UNAH-VS

UNIVERSIDAD NACIONAL
AUTÓNOMA DE HONDURAS
EN EL VALLE DE SULA

Departamento de
Ingeniería en Sistemas

IS-701 Inteligencia Artificial

Proyecto: Traductor de lenguaje de señas (informe de proyecto)

Integrantes: Carlos Antonio Gonzales Garza 20192000980
Jeyson Samuel Mejia Hernandez 20172002003

Lugar: San Pedro Sula, Cortes.

Fecha: 07 de diciembre del 2023

-La Felicidad se alcanza cuando lo que uno piensa, lo que uno dice y lo que uno hace, está en Armonía.

Mahatma Gandhi

Puntuación:

Introducción

Basados en la propuesta de proyecto del traductor de lenguaje de señas, a continuación detallaos a través de este informe, la parte operativa y descriptiva del desarrollo de nuestro proyecto, considerando los parámetros de desarrollo solicitados por la asignatura, describiremos el código utilizado canalizados por el framework propuesto, así como algunos ejemplos y eventualidades que se presentaron durante el desarrollo de este proyecto sobre la traducción de lenguaje de señas a través de detecciones de movimientos de mano implementados por modelos de aprendizaje *Tensorflow.*

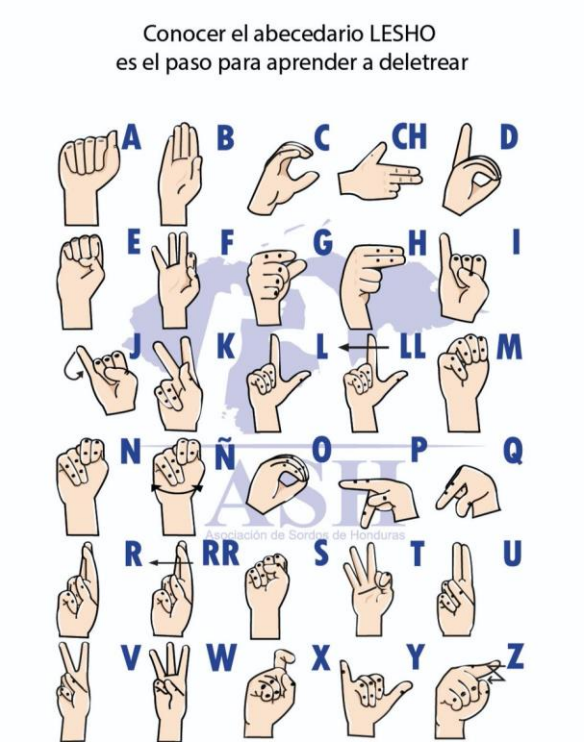
En Honduras, como en muchos otros países, las personas con discapacidad son un grupo en ocasiones invisibilizado en las respuestas y servicios. Sin embargo, un importante porcentaje de la población hondureña presenta una discapacidad. Es por ello que se ve la necesidad de crear herramientas que permitan la atención a este sector.

Las personas con discapacidad como parte de la población también pueden sufrir vulneración de derechos y pueden ser desplazados por violencia, con necesidades especiales de protección.

Ante la falta de señas en Lengua de Señas Hondureñas (LESHO) establecidas para los nuevos conceptos en temas de protección, este glosario pretende establecer señas a estos conceptos que faciliten la comunicación a través de intérpretes de LESHO y entre personas sordas para garantizar el acceso a servicios de protección.

La mayoría de las personas sordas o con deficiencia auditiva conocen y utilizan el lenguaje de señas para comunicarse, a su vez, amigos y familia también deben aprender a utilizar este lenguaje. Sin embargo, cada día más personas aprenden el lenguaje de señas como una habilidad más que desarrollar para la vida y que puede ayudar en gran manera a generar una verdadera inclusión de las personas con esta condición.

Conocer el lenguaje de señas puede ser muy útil en profesiones como son: paramédicos, médicos, enfermeras, policías, profesores y trabajadores sociales, aunque en realidad cualquier persona podría verse en la necesidad de conocer este lenguaje y hacer más sencilla la comunicación con las personas sordas.



Es por eso que, a través de nuestro traductor de lenguaje de señas, estaremos proyectando soluciones a las fronteras entre las personas discapacitadas y el resto de la sociedad, haciendo uso e implementación de las herramientas que ***Tensorflow*** en conjunto con los modelos de aprendizaje nos provee una solución viable para este problema.

Tensorflow es una plataforma de extremo a extremo utilizado para la construcción de aplicaciones de Machine Learning, y también es de código abierto. Tensorflow le permite construir gráficos y estructuras de flujo de datos para definir cómo los datos se mueven a través de un gráfico tomando entradas como una matriz multidimensional llamada Tensor.

Los modelos de Tensorflow son modelos pre-entrenados, y hay cuatro categorías definidas de ellos:

- **Visión:** Analizar características en imágenes y videos.
- **Cuerpo:** Detectar puntos clave y posa en la cara, las manos y el cuerpo con modelos de Media Pipe.
- **Texto:** Habilitar NLP en tu aplicación web usando el poder de BERT y otras arquitecturas codificadores Transformer.
- **Audio:** Clasificar el audio para detectar sonidos.

Todos estos modelos se desglosan en subs y para nuestro caso, estaremos haciendo uso del Modelo de Cuerpo que tiene la detección de poses de mano que necesitamos para detectar los signos de la mano.

Tensorflow

Este modelo utilizó una matriz multidimensional 2D y 3D que le permite predecir los puntos clave de las manos. Ejemplo de un 2D es [[1,2], [3,5], [7,8], [20,44]] y la de un 3D [[1,2,5], [3,5,8], [7,8,6], [20,44,100]].

Esta detección de pose de mano es un modelo del MediaPipe, y nos proporciona dos tipos de modelos que son lite y full. La exactitud de la predicción aumenta de lite a full mientras que la velocidad de inferencia se reduce, es decir, el tiempo de respuesta será más lento a medida que la precisión aumente.

Fingerposse es un clasificador de gestos para los hitos de mano detectados por la detección de poses de mano de *Media Pipe.* También nos permitió añadir nuestros propios gestos de mano, lo que significa que un gesto que significa la letra Z puede significar Hola en los datos de digito.

Para la integración del frame web que nos permitirá hacer uso de la *webcam* para la predicción de movimientos de manos y el lenguaje de señas, vamos a usar a **Vue** para esta ilustración. Empezaremos por mirar el HTML, y luego cubriremos el JavaScript. Nuestra plantilla será una HTML básica que tendrá una etiqueta de vídeo para que podamos mostrar un vídeo después de tener acceso a nuestra webcam.

```
<template>
  <div class="wrapper">
    <video
      ref="videoCam"
      class="peer-video"
      preload="auto"
      autoPlay
      muted
      playsInline
    />
  </div>
</template>
```

Con el “div” antes mostrado y la pestaña de vídeo, daremos acceso al vídeo que se utilizara para la traducción de señas cuando accedemos a la webcam. Seguidamente se escribirá el código JS necesaria para inicializar la webcam.

```
<script  setup>
import { onMounted, ref } from "vue";

const videoCam = ref();
function openCam() {
  let all_mediaDevices = navigator.mediaDevices;

  if (!all_mediaDevices || !all_mediaDevices.getUserMedia) {
    console.log("getUserMedia() not supported.");
    return;
  }
  all_mediaDevices.getUserMedia({
    audio: true,
    video: true,

  })
    .then(function (vidStream) {
      if ("srcObject" in videoCam.value) {
        videoCam.value.srcObject = vidStream;
      } else {
        videoCam.value.src = window.URL.createObjectURL(vidStream);
      }
      videoCam.value.onloadedmetadata = function () {
        videoCam.value.play();
      };
    })
```

```
    .catch(function (e) {
      console.log(e.name + ": " + e.message);
    });
}
onMounted(() => {
  openCam();
});
</script>
```

Tensorflow y detección de manos

Ahora que tenemos nuestra cámara web trabajando, actualizaremos la plantilla y el script para detectar, predecir y mostrar el alfabeto basado en la predicción del signo de la mano para cada una de las letras configuradas (entrenamiento)con **Fingerposse** previamente. Cabe mencionar que este entrenamiento es posible pulirlo de manera que la predicción de detección de poses de dedos sea más exacta, para nuestro proyecto haremos uso de las letras previamente programadas desde el *Media Pipe.*

```
<template>
  <div class="wrapper">
    <video
      ref="videoCam"
      class="peer-video"
      preload="auto"
      autoPlay
      muted
      playsInline
    />
    <div class="alphabet">{{ sign }}</div>
  </div>
</template>
```

El “div” con el nombre de clase *alphabet* mostrará el alfabeto en función de la predicción del signo de la mano, lo que se mencionaba antes, las poses de manos previamente configuradas para la traducción de cada letra.

Posteriormente se crearán dos nuevas funciones, *createDetectionInstance* y *handleSignDetection.* En primer lugar, comenzamos con el *createDetectionInstance* que es parte integral de la detección de señales de mano y luego vamos a presentar handleSignDetection que predice y muestra la letra de la mano detectada.

```
<script setup>
import { onMounted, ref } from "vue";
import * as handPoseDetection from "@tensorflow-models/hand-pose-detection";

let detector;
const videoCam = ref();

function openCam() {
   …
}

const createDetectionInstance = async () => {
  const model = handPoseDetection.SupportedModels.MediaPipeHands;
  const detectorConfig = {
    runtime: "mediapipe",
    modelType: "lite",
    solutionPath: "https://cdn.jsdelivr.net/npm/@mediapipe/hands/",
  } as const;
  detector = await handPoseDetection.createDetector(model, detectorConfig);
};

onMounted(async () => {
  openCam();
  await createDetectionInstance();
});
</script>
```

Para poder detectar poses a mano, necesitamos crear una instancia del detector de la aplicación, y aquí, creamos una función createDetectionInstance que es una función asincrónica. Ahora que hemos creado una instancia para detectar señales de mano, empecemos a detectar la mano. En esa luz, vamos a añadir la nueva función *handleSignDetection.*

```
<script setup>
import { onMounted, ref } from "vue";
import * as handPoseDetection from "@tensorflow-models/hand-pose-detection";

let detector;
const videoCam = ref();

function openCam() {
  …
}

const createDetectionInstance = async () => {
  const model = handPoseDetection.SupportedModels.MediaPipeHands;
  const detectorConfig = {
    runtime: "mediapipe",
    modelType: "lite",
    solutionPath: "https://cdn.jsdelivr.net/npm/@mediapipe/hands/",
  } as const;
  detector = await handPoseDetection.createDetector(model, detectorConfig);
};
```

```
const handleSignDetection = () => {
  if (!videoCam.value || !detector) return;
  setInterval(async () => {
    const hands = await detector.estimateHands(videoCam.value);
    if (hands.length > 0) {
      console.log(hands)
    }
  }, 2000);
};
onMounted(async () => {
  openCam();
  await createDetectionInstance();
  handleSignDetection();
});
</script>
```

El *handleSignDetection* funciona después de crear la instancia de detección. Tenemos un *set Interval* que se ejecuta cada 2, tiempo que también puede ser configurable a través del *Media Pipe,* para comprobar si hay algún signo de mano.

También tenemos una declaración condicional para asegurar que el elemento de vídeo exista, y la instancia de detección se creó en consecuencia.

Entonces, el detector llama a un método *estimateHands,* que trata de predecir la pose de la mano mediante la obtención de puntos clave con valores que están en 2D o 3D (Multidimensional Array).

Ahora que podemos detectar poses de mano, ahora añadiremos dígitos que ayudarán a predecir y mostrar el alfabeto basado en el signo de la mano.

```
<script setup>
import { onMounted, ref } from "vue";
import * as handPoseDetection from "@tensorflow-models/hand-pose-detection";
import * as fp from "fingerpose";
import Handsigns from "@/utils/handsigns";

let detector;
const videoCam = ref();
let sign = ref(null);

function openCam() {
  …
}
```

```
const createDetectionInstance = async () => {
  const model = handPoseDetection.SupportedModels.MediaPipeHands;
  const detectorConfig = {
    runtime: "mediapipe",
    modelType: "lite",
    solutionPath: "https://cdn.jsdelivr.net/npm/@mediapipe/hands/",
  } as const;
  detector = await handPoseDetection.createDetector(model, detectorConfig);
};

 const handleSignDetection = () => {
   if (!videoCam.value || !detector) return;
   setInterval(async () => {
     const hands = await detector.estimateHands(videoCam.value);
     if (hands.length > 0) {
       const GE = new fp.GestureEstimator([
         fp.Gestures.ThumbsUpGesture,
         Handsigns.aSign,
         Handsigns.bSign,
         Handsigns.cSign,
         …
         Handsigns.zSign,
       ]);

       const landmark = hands[0].keypoints3D.map(
         (value) => [
           value.x,
           value.y,
           value.z,
         ]
       );
       const estimatedGestures = await GE.estimate(landmark, 6.5);

       if (estimatedGestures.gestures && estimatedGestures.gestures.length > 0) {
         const confidence = estimatedGestures.gestures.map((p) => p.score);
         const maxConfidence = confidence.indexOf(
           Math.max.apply(undefined, confidence)
         );

         sign.value = estimatedGestures.gestures[maxConfidence].name
       }
     }
   }, 2000);
 };
 onMounted(async () => {
   openCam();
   await createDetectionInstance();
   handleSignDetection();
 });
 </script>
```

Suponiendo que nuestro detector de poses detectara una mano, nuestro código tomara en cuenta el entrenamiento para coincidir con el valor basado en los signos de la mano que creamos con el **Fingerposse.**

El *landmark* es un array 3D proporcionado por el valor clave de la mano. También hay un punto clave, que es un valor 2D, y ambos darán el mismo resultado en base a la posición de mano detectada. Ahora, usando *GE.estimate*, podemos generar un posible gesto que coincidía con el signo, y una puntuación/predicción se asignara a cada gesto a la espera de la cantidad de gesto predicho. Por lo tanto, el gesto con la puntuación más alta se selecciona ya que se estima que es el más cercano al signo de la mano de los signos de mano **Fingerposse** que previamente creamos.

El contenido de las librerías *Handsignsy* que importaremos, mostrara la configuración de cada una de las letras del alfabeto para que este coincida con la posición de manos que configuramos en **Fingerposse.** Este sería el *Handsignsy* configurado para detectar la letra “A”:

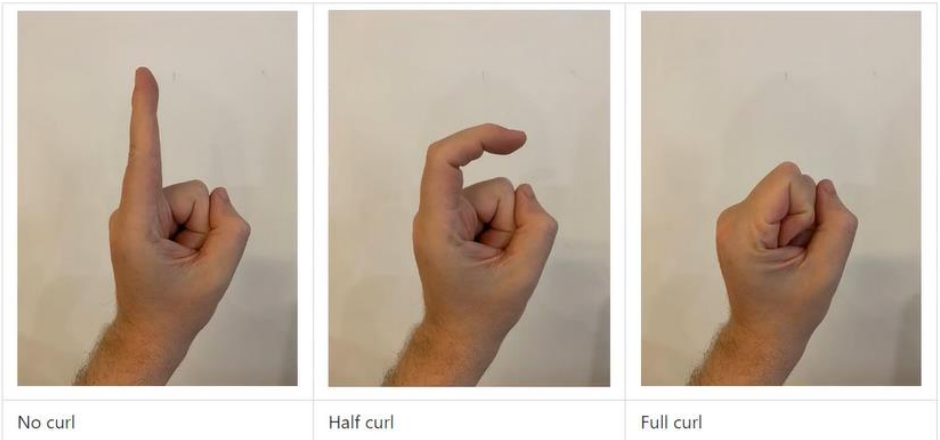
```
import {
  Finger,
  FingerCurl,
  FingerDirection,
  GestureDescription,
} from "fingerpose";

export const aSign = new GestureDescription("A");

//Thumb
aSign.addCurl(Finger.Thumb, FingerCurl.NoCurl, 1.0);
aSign.addDirection(Finger.Index, FingerDirection.DiagonalUpRight, 0.7);
// aSign.addDirection(Finger.Index, FingerDirection.DiagonalUpLeft, 0.70);

//Index
aSign.addCurl(Finger.Index, FingerCurl.FullCurl, 1);
aSign.addDirection(Finger.Index, FingerDirection.VerticalUp, 0.7);
// aSign.addDirection(Finger.Index, FingerDirection.DiagonalUpLeft, 0.70);

//Middle
aSign.addCurl(Finger.Middle, FingerCurl.FullCurl, 1);
aSign.addDirection(Finger.Middle, FingerDirection.VerticalUp, 0.7);
// aSign.addDirection(Finger.Middle, FingerDirection.DiagonalUpLeft, 0.70);

//Ring
aSign.addCurl(Finger.Ring, FingerCurl.FullCurl, 1);
aSign.addDirection(Finger.Ring, FingerDirection.VerticalUp, 0.7);

//Pinky
aSign.addCurl(Finger.Pinky, FingerCurl.FullCurl, 1);
aSign.addDirection(Finger.Pinky, FingerDirection.VerticalUp, 0.7);
```

Estas serían algunas de las librerías importadas para el resto de letras del alfabeto:

Los dedos configurables para **Fingerposse:**

Finger	Name
0	Finger.Thumb
1	Finger.Index
2	Finger.Middle
3	Finger.Ring
4	Finger.Pinky

Estados configurables para cada uno de los dedos:



Se detallan algunos ejemplos del desarrollo ya ejecutado:

Es importante aclarar que para la mejor predicción de poses de manos, es necesario analizar, evaluar y configurar los **Fingerposse** de cada una de las letras del alfabeto de señas hondureño de una manera más profunda, el desarrollo antes explicado y el proyecto presentado, es únicamente un prototipo de lo que posteriormente y mejor trabajado, podría convertirse en una herramienta totalmente operable para las personas incapacitadas que implementa lenguaje de señas para comunicarse.

Conclusión

El lenguaje de señas es una lengua natural de expresión y percepción visual gracias al cual las personas sordas/mudas pueden establecer un canal de comunicación; de manera exitosa y complementando nuestro desarrollo con los códigos antes detallados en este informe, el traductor de lenguaje de señas antes propuesto, ha completado su cometido. Es importante hacer hincapié que el uso de las herramientas que los modelos de aprendizaje nos proporcionan, pueden lograr a solucionar problemas tan complejos y que han acompañado a la humanidad por tanto tiempo sin cruzar las barreras que imposibilitan a personas con este tipo de discapacidad, lograr incluirse en el resto de la sociedad, esperamos nuestro proyecto se implemente en los entornos más provechosos.