Universidad CENFOTEC

Maestría en Tecnología de Bases de Datos

Documento Final de Proyecto de Investigación Aplicada 2

Tema:

Desarrollo de un Modelo de Machine Learning para la Detección y Traducción de Señas de LESCO al español

Autor:

Zúñiga Umaña Andrés

Octubre 2024

TRIBUNAL EXAMINADOR

Este proyecto fue aprobado por el Tribunal Examinador de la carrera: **Maestría en Tecnología de Bases de Datos,** requisito para optar por el título de grado de **Maestría,** para el estudiante: **Andrés Zúñiga Umaña.**

Digitally signed by MIGUEL PEREZ MONTERO (FIRMA)
Date: 2024.10.18 10:59:40 -06'00'

M.Sc. Miguel Pérez Montero
Tutor

LUIS CARLOS NARANJO ZELEDON (FIRMA)

Firmado digitalmente por LUIS CARLOS NARANJO ZELEDON (FIRMA)
Fecha: 2024.10.18 11:17:47 -06'00'

Dr. Luis Carlos Naranjo Zeledón
Lector 1

JASON ULLOA HERNANDEZ (FIRMA)

Firmado digitalmente por JASON ULLOA HERNANDEZ (FIRMA)
Fecha: 2024.10.18 17:43:11 -06'00'

M.Sc. Jason Ulloa Hernández
Lector 2

San José, Costa Rica, 17 de octubre 2024

Firmada digitalmente, de conformidad con la Ley de Certificados, Firmas Digitales y Documentos Electrónicos N° 8454, destacando el artículo 9°-

Desarrollo de un Modelo de Machine Learning para la Detección y Traducción de Señas de LESCO al español

Development of a Machine Learning Model for the Detection and Translation of LESCO Signs into Spanish

Andrés Zúñiga Umaña
azunigau@ucenfotec.ac.cr

Miguel Pérez Montero
mperez@ucenfotec.ac.cr

Universidad CENFOTEC, Maestría, Bases de Datos con Analítica

Resumen

La inclusión de las personas con discapacidad es un tema que, con el paso de los años, ha ido tomando más fuerza, y esto por sí solo es algo bueno, sin embargo, todos los cambios necesarios para mejorar la inclusión han descubierto que existe una gran brecha en los sistemas de acceso público para implementar procesos de inclusión para personas con discapacidad. Enfocándose específicamente en las personas sordas, procesos diarios como subirse al bus, realizar una transacción en un banco, o comprar algo en una tienda, significa un gran esfuerzo, puesto que la mayor parte de la población costarricense no tiene conocimiento de la lengua LESCO. El aporte de este artículo propone utilizar técnicas avanzadas de visión por computadora y aprendizaje automático, para facilitar la comunicación y la inclusión de las personas sordas en Costa Rica. La investigación se centra en un número limitado de señas y se espera que el modelo tenga un impacto significativo en la reducción de la discriminación y en la mejora de la integración social de la comunidad sorda. Asimismo, esta investigación debe ser un punto de inicio importante para que otros trabajos de investigación contribuyan a montar el esquema completo de LESCO utilizando las herramientas propuestas.

Palabras clave: Costa Rica, LESCO, Machine Learning, Visión por computadora, traductor, lengua.

Abstract

The inclusion of people with disabilities is an issue that, over the years, has gained more strength, and this in itself is a good thing. However, all the changes necessary to improve inclusion have discovered that there is a great gap in public access systems to implement inclusion processes for people with disabilities. Focusing specifically on deaf people, daily processes such as getting on the bus, making a transaction in a bank, or buying something in a store, mean a great effort, since the majority of the Costa Rican population does not have knowledge of the LESCO language. The contribution of this article proposes to use advanced computer vision and machine learning techniques to facilitate communication and inclusion of deaf people in Costa Rica. The research focuses on a limited number of signs and the model is expected to have a significant impact on reducing discrimination and improving the social integration of the deaf community.

Keywords: Costa Rica, LESCO, Machine Learning, Computer Vision, translate, language.

I: INTRODUCCIÓN

La primera mención en la historia sobre una “lengua de señas” viene por parte de Sócrates en el siglo 5 A.C., donde expresa: “Si no tuviéramos voz ni lengua y quisiéramos expresarnos cosas unos a otros, ¿no intentaríamos hacer señas moviendo las manos, la cabeza y el resto del cuerpo, tal como lo hacen los tontos actualmente?”. El filósofo griego da a entender que, de no tener una lengua hablada, el instinto humano llevaría a tratar de comunicarse por medio de señas y gestos sus cuerpos.

Existen muchos tipos de lenguas de señas alrededor del mundo, siendo la Lengua de señas americana (ASL) de las más conocidas. Hablando específicamente de Costa Rica, la Lengua de señas costarricense (LESCO) es la que se utiliza comúnmente por la comunidad sorda del país. La comunidad de personas sordas de Costa Rica, según el Instituto Nacional de Estadística y Censos, correspondía, aproximadamente, al 1.5% de la población en el 2016, lo cual demuestra la necesidad de implementar nuevos procesos y medidas de inclusión enfocados en esta porción del pueblo costarricense.

Como parte de los esfuerzos de inclusión realizados en los últimos años, se han hecho cambios para facilitar la comunicación con la comunidad sorda, entre estos cambios se puede destacar la inclusión de una persona traductora del español al LESCO en las comunicaciones del gobierno y otros programas de televisión, o la adición de subtítulos a programas grabados y distribuidos en Costa Rica.

Por otra parte, excluyendo pequeños proyectos dirigidos a resolver esta problemática, destacando el Traductor LESCO desarrollado por el Tecnológico de Costa Rica, el cual es una herramienta bastante limitada, no existen otras herramientas intencionadas a cambiar la situación actual para las personas sordas, demostrando también la falta de interés por parte de entidades del gobierno para solventar esta problemática, así como la falta de conocimiento del pueblo costarricense sobre la misma.

Con respecto a las tecnologías actuales, la implementación de inteligencia artificial y aprendizaje automático o ***Machine Learning*** ha comenzado a verse más comúnmente en los procesos diarios de las personas, desde algo tan simple como búsquedas en internet, hasta la creación de imágenes a partir de texto puede ser realizado por cualquier persona con su dispositivo personal. Basado en esto, se aborda el presente proyecto donde se considera la posibilidad de crear un modelo de aprendizaje automático que sea capaz de ser entrenado con datos reales de LESCO y que así mismo pueda detectar en tiempo real una seña realizada por un sujeto ante una cámara.

El desarrollo de este proyecto efectivamente significa uno de los primeros sistemas que implementan procesos innovadores de inteligencia artificial para mejorar el estilo de vida de las personas de la comunidad sorda de Costa Rica. Así mismo, es necesario recalcar que, debido a su carácter académico, el presente proyecto fue realizado con recursos limitados, lo cual puede afectar la precisión de los resultados del modelo.

II: METODOLOGÍA

Dado que el objetivo general del proyecto está basado en la creación y aplicación con datos del mundo real, se puede afirmar que la investigación es de tipo aplicado. Sin embargo, también es correcto definir un tipo

de investigación mixta, incluyendo también una sección exploratoria, debido a que el problema estudiado no ha sido aplicado ampliamente en el país.

A pesar de que los modelos de traducción de lengua de señas ya existen, siguen siendo un tema no tan explorado a nivel mundial, y menos a nivel nacional. Es por esto que se reconoce como una buena oportunidad aplicar este proyecto, y de esta forma obtener resultados nuevos y actualizados, además de generar nuevas ideas para proyectos futuros en personas o entidades con interés de profundizar en el tema.

Además de la literatura revisada, también se analizan una serie de proyectos de ejemplo, implementados por colegas del área, de forma que se pueda tomar dicha información como base para aplicar las diferentes librerías a utilizar en el proyecto y también poder presentar un modelo final más preciso y que refleje las intenciones de la investigación.
Con los resultados del modelo se generan una serie de conclusiones y recomendaciones para personas que deseen implementar proyectos similares en el futuro.

III: EJECUCIÓN

En esta sección se va a entrar en detalle del proceso de creación, entrenamiento e implementación del modelo de aprendizaje automático utilizado para detectar y traducir LESCO a español. Se da un enfoque especial en el área de manejo y transformación de datos para comprender cómo se pueden analizar y utilizar imágenes como fuente para este tipo de sistemas.

A. Definición del sistema

Antes de poder hablar sobre la recolección de las imágenes y su conversión a datos usables, se debe definir primero cuáles van a ser las tecnologías seleccionadas para proceder con la creación del modelo, esto incluye decisiones sobre el lenguaje de programación, librerías de captura de imágenes, de visión por computadora, de manejo de datos, y de entrenamiento de modelos de aprendizaje automático.

1) Lenguaje de programación

La ciencia de datos ha sido un ámbito de la informática que ha crecido y se ha expandido en los últimos años. De la misma forma, los lenguajes de programación utilizados han ido cambiando y mejorando con el paso de los años hasta llegar a la actualidad.

Actualmente, la mayor parte de las librerías de *machine learning*, utilizadas en el campo, son parte del lenguaje *Python*, sin embargo, en el pasado se han observado otros lenguajes que han sido más utilizados y populares. Desde una vista más general, se pueden basar los resultados en el estudio brindado por la IEEE, donde anualmente hacen una clasificación de los lenguajes de programación más utilizados según su popularidad. Como se puede observar en la figura 15, actualmente Python es el más utilizado desde una perspectiva general.

Principales lenguajes de programación 2023. Fuente IEEE. [1]

2) Librerías utilizadas

Como primer paso, es necesario explicar brevemente lo que es una librería y por qué es importante hacer uso de estas. En ese espíritu, se puede definir una librería como un conjunto o colección de código escrito por un individuo o entidad y que comúnmente puede ser utilizado de forma libre para ejecutar una tarea específica.

Aplicando esto al presente proyecto, las librerías a utilizar varían desde utilidades para el manejo de los datos, como la captura de imágenes y su transformación en datos usables, hasta las librerías que pueden ser utilizadas para la creación, entrenamiento, prueba y uso de un modelo de aprendizaje automático.

Para el paso de captura de las imágenes, se utiliza *OpenCV*, esta es una librería que ayuda a utilizar una imagen existente o una webcam de la máquina para la captura en tiempo real de datos. A pesar de que *OpenCV* ofrece capacidades de detección de objetos, para el propósito de detectar las características y poses del cuerpo humano se va a utilizar la librería *Mediapipe* creada por Google.

Mediapipe es una librería que, entre otras cosas, habilita el reconocimiento simultáneo de múltiples estructuras del cuerpo humano, incluyendo los brazos, manos, y gestos faciales. Esta librería es especialmente útil en el caso de uso del presente proyecto, puesto que va a facilitar el reconocimiento de las señas realizadas en LESCO.

Demostración de librería MediaPipe. Fuente MediaPipe [2]

Con respecto al manejo de los datos, Mediapipe es capaz de capturar puntos de referencia del cuerpo y su posición en tiempo real, sin embargo, estos datos “crudos” no son útiles para entrenar un modelo. Es por este motivo que para el almacenamiento de los datos en un formato entendible por las librerías de aprendizaje automático se va a utilizar la librería *Numpy* la cual permite la creación de matrices y vectores multidimensionales, especialmente útiles en esta aplicación para almacenar los datos complejos obtenidos del proceso de captura de imágenes.

Finalmente, y con respecto al modelo en sí, es necesario hablar sobre los diferentes métodos y algoritmos utilizados en estos modelos. Inicialmente se habla sobre 2 tipos distintos de procesos a ejecutar, el primero, también llamado regresión, equivale al proceso de predecir o determinar un valor, comúnmente continuo o linear, como un precio, edad, salario, entre otros. El segundo proceso se conoce como clasificación, donde un modelo es entrenado con valores “comunes” para una serie de datos, de forma que se pueda proveer la correspondiente etiqueta o valor para datos nuevos y desconocidos, ejemplos de esto pueden ser clasificar entre hombre o mujer, correos spam, o en el caso del presente proyecto: ser capaz de identificar y clasificar señas de LESCO, entre las señas que fueron utilizadas a la hora del entrenamiento.

Para el proyecto se consideran 2 métodos diferentes de realizar la clasificación. El primero es el conocido como “Bosque aleatorio”, este método implementa un tercer método de clasificación llamado “Árbol de decisión”, el cual asemeja la estructura de un árbol, donde cada rama equivale a una pregunta que debe ser contestada utilizando las características obtenidas de los datos, por ejemplo, si se quiere saber si una persona tiene riesgo de tener problemas del corazón se pueden hacer preguntas como la edad, el peso, si es fumador, bebe, entre otras. Todas estas preguntas van a guiar al modelo a tomar una decisión informada sobre el sujeto.

Un bosque aleatorio implementa múltiples instancias de árboles, tomando el resultado de todos los árboles y utilizando un método de voto mayoritario para determinar la mejor respuesta. Este tipo de modelos son muy útiles, pues implementan la simplicidad de los árboles de decisión, pero sin las desventajas de estos. De igual forma, los bosques aleatorios van a requerir de más recursos a la hora de entrenarlos, y también pueden tardar más en obtener una respuesta de clasificación, puesto que deben considerar todas las posibilidades de cada rama de cada árbol.

El segundo método utilizado en el proyecto se conoce como “Red Neuronal”, el cual pretende asemejar la forma en la que funciona el cerebro humano, en una computadora, para procesar datos. Implementa los conceptos de nodos o “neuronas” las cuales están interconectadas entre sí y que crean varias capas, resultando en un proceso de aprendizaje profundo, donde el modelo aprende de sus errores y es capaz de corregirlos, resultando en una mayor precisión del mismo. Los modelos que implementan redes neuronales son muy comunes para ejecutar tareas de visión por computadora, sin embargo, también cuentan con sus propias desventajas, siendo la intensidad computacional la principal, y la necesidad de conjuntos de datos más grandes otra consideración.

Aplicada al proyecto, la implementación de la red neuronal se hace por medio de la librería *Tensorflow*, desarrollada por Google, y que permite la creación de modelos de aprendizaje profundo y redes neuronales. Por medio del API de *Keras*, interfaz en Tensorflow, se puede entrenar un modelo simple utilizando los vectores creados con Numpy anteriormente, utilizando pocas líneas de código, estos modelos pueden ser exportados e implementados en computadoras, celulares, páginas web o la nube, facilitando la implementación masiva de un modelo de este tipo.

3) Implementación

Random Forest

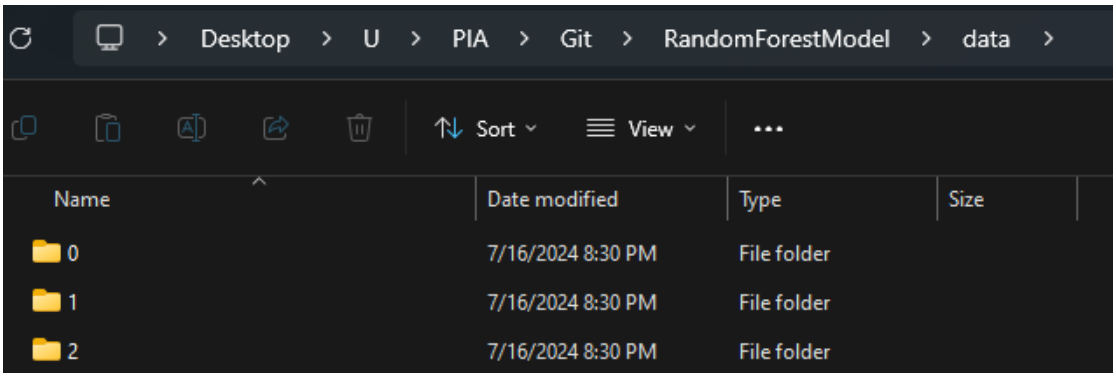
Como fue explicado en la sección anterior, la implementación del modelo fue hecha de manera escalonada y por medio de un método de experimentación para evaluar cuál esquema provee los mejores resultados. Inicialmente se implementa un modelo entrenado por medio del método de clasificación *Random Forest,* este método se destaca por tener un proceso de entrenamiento rápido, y por tener una alta precisión a la hora de clasificar señas estáticas. El proceso de implementación fue el siguiente:

a) El primer paso es la captura de las imágenes, lo cual se hace por medio de la librería OpenCV en Python.
   i. Cabe declarar que la mayor parte del código utilizado en esta sección fue referenciada de la documentación original, donde se detalla un proceso de cómo utilizar la función de captura de imágenes de una forma simple. También se agregaron, proactivamente, comentarios al código para su entendimiento.
   ii. La parte más importante es el siguiente bloque de código, el cual define el bucle que captura las imágenes de las señas.

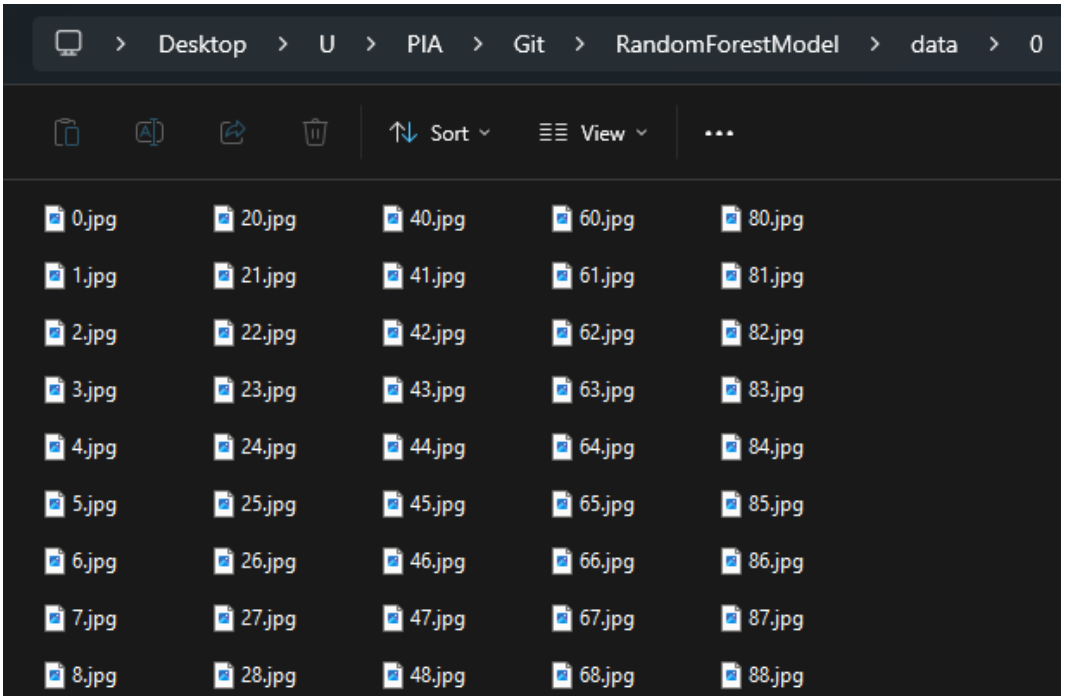
```
# Define counter variable to give all images a unique name
capture_counter = 0
# Capture a new image every 25 seconds
while capture_counter < sign_size:
    ret, frame = cap.read()
    cv2.imshow('frame', frame)
    cv2.waitKey(25)
    # Write to sub directory.
    cv2.imwrite(os.path.join(IMG_DIR, str(i), '{}.jpg'.format(capture_counter)), frame)

    capture_counter += 1
```

Fuente: Imagen capturada por el autor.

   iii. Una vez terminado el proceso se tiene un resultado como el siguiente:



Fuente: Imagen capturada por el autor.



Fuente: Imagen capturada por el autor.

b) El siguiente paso es la conversión de las imágenes a datos entendibles por el modelo que se va a entrenar, para esto se utiliza la librería *Mediapipe,* para guardar en un vector los valores relacionados a la posición de las manos, y otro vector con los “labels” o etiquetas, los cuales son los valores que representan cada seña que se va a traducir. Finalmente, ambos vectores se van a guardar en un archivo serializado utilizando la librería *Pickle.*

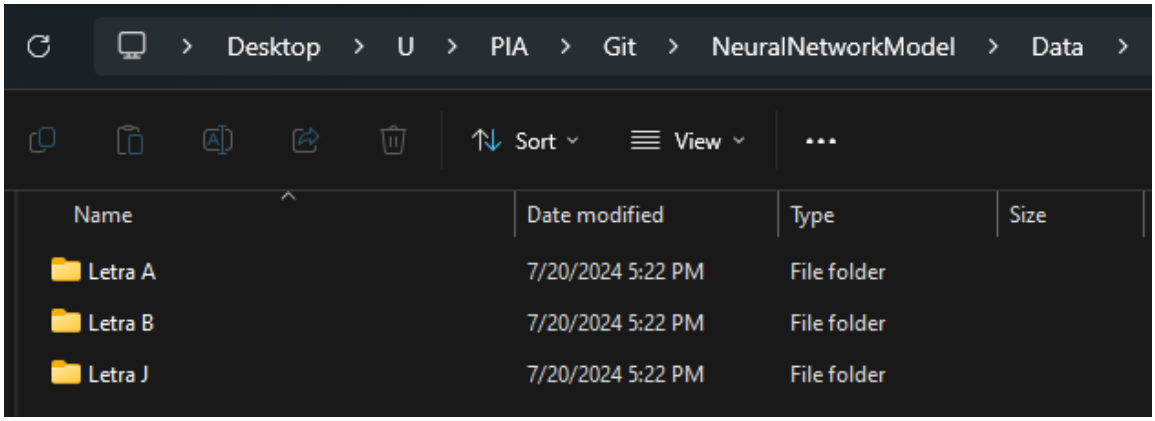
c) Una vez listos los datos en forma de lista y guardados en un archivo serializado, es hora de entrenar el modelo. Este paso se realiza de forma simple, utilizando la librería Scikit-Learn, específicamente el módulo de *RandomForestClassifier*. De la misma forma se realiza una validación rápida de la precisión del modelo recién entrenado, y finalmente se guarda en un archivo para su posterior uso.

d) El paso final del proceso es utilizar el modelo para hacer inferencia, o, en otras palabras, para comenzar a predecir señas utilizando el modelo recién entrenado. Para esto, se va a utilizar nuevamente la librería OpenCV, para poder capturar video por medio de una webcam, y también *Mediapipe* para detectar la posición de las manos del sujeto, los datos obtenidos de Mediapipe, en tiempo real, pueden ser enviados al modelo y el resultado será mostrado en pantalla, de forma clara, utilizando un encapsulador rectangular alrededor de la mano detectada.

Red Neuronal con señas combinadas

El siguiente método implementado fue por medio de una red neuronal simple, además, a diferencia del modelo de Random Forest, este nuevo modelo sí permite el entrenamiento de señas dinámicas, o sea, señas que incluyen algún tipo de movimiento. El entrenamiento de este modelo fue computacionalmente intensivo y tomó más tiempo en finalizar. El proceso de implementación fue el siguiente:

a) El modelo de red neuronal es muy similar al de Random Forest, sin embargo, el proceso de manejo de datos se realiza de forma diferente.
   i. La diferencia principal es que, en lugar de capturar y guardar imágenes en un directorio local, se guardan archivos .npy, los cuales contienen los datos de los puntos de referencia de las manos del sujeto que se está capturando. De esta forma se elimina un paso extra del proceso y se agiliza el mismo.
   ii. La estructura de los directorios de los datos es similar al modelo anterior, pero en este caso el nombre de la carpeta ya incluye la seña a la que se hace referencia.
   iii. El tamaño total de los datos capturados puede ser modificado, aunque inicialmente se realiza con un total de 30 vídeos, los cuales cuentan con 30 *frames* o imágenes.



Fuente: Imagen capturada por el autor.

b) Una vez capturados y guardados los datos, se puede proceder al proceso de separarlos en la sección de entrenamiento y la sección de prueba, dichos datos separados van a ser los utilizados para entrenar el modelo.
c) El modelo en sí es una red neuronal creada en *Tensorflow Keras* de forma secuencial y con múltiples capas. La configuración específica de las capas del modelo no es relevante para el artículo, pero para usuarios técnicos con interés de conocer más, se creó una red con 7 capas, 3 de las cuales son de tipo *Long short-term memory,* 2 de tipo denso, 1 de entrada y finalmente una de salida. Los parámetros de cada capa pueden ser modificados, así como se pueden eliminar o agregar capas en forma de experimentación para obtener diferentes resultados.
   i. Al igual que con el método anterior, el modelo es guardado de forma local en el directorio de trabajo.
   ii. Finalmente, con respecto a las métricas del modelo, se crea una matriz de confusión y se obtiene el puntaje de precisión.
d) Ya teniendo el modelo entrenado, analizado y guardado, se puede comenzar a utilizarlo. Este paso es, de igual forma, bastante parecido al método anterior, con la diferencia de que los datos se manejan de una forma distinta. El programa en tiempo real detecta las manos en la cámara, obtiene sus puntos de referencia, les da el formato correcto utilizando Numpy, y envía dichos datos al modelo previamente entrenado para obtener predicciones de clasificación.
   i. Los resultados se muestran en pantalla al usuario o pueden ser escritos en la terminal donde se ejecuta el programa.

Red Neuronal con señas separadas

El último modelo revisado en este proyecto fue creado por un tercero. El proyecto está disponible en un repositorio público, el cual fue encontrado durante la investigación. El autor, Kazuhito Takahashi, propone un modelo similar a la red neuronal creada en este proyecto, sin embargo, implementa lógica adicional para poder realizar fácilmente una diferenciación entre señas estáticas y señas en movimiento. La implementación del modelo es la siguiente:

a) La captura de las imágenes y recolección de los datos es muy similar a la implementación en los otros modelos, con la diferencia principal de que, en este proyecto se crea una aplicación central, la cual sirve para la captura de los datos, así como para el uso del traductor y modelo.
    i. Los datos son guardados en un archivo CSV almacenado de forma local, el cual contiene los puntos de referencia de las señas detectadas durante la captura, adicionalmente, la primera columna del archivo CSV equivale al número de la seña capturada.
    ii. Para comenzar a capturar señas con la aplicación solamente se debe presionar la tecla “k” en el teclado, y a continuación, presionar en el teclado un número del 0 al 9 para identificar en el CSV a cuál seña equivalen los datos.
    iii. De igual forma, la aplicación permite la captura de señas dinámicas, por ejemplo, la letra Z en LESCO, donde, al levantar el dedo índice, se debe realizar el movimiento respectivo y guardar estos datos en otro CSV por aparte.
    iv. Finalmente, las señas y movimientos guardados en los respectivos archivos CSV se correlacionan con otros 2 archivos CSV que contienen los valores de las señas realizadas, por ejemplo, la seña con índice 0 en el CSV de datos, equivale a la letra A en LESCO.
b) El modelo creado para la aplicación también se implementa de forma simple, separando los datos en grupos de entrenamiento y prueba, y entrenando una red neuronal con la librería TensorFlow y el API Keras
    i. El modelo creado es también analizado y probado en el mismo libro de código.
    ii. Cabe destacar que este proyecto crea 2 modelos separados, uno para señas estáticas y el otro para las señas dinámicas cuando se detectan.

Para brindar valor adicional, el repositorio original fue traducido al español y la nueva versión está disponible para los lectores interesados en la sección de recursos del presente artículo.

4) Resultados

Random Forest

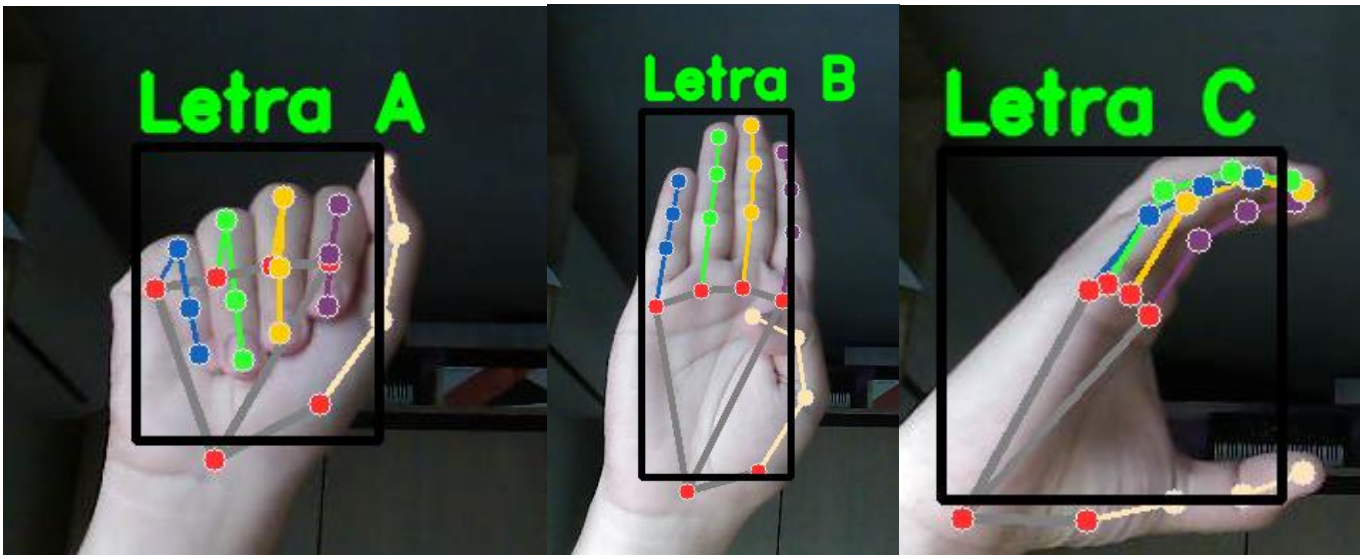
El modelo creado utilizando el método de Random Forest presentó resultados muy buenos al analizarse con una cantidad baja de señas, además considerando que solamente soportaba el análisis de señas estáticas. Los resultados del modelo y su precisión se fueron reduciendo conforme nuevos datos eran agregados para el entrenamiento.

Como se puede ver en la siguiente imagen, el análisis de precisión del modelo, con tan solo 3 señas de entrenamiento, es capaz de obtener un 100% de efectividad en la clasificación de las señas.

```
C:\Users\andre\Desktop\U\PIA\Git\RandomForestModel>python "S.3 - Train_model.py"
100.0% of samples were classified correctly !
```

Fuente: Imagen capturada por el autor.

Sin embargo, al introducir un número mayor de señas, en este caso 10, el puntaje de precisión se ve reducido considerablemente. Esto, junto con el hecho de que el modelo no es capaz de detectar señas dinámicas como la letra J o la letra Z, hizo que este modelo no fuera considerado para seguir con su desarrollo.



Fuente: Imágenes capturadas por el autor.

Red Neuronal con señas combinadas

La red neuronal creada tenía el beneficio de que también era capaz de soportar señas estáticas, así como dinámicas. El proceso de captura de imágenes y entrenamiento fue más largo no solo por la metodología utilizada, sino que, se vio afectado también a la hora de incluir más señas para clasificar con el modelo.

Inicialmente el modelo también presenta una precisión del 100% como se puede observar en la siguiente imagen.

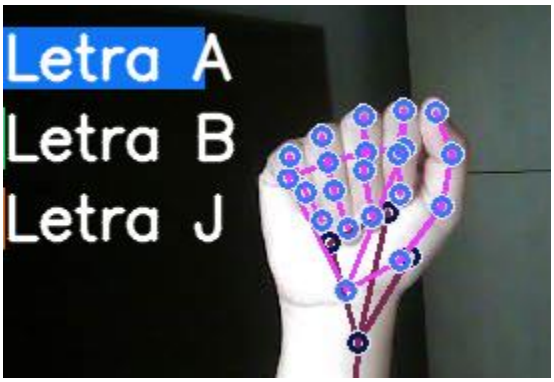
```
 Total params: 711,755 (2.72 MB)
 Trainable params: 237,251 (926.76 KB)
 Non-trainable params: 0 (0.00 B)
 Optimizer params: 474,504 (1.81 MB)
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 208ms/step
100.0% of samples were classified correctly !
```

Fuente: Imagen capturada por el autor.

Pero de la misma forma que el modelo anterior, esta precisión se ve afectada conforme se van agregando nuevos datos al proceso de entrenamiento. Esta precisión significa también que cuando llegue el momento de utilizar el modelo para clasificar en tiempo real, va a ser más común que este se equivoque y produzca valores incorrectos.

Además de las limitaciones ya comentadas, este modelo sufre de un problema adicional, pues si bien soporta ambas señas estáticas como dinámicas, por la forma en la que funciona la clasificación, por medio de ciclos de un tiempo predeterminado, las señas estáticas toman un tiempo para ser detectadas, y las señas

dinámicas deben comenzar justo en el comienzo del nuevo ciclo para que el modelo tenga la mejor posibilidad de clasificar correctamente.



Fuente: Imagen capturada por el autor.

Red Neuronal con señas separadas

Los descubrimientos obtenidos en los 2 modelos anteriores provocaron la necesidad de implementar una metodología diferente, es por esta razón que el modelo creado por Kazuhito fue seleccionado para pruebas. Al implementar procesos y modelos distintos para señas estáticas y dinámicas, evita todos los problemas presentados con respecto al manejo de las diferencias de estas.

Los beneficios obtenidos por la separación de los 2 procesos no significan que otros problemas no van a estar presentes, puesto que este modelo presenta un comportamiento similar a los otros 2 cuando se aumenta la cantidad de señas capturadas para entrenar el modelo, donde la precisión disminuye conforme los datos aumentan. Inicialmente esto puede ser solucionado por medio de procesos de afinación del modelo y sus parámetros, o simplemente con mejorar las condiciones de captura de las señas.

Fuente: Imágenes capturada por el autor.

5) Limitaciones

El proyecto, al ser desarrollado en un ambiente académico, presentó ciertas limitaciones en diferentes áreas que modificaron el flujo del mismo. La principal limitación fue el acceso a datos útiles y veraces que pudieran ser utilizados como material de entrenamiento y validación para los modelos. La limitación de tiempo no permitió recolectar datos de múltiples sujetos, por lo que el material utilizado para el entrenamiento fue conseguido de manera personal, capturando las imágenes y referenciando guías de LESCO de acceso público, así como demostraciones en vídeo de cómo realizar ciertas señas.

Otro punto a destacar, con respecto al material utilizado, es el ambiente donde fue capturado, ya que para este proyecto se utilizó una *webcam* de acceso común e iluminación no ideal. Todos estos parámetros tienen un impacto en la calidad y precisión de los modelos entrenados, y mejorando solamente una de estas limitaciones se podrán observar mejoras en el modelo también.

Otro desafío experimentado durante el desarrollo del proyecto fue con respecto al entrenamiento del modelo, el cual es computacionalmente muy intensivo y requiere de un sistema que pueda soportar la carga en algunos casos por largos plazos de tiempo. Por este motivo, los modelos creados tienen una configuración relativamente básica que pudiera ser manejada por el equipo casero con el cual fue realizado el proyecto.

Adicionalmente, con respecto al material de apoyo, el poco desarrollo de esta tecnología en el país también fue una limitación, no tener proyectos vigentes, los cuales puedan ser referenciados, significó que el desarrollo tuvo que ser creado desde cero con referencias a proyectos externos de otros países, y adaptado manualmente al funcionamiento de LESCO.

Finalmente, con respecto a LESCO, se comprende que la Lengua de Señas Costarricense es compleja y compuesta por diferentes tipos de señas, como lo son las estáticas, dinámicas, e incluso existen señas que se realizan de la misma forma, o muy similar, y que tienen significados completamente diferentes dependiendo del contexto donde se utilicen. Todo esto significa que, para un modelo de inteligencia artificial, el cual no tiene entendimiento del contexto, va a ser muy complejo clasificar correctamente las señas visualizadas.

Por esta razón, para el proyecto se limitó no solamente la cantidad máxima de señas a utilizar al mismo tiempo, sino que también cuáles iban a ser utilizadas, limitándose a las letras del abecedario y los números del 1 al 10, los cuales incluyen señas tanto estáticas como dinámicas.

6) Conclusiones

A pesar de las limitaciones ya mencionadas, y el hecho de desarrollar el proyecto con recursos limitados, se puede concluir que el proyecto fue un éxito, pues se cumplió con el cometido inicial de desarrollar un modelo capaz de clasificar señas en tiempo real, y que en esencia define una base clara para futuros proyectos e investigaciones relacionadas al ámbito de la inclusión por medio de inteligencia artificial.

Como principales conclusiones se destacan los siguientes puntos:

- La creación de modelos de inteligencia artificial, actualmente, es un proceso que, gracias a sus avances tecnológicos, cada día está al alcance de más personas, y que, por medio de las librerías públicas, los modelos pueden ser entrenados por prácticamente cualquier persona con los conocimientos correctos y el equipo necesario. Esto significa que en el futuro cercano se comenzará a ver cada vez más proyectos que utilicen esta tecnología, impulsando un ambiente de avance continuo.
- Se puede afirmar que los datos utilizados en los modelos son la pieza más importante de estos, ya que, por medio de su calidad, cantidad y variedad, se puede entrenar un modelo muy diferente de otro y que dé mejores resultados, haciendo uso del mismo proceso. La captura, almacenamiento, limpieza y manejo de los datos son procesos que nunca se deben ignorar al trabajar con aprendizaje automático.

- Además de los datos, el proyecto debe implementar cierta lógica que le permita clasificar señas tanto estáticas como dinámicas, por este motivo se concluye que la implementación del tercer modelo mencionado en este proyecto, creado por Kazuhito, es actualmente la recomendada, pues fue la que presentó mejores resultados.
- Los resultados de los modelos creados demuestran que la tecnología es capaz de realizar la clasificación de las señas de LESCO correctamente, y que, al realizar procesos de afinación, configuración y experimentación con diferentes parámetros y datos, se pueden obtener resultados capaces de ser implementados en ambientes de producción y que puedan mejorar la calidad de vida de los usuarios que lo utilicen.
- El análisis de las métricas de calidad de los modelos entrenados es un proceso que puede ayudar a entender el comportamiento de estos, incluso antes de ser utilizados en casos reales. Por esta razón también es importante implementar procesos de análisis y comparativa de métricas entre diferentes modelos para poder determinar cuáles parámetros son los que presentan mejores resultados.

7) Recomendaciones

Basado en los resultados, y la experiencia obtenida al desarrollar este proyecto, se presentan las siguientes recomendaciones:

- El proceso de captura de datos debe ser realizado de la forma más limpia y consistente posible, de esta forma se minimizan los impactos en el modelo causados por inconsistencias en los datos, así como problemas con su calidad e iluminación. Al ser un proyecto de alcance público, se puede considerar crear un base de datos que contenga todo el material de entrenamiento del modelo y que otras personas puedan contribuir con la ampliación del mismo.
- Como ya fue mencionado anteriormente, el modelo es tan bueno como sus datos lo sean, por esta razón es importante considerar implementar un proceso de reentrenamiento, en el cual se le pueda dar retroalimentación al modelo sobre sus predicciones y, de esta forma, realizar los cambios necesarios para realizar mejores predicciones en la siguiente iteración.
- LESCO es una lengua que utiliza una gran parte del pueblo costarricense, por este motivo se puede considerar involucrar entidades del gobierno, que puedan brindar el ambiente correcto para implementar este proyecto de forma organizada y brindar el producto final, de manera gratuita a quien lo necesite.
- Una vez creado un modelo capaz de manejar las suficientes señas como para ser útil, se recomienda realizar pruebas de campo en ambientes reales como academias de LESCO, donde los estudiantes puedan probarlo, y de esta forma obtener resultados con situaciones del mundo real.

8) Trabajos futuros

Una de las metas del presente proyecto era poder brindar los resultados del proyecto al público interesado para continuar con su desarrollo, y así poco a poco poder tener una herramienta útil y que pueda brindar valor para sus usuarios. Por esta razón se sugieren los siguientes pasos de mejora para el proyecto:

- Investigar cómo otros gobiernos han implementado proyectos similares para sus habitantes, ya sea que hayan tenido éxito o no, para poder obtener datos reales sobre proyectos con el mismo objetivo.
- Optimización de proceso de diferenciación de señas estáticas y dinámicas, que le permita al sistema determinar cuándo debe utilizar un modelo sobre el otro. Esto debe ser realizado con ayuda de un experto en LESCO, ya que la lengua cuenta con ciertas reglas que deben ser consideradas para poder configurar la detección de las señas.

- Realizar procesos de afinación del modelo en sistemas capaces de soportar la carga computacional del entrenamiento de estos, obteniendo mejores resultados, lo que significa en teoría un mejor modelo.
- Una vez el modelo sea capaz de identificar y clasificar las señas con un alto grado de precisión, se puede considerar crear un modelo nuevo, el cual pueda recibir las predicciones de las señas, y que, con el contexto brindado, pueda crear oraciones que representen correctamente la idea que el usuario de LESCO está tratando de compartir.

9) Recursos

Como parte del artículo, se ofrece el acceso público al código utilizado durante el desarrollo del proyecto para su revisión y modificación. El código está disponible en 2 repositorios de GitHub los cuales pueden ser accesados por medio de los siguientes enlaces:

- RandomForest y Neural Network: https://github.com/AndresZU/PIA02_codigo_modelos/tree/main
- Neural Network modificada: https://github.com/AndresZU/reconocimiento-de-senas-con-mediapipe/tree/main
    - Repositorio original del autor: https://github.com/Kazuhito00/hand-gesture-recognition-using-mediapipe

10) Referencias

[1] IEEE. (2023). IEEE Spectrum. The Top Programming Languages 2023. https://spectrum.ieee.org/the-top-programming-languages-2023
[2] Google. (2024). Mediapipe solutions. https://ai.google.dev/edge/mediapipe/solutions/guide