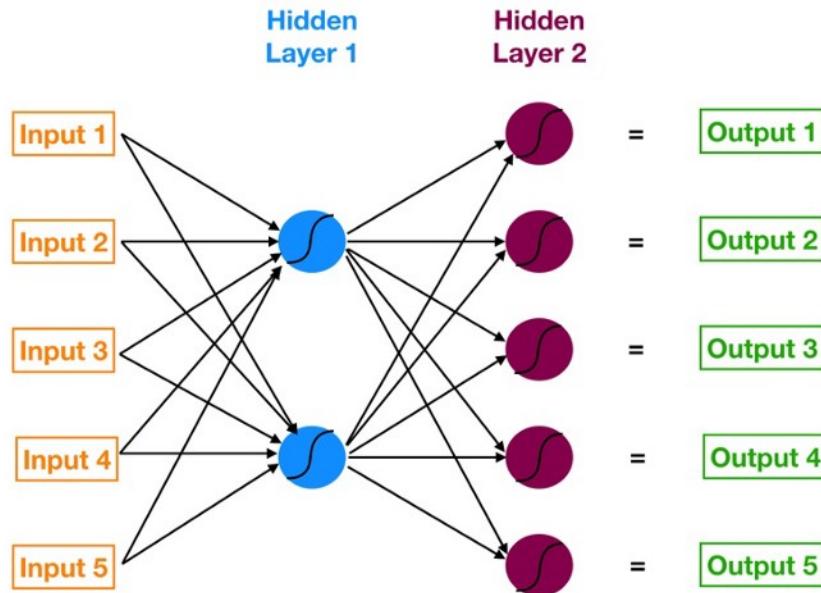


Figura 2.1: Red neuronal con dos capas ocultas.

Con las redes neuronales es posible crear un modelo a gusto del diseñador y conforme a los requerimientos del problema a resolver con el objetivo de hacer una buena predicción acerca del comportamiento del problema estudiado. De forma general con las redes neuronales se tiene un conjunto de entradas y un conjunto de valores objetivo, y se intentan obtener predicciones que coincidan lo más posible con esos valores objetivo. Las redes neuronales más complejas son solo modelos con más capas ocultas y eso significa más neuronas y más conexiones entre neuronas, aunque no necesariamente más neuronas o más capas aseguran una mejor aproximación a la solución del problema. Y a esta red se le entrena mediante un proceso iterativo para ajustar valores que parametrizan los datos que entran a la red intentando encontrar patrones entre los datos, es decir relaciones ocultas entre los mismos que pueden no ser tan fáciles de identificar por métodos tradicionales y con ello converger a una solución.

En una red neuronal, cambiar el peso de cualquier conexión (o el sesgo de una neurona) tiene un efecto de reverberación en todas las demás neuronas y sus activaciones en las capas posteriores, eso es porque cada neurona en una red neuronal modela características individuales del conjunto de datos.

2.1.1 | Redes Neuronales Convolucionales

Las redes neuronales convolucionales son un algoritmo de aprendizaje profundo que es una rama de la inteligencia artificial que intenta lidiar con redes neuronales grandes, lo que aumenta su complejidad, y causa algunos inconvenientes, si solo se implementan con el método tradicional. En específico las redes neuronales convolucionales están construidas principalmente para procesamiento de imágenes intentando encontrar patrones en los datos de forma espacial, relacionando un píxel con sus píxeles vecinos, con ello asignar importancia (pesos y sesgos aprendibles) que serán una representación cada vez mas abstracta del contenido en la imagen. Se intenta reconocer varios aspectos / objetos de la imagen y poder diferenciar uno de otro, discriminando el segundo plano en la imagen, y identificando un objeto de interés en primer plano. Una de las ventajas que tiene el uso de este tipo de redes, es que cuando se procesa una imagen a través de un perceptrón multicapa, el numero de parámetros (pesos y sesgos) requeridos es elevado, lo que hace el proceso de entrenamiento y verificación procesos con un elevado coste computacional, por el contrario pre-procesamiento requerido en una red convolucional es mucho menor debido a que introduce conceptos como los pesos compartidos, que además mejoras la eficacia en la búsqueda de patrones en las imágenes.

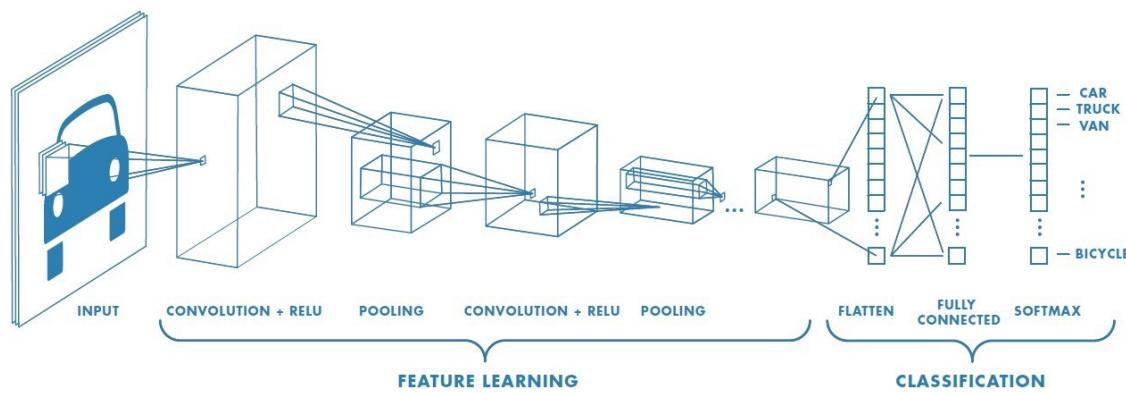


Figura 2.2: Red neuronal con capas convolucionales

La arquitectura de una red convolucional es análoga a la del patrón de conectividad de las neuronas en el cerebro humano y se inspiró en la organización de la corteza visual. Las neuronas individuales responden a los estímulos solo en una región restringida del campo visual conocida como campo receptivo. Una colección de estos campos se superponen para cubrir toda el área visual. Lo que convierte a las redes neuronales en un método de extracción de características, y cuando se tienen los datos significativos

se utiliza un perceptrón multicapa en la mayoría de las ocasiones para realizar tareas de clasificación. Recientemente también se han utilizado para implementar propuestas de generación de datos mediante uso en auto-codificadores, auto-codificadores variacionales, y técnicas de transferencia de estilo.

2.2 | Visión Computacional

La visión por computadora es el campo de la ciencia de la computación que se enfoca en replicar partes de la complejidad del sistema de visión humana y permitir que las computadoras identifiquen y procesen objetos en imágenes y videos de manera similar a como lo hacen los humanos para tratar de hacer procesos de identificación de objetos o de acontecimientos. Antes de la popularización de las redes neuronales, la visión por computadora solo funcionaba en una capacidad limitada y requería de la conjunción de varios tipos de algoritmos para resolver un problema. Estos algoritmos buscan encontrar propiedad en la imagen realizando transformaciones a las mismas, y buscar activaciones de tipos específicos dentro de una imagen.

Como seres humanos, percibimos la estructura tridimensional del mundo que nos rodea con aparente facilidad. Piense en lo vívida que es la percepción tridimensional cuando mira un jarrón de flores en la mesa a su lado. Puede saber la forma y la translucidez de cada pétalo a través de los patrones sutiles de luz y sombras que juegan en su superficie y segmentan sin esfuerzo cada flor del fondo de la escena. Al mirar un retrato de grupo enmarcado, puede contar (y nombrar) fácilmente a todas las personas en la imagen e incluso adivinar sus emociones a partir de su apariencia facial. Los psicólogos de la percepción han pasado décadas tratando de entender cómo funciona el sistema visual y, aunque pueden idear ilusiones ópticas para desentrañar algunos de sus principios.

Los investigadores en visión por computadora han estado desarrollando, en paralelo, técnicas matemáticas para recuperar la forma y apariencia tridimensional de los objetos en imágenes. Ahora disponemos de técnicas fiables para calcular con precisión la tridimensionalidad de estas imágenes a partir de miles de fotografías. Sin embargo, a pesar de todos estos avances, el sueño de que una computadora interprete una imagen al mismo nivel que un niño de dos años (por ejemplo, contando todos los animales en una imagen) sigue siendo esquivo. ¿Por qué es tan difícil la visión? En parte, se debe a que la visión es un problema inverso, en el que buscamos recuperar algunas incógnitas

ante una información insuficiente para concretar completamente la solución. Por tanto, debemos recurrir a modelos probabilísticos y basados en la física para eliminar la ambigüedad entre posibles soluciones. Sin embargo, modelar el mundo visual en toda su rica complejidad es mucho más difícil que, digamos, modelar el tracto vocal que produce los sonidos hablados [Szeliski \(2010\)](#).

Gracias a los avances de la inteligencia artificial y las innovaciones en el aprendizaje profundo, el campo ha sido capaz de generar grandes avances en los últimos años superando en algunos casos a los seres humanos en algunas tareas relacionadas con la detección y etiquetado de objetos.

Los avances en la visión por computadora con aprendizaje profundo se han construido y perfeccionado con el tiempo, principalmente sobre un algoritmo particular: la red neuronal convolucional. En los métodos primitivos los filtros se diseñan a mano, donde se busca utilizar patrones de encendido y apagado en un espacio bidimensional, para buscar activaciones sobre una imagen, por otro lado, con suficiente entrenamiento, las redes convolucionales tiene la capacidad de aprender estos filtros / características para cada conjunto de datos presentado, y donde varían diversas condiciones en las imágenes como iluminación, tonalidad, definición, entre otros.

Otro de los factores que impulsan el crecimiento de la visión por computadora es la cantidad de datos que generamos hoy en día, y que se utilizan para entrenar y mejorar implementaciones de visión por computadora. En cierto nivel, la visión por computadora tiene que ver con el reconocimiento de patrones, haciendo posible identificar bordes, figuras, tonalidades contenidas en las imágenes. Entonces, una forma de entrenar a una red neuronal convencional para que comprenda los datos visuales es alimentarla con imágenes, muchas imágenes, miles, millones si es posible que hayan sido etiquetadas, cabe destacar que dichas imágenes deben de tener en común que contienen una representación de lo que se requiere que la red “aprenda”, y para luego darlos como entrada a la red convolucional.

Algunas aplicaciones importantes de técnicas dentro de área, se pueden ver hoy en día en campos como el manejo de coches autónomos, el reconocimiento de personas mediante cámaras de vigilancia, reconocimiento y clasificación de productos, navegación de robots, identificación de anomalías en cuestiones de salud u otras áreas, estimación de calidad en productos y procesos etc.

2.3 | Drones

El rápido desarrollo y crecimiento de los vehículos aéreos no tripulados (UAV) como plataforma de teledetección, así como los avances en la miniaturización de la instrumentación y los sistemas de datos, han dado como resultado una creciente aceptación de esta tecnología en las comunidades científicas ambientales y de teledetección. Aunque las estrictas regulaciones en todo el mundo aún pueden limitar el uso más amplio de los UAV, su uso en agricultura de precisión, ecología, investigación atmosférica, bioseguridad de respuesta a desastres, monitoreo ecológico y de arrecifes, silvicultura, monitoreo de incendios, mediciones de respuesta rápida para desastres de emergencia, investigación de ciencias de la Tierra , el muestreo de gas volcánico, el monitoreo de gasoductos, plumas mineras, las observaciones humanitarias y las tareas biológicas / de detección de quimioterapia continúan aumentando [Toro and Tsourdos \(2018\)](#).

Por otro lado, se tienen drones con otro tipo de propósito, como los drones militares o los drones comerciales de uso recreativo, que generalmente se intenta que sean eficientes, pero de relativo bajo costo, ya que se utilizan como objeto de entretenimiento, aun que también existen drones comerciales para desarrollar trabajos de fotografía y video. A raíz de ello, recientemente también han salido drones comerciales con propósito educativo, estos tienen por objetivo brindar la capacidad de ejecutar comando de acción sobre los mismos mediante algún lenguaje de programación, u otro tipo de programa. Dado que la ciencia en general y la robótica en particular se han convertido en partes cada vez más importantes de la educación moderna, y cada vez más es el uso de drones con propósito educativo, para que se abran las puertas a mas y diversos desarrollos con este tipo de objetos, y se obtenga una buena base de aprendizaje de ello.



Figura 2.3: DJI Tello

Elementos Utilizados y Proceso de Construcción

Para la creación de nuestro proyecto se utilizó un conjunto de componentes tanto de software como de hardware. El proyecto se compone de cuatro partes, la primera donde se entrena nodelos de redes neuronales convolucionales para la **clasiación de letras**, dichas letras son introducidas por el usuario mediante la realización de un trazo para **escritura en el aire** que se identifica con la portación de un objeto en el aire con un color característico, la construcción de dicho proceso constituye la segunda parte que integra el proyecto. El uso de modelos entrenados para la **detección de pose humana** es una de las herramientas de reciente aparición dentro de las aplicaciones del deep learning. Y es por ello que se ha buscado tomar dichos modelos para obtener información sobre pose humana en tiempo real para la tercera parte del proyecto. Por último, la cuarta parte del proyecto involucra buscar la manera de controlar el **drone** DJI Tello con el lenguaje de programación utilizado e integrar las funcionalidades de control con los datos obtenidos en las partes previas del proyecto.

Los componentes utilizados para el desarrollo y experimentación llevada a cabo en este proyecto, fueron los siguientes:

3.1 | Hardware

■ Parte D: Drone

- Drone DJI Tello (figura 2.3)
- Objeto verde fosforecente

■ Parte B: Escritura en el aire

- Cámara Web
- Objeto verde fosforescente

■ Parte C: OpenPose

- Cámara Web

3.2 | Software

■ Parte A: Clasificación de letras y Parte B: Escritura en el aire

- Python
- Dataset de EMNIST [Cohen et al. \(2017\)](#)
- Librerías de python para inteligencia artificial como [Tensorflow](#) y [PyTorch](#), así como sus dependencias.
- Un ambiente independiente para Python 3 en anaconda para la descarga de librerías necesarias.

■ Parte C: OpenPose

- OpenPose
- CMake
- OpenCV
- Visual Studio
- Controladores de CUDA y CuDNN (en caso de usar GPU)

■ Parte D: Drone

- Librería [djitellopy](#)

3.3 | Instalaciones

Para el correcto funcionamiento de la Parte C del sistema se requiere hacer un cuidadoso proceso de instalación de todas las dependencias escritas dado que ayudan a realizar la construcción de OpenPose desde su código fuente, el proceso depende mucho del sistema operativo donde se este trabajando y las condiciones de hardware con las que se cuente. En el siguiente link se detalla la secuencia de pasos a seguir para realizar la construcción bajo distintas condiciones: [Instalación OpenPose](#). Es importante tomar en cuenta que para que se puedan usar las estimaciones de OpenPose dentro de nuestros códigos en Python, es necesario construir el sistema desde el código fuente.

3.4 | Funcionamiento del Sistema

En este proyecto debido a las condiciones de trabajo la mayor parte del proceso fue experimental, ya que es preciso realizar muchos experimentos de calibración, el funcionamiento no tiene una linea definida, y es cambiante conforme a las condiciones que se presentan al momento de utilizar el sistema. Existen diversos factores a considerar. Como ya se ha precisado anteriormente, la idea de este proyecto, es tomar un dispositivo de hardware de relativa fácil accesibilidad, y utilizarlo como medio aplicativo para mostrar algunas de las funcionalidades que se le puede dar a las redes neuronales convoluciones, y que en este caso se utilizan para desarrollar un sistema de uso recreativo, pero en realidad pueden aplicarse a zonas de impacto directo en la productividad industrial, cuestiones de salud, aprendizaje, entre otros.

De forma específica se tomo un drone comercial de dimensiones pequeñas, cuyo nombre comercial es **DJI Tello**, con ello se realizaron cuatro implementaciones de código: Código de control del drone, y preparación para su uso en interfaces de interacción con un usuario mediante el lenguaje de programación Python; Código de aplicación de reconocimiento de letras mayúsculas manuscritas, Código de captura de trazos de dibujo en el aire y con ello su respectivo mapeo a una imagen, el ultimo código implementado fue donde se utiliza un modelo ya entrenado para reconocimiento de pose humana, conocido como **OpenPose**, y se aplica para establecer ordenes de control e interacción mediante gestos con el cuerpo humano.

3.4.1 | Parte A: Clasificación de letras

El modelo neuronal que se utiliza para clasificación de trazos en el aire como letras, fue construido dos veces, para probar eficiencia en el entorno real, y se realizó mediante el uso de dos bibliotecas para desarrollo de Deep Learning implementadas para uso mediante el lenguaje de programación Python, con cada una se creó un modelo. Las bibliotecas utilizadas fueron **TensorFlow** mediante su API, **Keras** para construir uno de los modelos, y **Pytorch**, para construir el segundo modelo. Ambos modelos son redes neuronales convolucionales, con arquitectura similar, e hiper parámetros de igual forma similares, ello para no alterar los resultados de eficiencia respecto a las variaciones en arquitectura. Además se realizó una elección de hiper parámetros igual para ambos modelos, como el tasa de aprendizaje, y la función de pérdida utilizada. Los datos utilizados para el entrenamiento fueron obtenidos de forma directa desde los repositorios de las bibliotecas utilizadas. El conjunto de datos utilizado, fue el denominado EMNIST, el cual se compone de 88,800 imágenes de entrenamiento, y 14,800 imágenes de prueba de 28×28 píxeles, de caracteres escritos a mano, y entrenados el mismo número de épocas. Se cuidó que la arquitectura, no fuese tan grande ni compleja pero intentando conseguir buenos resultados de exactitud, para que al momento de hacer la evaluación el procesamiento sea rápido. Las imágenes recibidas para evaluación son preprocesadas para ser ajustadas al formato que reciben ambas arquitecturas, y que sean consistentes con los datos de entrenamiento, a la salida se obtiene un número que representa la salida con más alta probabilidad, es decir la letra que es más probable que haya ingresado el usuario, y se espera que sea la correcta.

Una vez que se le da el tratamiento adecuado, la imagen ya procesada, se evalúa en una red neuronal convencional previamente entrenada para el reconocimiento de letras mediante el uso de una base de datos de caracteres escritos a mano conocida como: EMNIST, y se obtiene la clasificación del trazo realizado por usuario, convergente a una letra, con ello el resultado es mostrado en la pantalla de transmisión de los datos, es decir imágenes obtenidas por la cámara.

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 32, 28, 28]	320
BatchNorm2d-2	[-1, 32, 28, 28]	64
LeakyReLU-3	[-1, 32, 28, 28]	0
ConvBlock-4	[-1, 32, 28, 28]	0
Conv2d-5	[-1, 32, 14, 14]	9,248
BatchNorm2d-6	[-1, 32, 14, 14]	64
LeakyReLU-7	[-1, 32, 14, 14]	0
ConvBlock-8	[-1, 32, 14, 14]	0
Conv2d-9	[-1, 64, 14, 14]	18,496
BatchNorm2d-10	[-1, 64, 14, 14]	128
LeakyReLU-11	[-1, 64, 14, 14]	0
ConvBlock-12	[-1, 64, 14, 14]	0
Conv2d-13	[-1, 64, 7, 7]	36,928
BatchNorm2d-14	[-1, 64, 7, 7]	128
LeakyReLU-15	[-1, 64, 7, 7]	0
ConvBlock-16	[-1, 64, 7, 7]	0
Flatten-17	[-1, 3136]	0
Linear-18	[-1, 128]	401,536
BatchNorm1d-19	[-1, 128]	256
LeakyReLU-20	[-1, 128]	0
Dropout-21	[-1, 128]	0
Linear-22	[-1, 27]	3,483
<hr/>		
Total params:	470,651	
Trainable params:	470,651	
Non-trainable params:	0	
<hr/>		
Input size (MB):	0.00	
Forward/backward pass size (MB):	1.46	
Params size (MB):	1.80	
Estimated Total Size (MB):	3.26	
<hr/>		

Figura 3.1: Resumen de la arquitectura en PyTorch

Layer (type)	Output Shape	Param #
<hr/>		
input_5 (InputLayer)	(None, 28, 28, 1)	0
conv2d_17 (Conv2D)	(None, 28, 28, 32)	320
batch_normalization_19 (BatchNormalization)	(None, 28, 28, 32)	128
leaky_re_lu_19 (LeakyReLU)	(None, 28, 28, 32)	0
conv2d_18 (Conv2D)	(None, 14, 14, 32)	9248
batch_normalization_20 (BatchNormalization)	(None, 14, 14, 32)	128
leaky_re_lu_20 (LeakyReLU)	(None, 14, 14, 32)	0
conv2d_19 (Conv2D)	(None, 14, 14, 64)	18496
batch_normalization_21 (BatchNormalization)	(None, 14, 14, 64)	256
leaky_re_lu_21 (LeakyReLU)	(None, 14, 14, 64)	0
conv2d_20 (Conv2D)	(None, 7, 7, 64)	36928
batch_normalization_22 (BatchNormalization)	(None, 7, 7, 64)	256
leaky_re_lu_22 (LeakyReLU)	(None, 7, 7, 64)	0
flatten_5 (Flatten)	(None, 3136)	0
dense_6 (Dense)	(None, 128)	401536
batch_normalization_23 (BatchNormalization)	(None, 128)	512
leaky_re_lu_23 (LeakyReLU)	(None, 128)	0
dropout_4 (Dropout)	(None, 128)	0
dense_7 (Dense)	(None, 27)	3483
activation_4 (Activation)	(None, 27)	0
<hr/>		
Total params:	471,291	
Trainable params:	470,651	
Non-trainable params:	640	15

Figura 3.2: Resumen de la arquitectura en Tensorflow-Keras

3.4.2 | Parte B: Escritura en el Aire

Para esta parte del sistema se tomo una aproximación donde mediante la adquisición de imágenes en tiempo real a través de una cámara se realiza el mapeo de movimiento en tiempo real de un objeto con un color característico, en este caso verde fosforecente. Dicho movimiento del objeto característico, debe ser el intento de trazo de una letra en el aire por parte el usuario. Esto se realiza mediante el procesamiento de imágenes a través de sus capas, extrayendo la sección con el color característico del marco o imagen en cada tiempo de un vídeo, que puede descomponerse en una secuencia de imágenes. Una que se tiene identificado un color característico, se puede realizar la activación mediante un medio físico, como el pulsar un botón o una tecla para comenzar a marcar el camino o trazo que el usuario realiza con dicho objeto, para ello se obtiene la máscara de la zona específica donde se ubica el objeto, y se marca el camino que el usuario realiza con dicho objeto. Cuando el usuario ha finalizado el trazo, puede nuevamente mediante el pulsar de una tecla detener el grabado del trazo. Una vez que se tiene el trazo completo realizado por el usuario, se acota la zona donde se ubica el trazo, mediante una búsqueda de zonas con el color característico, se ubica la de mayor área, y se llenan espacios vacíos entre el trazo, ello para tener un trazo sólido. Una vez que se ubica por completo la zona del trazo, y se delimita, la misma se corta y se ajusta con medidas específicas para poder posteriormente procesarse. Posterior a ello la imagen obtenida es mandada a ser evaluada por el modelo de clasificación de letras descrito anteriormente.

La cámara utilizada para adquisición del trazo de usuario es la integrada al drone DJI Tello, y una vez que se obtiene el resultado de clasificación del trazo, se ejecuta una serie de comandos de control para realizar con el drone en el aire, el trazo de la letra clasificada, y se muestra al usuario. Algunas muestras de los resultados se encuentran en las figuras 3.3 y 3.4

3.4.3 | Parte C: OpenPose

OpenPose es un software de código abierto el cual permite mediante un modelo ya entrenados realizar el reconocimiento de formas humanas en imágenes, vídeos almacenados o en tiempo real, tiene una alta efectividad tanto teórica como práctica para esta tarea. Al rededor de esta tecnología se han desarrollado diversos tipos de sistemas, que se privilegian de su uso. Una de las aplicaciones fuertes, a las que se ha aplicado OpenPose es para el control a distancia. Este es el caso presentado en este proyecto, re-

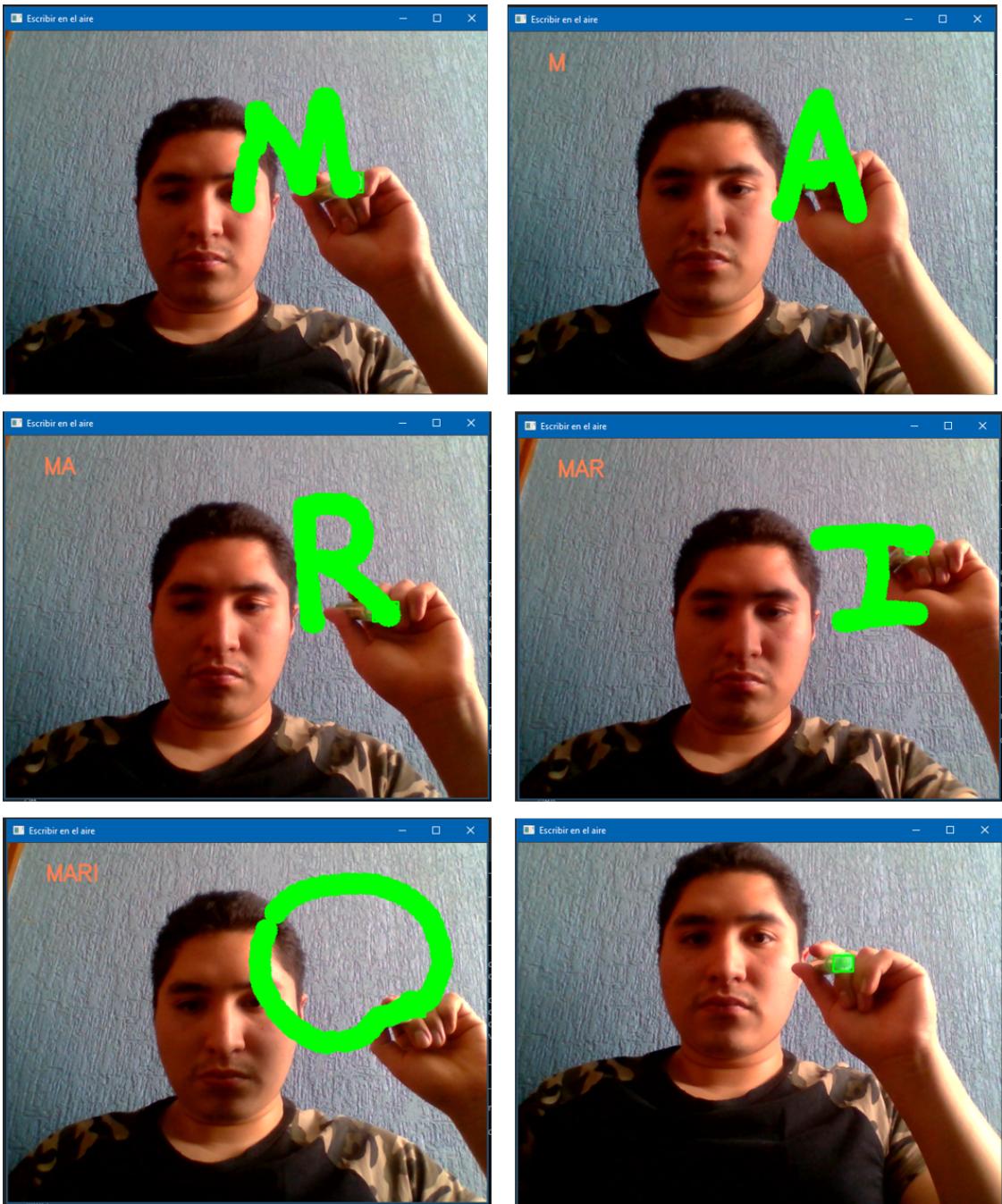


Figura 3.3: Mario escribiendo en el aire

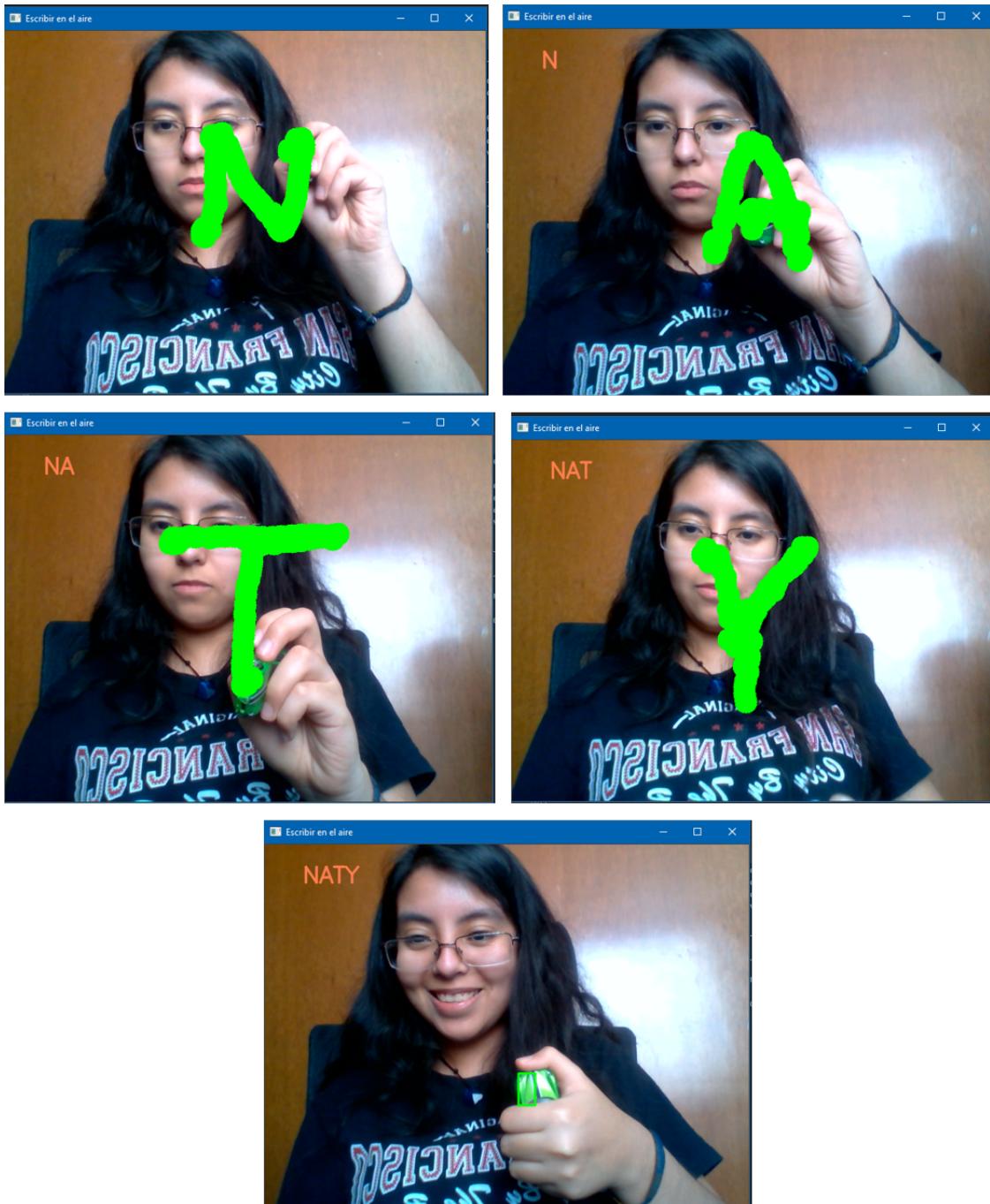


Figura 3.4: Natalia escribiendo en el aire

alizando una implementación, que requiere de la construcción de su código fuente, en vez del uso de su versión portable, ello debido a que se quiere extraer información del reconocimiento de los puntos en poses humanas realizado además de el procesamiento y la ejecución de control del drone mediante señales de posición humana todo ello a través del lenguaje de programación Python. En las figuras 3.5 y 4.1, se muestra la estimación de pose que registra OpenPose.

Dentro de un script se realiza el control del drone, y se ejecuta el modelo de análisis de pose Humana de OpenPose, se obtienen los puntos identificados de la pose humana, para posteriormente realizar evaluaciones de los valores en la relación de la posición espacial de las partes de cuerpo para determinar si el usuario esta intentando dar una señal de control mediante su posición corporal. Para ello, si el sistema no tiene identificada a una persona, realiza una búsqueda mediante giros, para intentar localizar una forma humana, una vez que la identifica, se mantiene enfocando hacia la persona, y realizando movimientos conforme a las instrucciones que brinda el usuario mientras cambia de posición. Los movimiento que realiza el drone cambian conforme a se actualiza la información de los puntos provenientes de la pose de una persona.

3.4.4 | Parte D: Drone

El control del drone DJI Tello, se realiza mediante el uso de la biblioteca djitellopy, ya que la misma cuenta ya con una amplia variedad de funciones para control del drone mediante el lenguaje de programación Python, dichas indicaciones pueden ser mandadas en tiempo real al drone mediante la ejecución del código en un dispositivo conectado a través de WI-FI al mismo, esta biblioteca permite explotar la capacidad del drone incluso mas que en su funcionalidad de control normal (mediante una aplicación para dispositivos móviles), debido a que se pueden establecer secuencias mas complejas de pasos y movimientos, lo que nos lleva a tener mas combinaciones de métodos de entrada para ejecutar dichas acciones. La biblioteca mencionada ya cuenta con una amplia variedad de acciones que se pueden mandar al drone, sin embargo, debido a los requerimientos específicos de nuestro sistema se tuvieron que construir algunas estructuras, a forma de rutinas, para que el drone reaccionara de forma adecuada ante diversas situaciones, tal como la identificación de una letra proveniente del trazo de un usuario, o alguna señal de control identificada mediante el uso de Openpose. Con ello se realizaron integraciones a los respectivos sistemas tanto el de identificación de letras en el aire como el de reconocimiento de pose humana para integrar las funcionalidades que los mismos proveen de identificación de patrones y con ello ejercer el control.

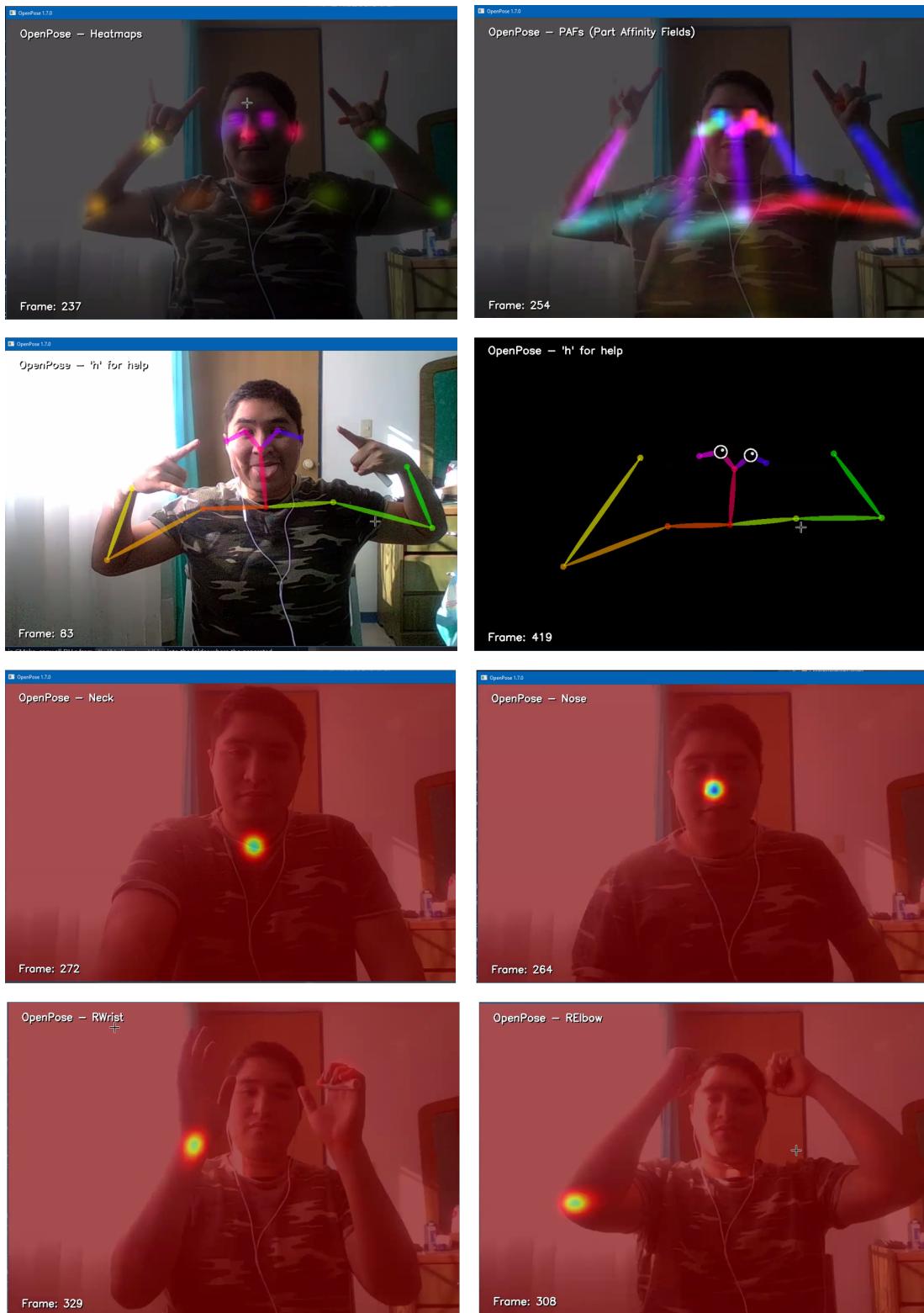


Figura 3.5: Mario en OpenPose

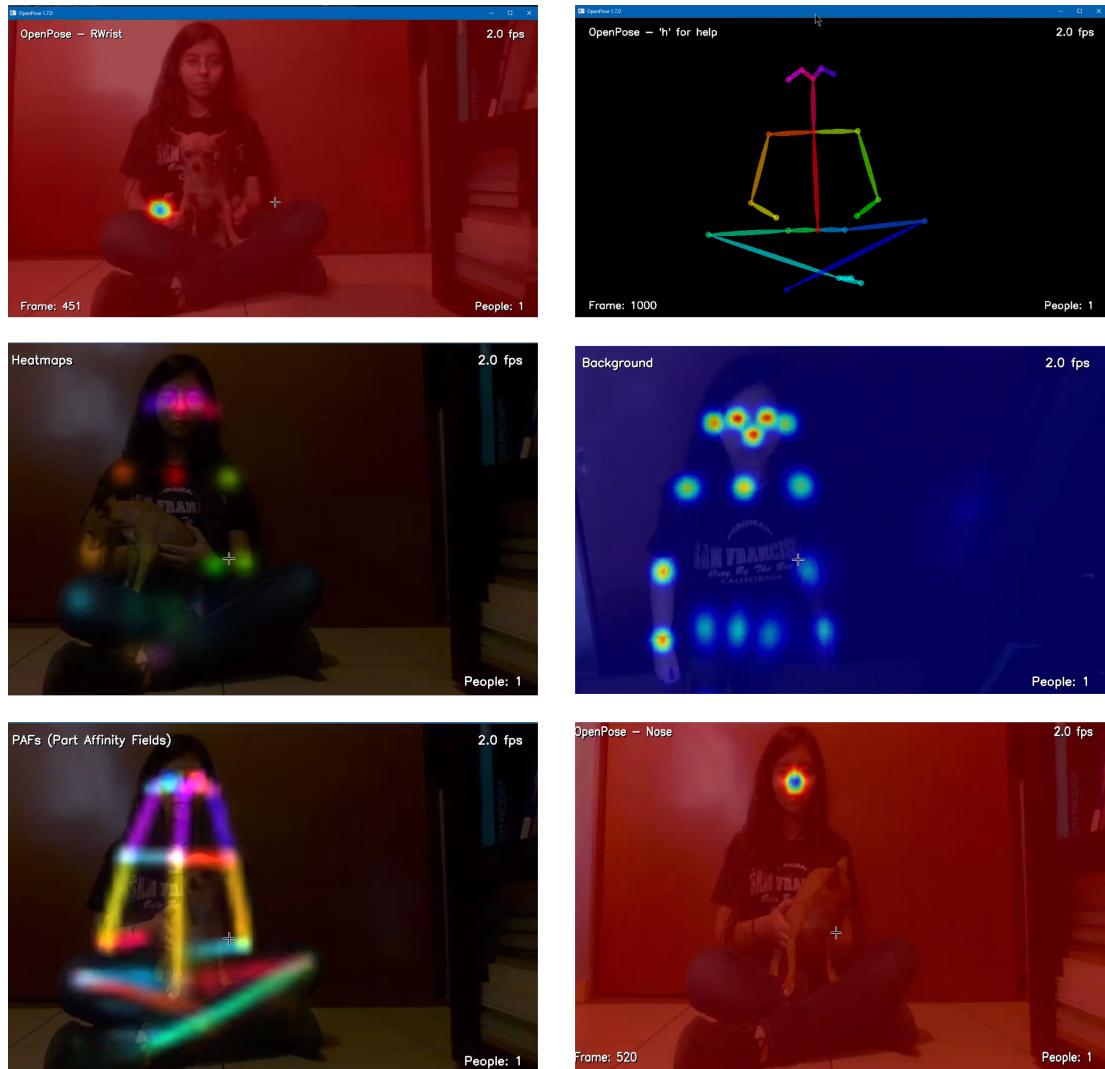


Figura 3.6: Natalia en OpenPose

Resultados obtenidos

4.1 | Uso de redes neuronales

Para la primera parte del proyecto se realizaron dos entrenamientos con el mismo conjunto de datos y la misma arquitectura, pero realizada por medio de bibliotecas diferentes en Python, un modelo con TensorFlow y otro con PyTorch. El primer entrenamiento se hizo utilizando la biblioteca de TensorFlow, fue un entrenamiento de 10 épocas que se realizó en 27 minutos aproximadamente, haciendo uso de una GPU GTX1060, y se alcanzó una exactitud del 92.97% en el set de validación con la configuración de la arquitectura mostrada en la Figura 3.2, como función de pérdida el optimizador ADAM, y una tasa de aprendizaje de 5^{-4} , mientras que al utilizar la biblioteca de Pytorch, con la misma arquitectura, que se muestra en la figura 3.1 y los mismos hiper-parámetros se alcanzó una exactitud del 91.50%, pero aumentando el tiempo de ejecución a 32 minutos.

Se puede observar que a pesar de la poca cantidad de épocas, se obtuvieron buenos resultados, esto debido a una buena elección de parámetros para el modelo. A la hora del reconocimiento en tiempo real mediante los datos obtenidos de la cámara se puede concluir que ambos modelos funcionan bien, y realizan el reconocimiento aun que el pulso de la persona no sea muy bueno, sin embargo, si deben ser los trazos relativamente bien definidos, y sin cruces a normales, por que eso hace que brinde un resultado erróneo, es decir tiene alta efectividad en casos donde los datos son claros, pero es bastante sensible a variaciones relativamente pequeñas en la forma de la letra que se quiere ingresar. Además, llegan a mostrar un sesgo con alguna letra, tal es el caso del modelo entrenado con TensorFlow, el cual cuando se le da una figura que no parece ninguna letra siempre la marca como una “Q”. Pero en general el reconocimiento de las letras en el aire es efectivo, como se muestra en la figura ??, y en la figura 3.4.

4.2 | Implementación de OpenPose

La implementación con OpenPose es complicada si no se siguen los pasos con rigurosidad, y prestando atención todo el tiempo a lo que se está haciendo, además de que se tienen que tener presente características del hardware utilizado, para poder seguir el proceso de instalación correcto. Además para la instalación, dependiendo del sistema operativo puede requerir de instalar previamente software muy pesado, y que puede ser tardado de descargar e instalar, si es que no se cuenta previamente con el, para poder construir OpenPose desde el código fuente. Posterior a ello, una vez que se tiene construida la solución, el software tiene requerimientos computacionales exigentes, y aun con procesadores de gráficos dedicados los frames por segundo llegan a ser bajos, es decir no se presenta un análisis tan continuo de la actividad. Sin embargo, el modelo de OpenPose es muy bueno, y lo muestra en como nos segmenta durante las pruebas, incluso cuando hay partes del cuerpo no visibles explícitamente para la cámara, el modelo hace una estimación bastante precisa de la localización de dicho segmento. Esto lo hace de gran interés y de gran fiabilidad para la implementación de sistemas que requieras control por medio de pose corporal analizada por vídeo, sin embargo debido a requerimientos, para dispositivos móviles o de bajo poder computacional como el drone, se requiere de un dispositivo intermediario para realizar el procesamiento del modelo. Además las capacidades de openpose pueden enfocarse dependiendo la necesidad ya que cuenta con modelos e estimación de pose, de mano y de rostro y en alguno de ellos, variantes de modelos enterados en diferentes plataformas. El que nosotros usamos es el modelo de estimación de pose “*body25*”, y con el cual obtuvimos buenos resultados.

4.3 | Sistema de control del Drone

En nuestra búsqueda sobre como llevar a cabo el control del drone, nos encontramos con al menos tres bibliotecas distintas que cuentan con funciones para el control del drone que nosotros utilizamos, sin embargo al estarlas probando, se tenían problemas de conexión o generaban algunos fallos internos, en el código de las propias bibliotecas. Al final decidimos probar mas a fondo con dos de ellas, la primera **TelloPy**, y la segunda **DJITelloPy**, ambas en las pruebas iniciales realizadas mostraron algunas debilidades, como el fallo en el envío de algunos mensajes, y falta de control de estabilidad del drone, el mismo, a diferencia de cuando se activa mediante la aplicación nativa del mismo, cuando esta detenido en el aire sin desplazarse se mantiene en un lugar fijo, con ligeros movimientos, pero manteniéndose en una sola zona, sin embargo, en el control con las

bibliotecas, el mismo si no se le daba alguna indicación se iba hacia algún lado, hasta que se ejecutara sobre su control una acción contraria, ello añadido al fallo repentino en el envío de mensajes ocasiono choques inesperados en mas de una ocasión, provocando que se perdiera de vista el objetivo, es decir el punto de donde obtener información a analizar. Sin embargo, decidimos quedarnos con la biblioteca **DJITelloPy**, debido a que utilizaba soluciones mas flexibles y contaba con más condiciones de control ya implementadas, tal como volteretas, o rutas de curva suave, etc.

4.4 | Practicidad de las bibliotecas y softwares requeridos

En el presente proyecto se hizo uso de una gran cantidad de librerías de libre uso con resultados y versatilidad de condiciones sorprendente, sin embargo el acondicionamiento de los ambientes dentro del sistema computacional como su correcta ejecución pueden llegar a ser bastante complicados y se deben tomar en cuenta muchas precauciones para implementarlo.

Una vez tomado esto en cuenta, es importante mencionar que son librerías que pueden facilitar gran parte del proceso a la larga, puesto que contienen una gran cantidad de funciones y características que pueden ser aplicadas o utilizadas para la obtención de información de una manera muy sencilla.

4.5 | Uso integrado de hardware y software

Se realizó una multitud de pruebas, sin embargo estas duraban muy poco tiempo al tener la limitante del uso de la batería, condiciones cerradas para maniobrar y lo cambiante de la luz durante el transcurso del día.

Durante las pruebas en todo el proyecto se pudo observar que cuando el drone comienza a trabajar ejecuta todos los movimientos e instrucciones perfectamente, pero a medida que pasa el tiempo, y sucede de manera rápida, puede perder potencia y precisión a la hora de volar y maniobrar, esto puede dificultar la interacción con el y es importante dejar reposar las baterías y el drone antes de cargarlos y de volverlos a utilizar.

Debido a que este proyecto fue altamente experimental y práctico al fusionar el sistema de control del drone con el código para realizar el trazo de letras en el aire, y mandar una rutina mediante el reconocimiento, la parte de procesamiento de imagen, es decir del trazo en el aire, es en la mayoría de los casos exitosa, encontrando la solución correcta, sin embargo, la traducción de dicha información clasificada a comandos de control, aun que a nivel de software se realiza de forma correcta, el comportamiento real del drone llega a ser muy cambiante, incluso para la misma acción, ya que influyen los entornos en los que se realiza el experimento, el estado de la conexión con la computadora, y el estado de batería del drone, la cual solo da hasta 13 minutos de vuelo. Sin embargo en general el comportamiento de dicha funcionalidad es en general aceptable la mayoría de las ocasiones.

Para el apartado de OpenPose, como ya se comentó el software funciona muy bien, en casi todos los casos, no afectándole mucho la variabilidad de entornos en los que se pruebe, las condiciones de iluminación, a menos que sea en casos extremos, o incluso en casos donde se encuentre una multitud de personas. Sin embargo, el factor que si puede afectar no su rendimiento en estimación de pose, si no, en la viabilidad de uso para integración con el drone, es los recursos computacionales con los que cuenta el dispositivo donde se ejecuta el programa. Ello causo que buscáramos algunas soluciones alternativas, como llamar al procesamiento de una serie de imágenes, tomadas de vídeo en tiempo real, pero en lugar de ejecutar el procesador de vídeo en vivo de OpenPose, guardáramos los frames del vídeo en tiempo real en un directorio, y al mismo tiempo los tomáramos con la funcionalidad de OpenPose para estimación de poses en imágenes ya almacenadas, debido a que este modelo gasta un poco de menos recursos resulta mas rápido, y es posible ver la secuencia de toma de captura y el procesamiento con OpenPose en tiempo real de forma mas fluida que con el uso directo del análisis de vídeo de OpenPose, sin embargo a costa del uso de memoria para alojar las imágenes generadas, aun que el mismo directorio era vaciado al final de la ejecución. Debido a ello, y a que encontrar relaciones correctas en algunos casos para determinar correctamente cuando mandar una acción correspondiente a una pose requiere de mucha experimentación y análisis de valores, la función con el sistema de control del drone, con el análisis en tiempo real con OpenPose, aun no trabaja de forma óptima, y es un proceso que requiere bastante tiempo mas de desarrollo para refinar el sistema. Sin embargo se busca conseguir el estado óptimo deseado continuando posteriormente con este proyecto.



Figura 4.1: Pruebas del sistema final integrado

Conclusiones

En este proyecto se realizó la experimentación de una implementación para un sistema de interacción con un drone de propósitos educativos mediante el uso de técnicas populares de visión computacional, en específico se trabajo con redes neuronales, y modelos pre-entrenados, ello con el fin de hacer posible el mandar instrucciones de control mediante el análisis de la acción de dibujar letras en el aire frente a una cámara o hacer gestos utilizando las manos, y otras poses corporales. Para esto se uso el lenguaje de programación Python en archivos .py y .ipynb el cual contiene características que nos ayudan a realizar el manejo de datos de forma eficiente y a preservar una sintaxis clara de la estructura de funcionamiento general, por otro lado el uso de los notebooks de python nos ayudo observar paso a paso los procesos realizados, sobre todo en las dos primeras partes de proyecto, pero para la parte final es mucho más eficiente utilizar únicamente los archivos .py, es decir scripts de Python, permitiendo así observar lo que sucede en la cámara del drone a través de la transmisión de vídeo a la computadora y a los mensajes enviados a la terminal. Para dar formato y realizar el proyecto, este se dividió en cuatro partes, una para el reconocimiento y clasificación de trazos en el aire, la segunda parte para la implementación de modelos pre-entrenados de estimación de pose, en la tercera parte, se trabajo sobre la forma de ejercer control sobre el comportamiento de drone, y por ultimo la integración de los elementos trabajados. De forma individual se consiguió el correcto funcionamiento de las tres primeras partes del proyecto, para la última parte se trabajo sobre ella consiguiendo resolver algunas tareas de forma eficiente, sin embargo, para otras, es necesario realizar mucho más trabajo en el desarrollo de la implementación, lo cual implica tanto tiempo de desarrollo de código, como de experimentación, con esto sería posible alcanzar una eficiencia en el sistema, tal que llegar a ser intuitivo para la mayoría de los usuarios, y funcionar correctamente bajo distintas condiciones en el

ambiente. El objetivo principal de este proyecto se cumplió al poder realizar un sistema de control e interacción integrando el uso de redes neuronales convolucionales a su desarrollo. Esto desde su concepción teórica nos permitió tener un acercamiento a los conceptos vistos en clase y que optimizan en gran medida los esquemas generales de funcionamiento de sistemas dentro del área de Inteligencia Artificial. Con lo cual se ve de forma directa una aplicabilidad de estos conceptos a soluciones tecnológicas actuales y con gran popularidad como lo son el vuelo semi-autónomo de drones recreativos, propósitos educacionales, comerciales, etc. Durante el desarrollo se encontraron algunas limitantes como la compatibilidad entre bibliotecas, distribuciones, sistemas operativos, controladores, conexiones a la red, requerimientos de capacidad de computo, entre otras cosas, pero siendo una de las principales limitantes el tiempo de desarrollo, debido a eso el problema tuvo que ser acotado más de lo que se había planeado en un inicio, restringiendo ciertos movimientos por parte del usuario y también el tiempo de vuelo. Respecto a las pruebas realizadas se lograron resultados bastante precisos a la hora de su ejecución ya integrada, sin embargo el tema de accesibilidad a ello es aún una cuestión que queda por resolver. Esto es posible lograrlo encontrando alternativas a ciertas bibliotecas utilizadas actualmente y aumentando la complejidad del código añadiendo mas funciones que permitan realizar las actividades de manera más fluida y ordenada.

5.1 | Posibles extensiones del proyecto

Hay gran cantidad de manera de ampliar el alcance de este proyecto, comenzando con el tema de accesibilidad a él, el añadir funciones que permitan un mayor rango de comportamientos e interacción, aumentar la autonomía del vuelo del drone y brindar una mayor flexibilidad respecto a las condiciones de uso de nuestro sistema. Sin embargo la mayoría de ellos es posible solventarlos, invirtiendo mas tiempo al desarrollo y experimentación. aun que en realidad el caso ideal, y que incrementaría la viabilidad de uso de este tipo de algoritmos de forma recurrente, en dispositivos como drones, es el aumento en la capacidad de calculo de dichos dispositivos, o en el desarrollo modelos eficientes, pero requieran de menores recursos computacionales, para eliminar por completo la necesidad de un equipo intermediario. Otra aproximación que sí hace uso de intermediarios pero añade portabilidad al sistema es el poder implementar los códigos junto con versiones reducidas de los modelos necesarios dentro de una placa de desarrollo Raspberry Pi, de manera que los cálculos puedan reducirse y sea más fácil

implementar un proyecto de este tipo.

5.2 | Comentarios Finales

Con la realización de este proyecto, quedamos satisfechos con los resultados obtenidos dadas las condiciones y herramientas utilizadas, con la experimentación realizada, el cumplimiento de todos los objetivos y la experiencia de desarrollo adquirido. Logramos fortalecer nuestras habilidades de desarrollo específicamente en Python aplicado a hardware e inteligencia artificial y nuestra experiencia acerca de la integración de herramientas en línea para la implementación de este tipo de sistemas. Nos apoyamos de preguntas realizadas en foros de programación y en la documentación dada por distintas librerías utilizadas. Dada la complejidad de las acciones que realiza el sistema fueron utilizadas librerías poco comunes como:

- [OpenPose](#)
- [DjiTelloPy](#)
- [Tensorflow](#)
- [PyTorch](#)
- [OpenCV](#)

Referencias

- Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. Emnist: Extending mnist to handwritten letters. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2921–2926. IEEE, 2017.
- Larry Hardesty. Explained: Neural networks ballyhooed artificial-intelligence technique known as “deep learning” revives 70-year-old idea. 2017.
- Michael A Nielsen. *Neural networks and deep learning*, volume 25. Determination press San Francisco, CA, 2015.
- A. Shah, B. Jain, B. Agrawal, S. Jain, and S. Shim. Problem solving chatbot for data structures. In *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 184–189, 2018. doi: 10.1109/CCWC.2018.8301734.
- Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.
- Felipe Gonzalez Toro and Antonios Tsourdos. *UAV or drones for remote sensing applications*. MDPI-Multidisciplinary Digital Publishing Institute, 2018.