UNIVERSIDAD CATÓLICA BOLIVIANA "SAN PABLO"

UNIDAD DE POSGRADO

PROGRAMA: MAESTRIA EN CIENCIA DE DATOS V4

MÓDULO: MACHINE LEARNING

"DESARROLLO DE UN SISTEMA DE RECONOCIMIENTO DE VOCALES EN LENGUA DE SEÑAS MEDIANTE VISIÓN POR ORDENADOR Y DEEP LEARNING PARA FACILITAR LA COMUNICACIÓN INCLUSIVA"

PRESENTADO POR:

María Lucero Yañez Guzmán

Katty Odé Patón

Daniel Olivera Flores

Septiembre 2023

Contents

1. INTRODUCCIÓN

El lenguaje de signos es una herramienta de comunicación fundamental para las personas con discapacidad auditiva. Sin embargo, la barrera lingüística persiste cuando las personas sordas interactúan con quienes no están familiarizados con el lenguaje de signos. Para hacer frente a este desafío, proponemos un proyecto de investigación que utiliza la visión por ordenador y el Deep Learning para crear un sistema de reconocimiento de vocales en lengua de signos. El sistema tiene como objetivo promover una comunicación fluida entre individuos sordos y aquellos que no dominan el lenguaje de signos.

2. OBJETIVOS

2.1. Objetivo General

Crear un sistema de reconocimiento de vocales en lengua de señas utilizando tecnologías de visión por ordenador y Deep Learning. con el propósito de facilitar una comunicación efectiva y fluida entre personas sordas y aquellos que no comprenden el lenguaje de señas.

2.2. Objetivos Específicos

- Realizar una investigación para identificar los signos específicos que representan cada vocal en lenguaje de señas.
- Recopilar un conjunto de datos que incluya signos de alguna región de habla hispana.
- Desarrollar una arquitectura de CNN adecuada que pueda procesar imágenes de gestos de lenguaje de señas y extraer características relevantes para la identificación de vocales.
- Entrenar el modelo utilizando el conjunto de datos recopilado, aplicando técnicas de aprendizaje profundo para mejorar la precisión y robustez del reconocimiento.
- Implementar algoritmos de clasificación que permitan asignar las vocales correspondientes a los gestos identificados.
- Medir la tasa de identificación de vocales correctas y la eficiencia en la comunicación, recopilando comentarios y retroalimentación de los usuarios.

3. DECLARACIÓN DEL PROBLEMA

La comunicación efectiva es esencial para la inclusión y participación de las personas con discapacidades auditivas en la sociedad. El lenguaje de señas se ha convertido en una forma crucial de comunicación para esta comunidad, permitiéndoles expresar sus pensamientos, sentimientos y necesidades de manera significativa. Sin embargo, la barrera lingüística persiste cuando las personas sordas interactúan con aquellos que no conocen el lenguaje de señas.

El reconocimiento preciso y en tiempo real de gestos y signos en el lenguaje de señas es un desafío importante que limita la accesibilidad de las personas sordas a diversos aspectos de la vida cotidiana, como la educación, el empleo y la atención médica. En particular, la identificación de las vocales en el lenguaje de señas representa un componente crítico para la comprensión efectiva de las palabras y oraciones.

Para abordar este problema, proponemos un proyecto de investigación y desarrollo que aprovecha las capacidades de visión artificial y la potencia de un modelo de Deep Learning basado en una Red Neuronal Convolucional (CNN). El objetivo principal de este proyecto es diseñar, implementar y evaluar un sistema de reconocimiento de vocales en lenguaje de señas que permita una comunicación fluida entre las personas sordas y aquellas que no conocen el lenguaje de señas.

El proyecto se enfrentará a diversos desafíos técnicos, como la detección precisa de gestos de las manos y los dedos, la identificación de las formas específicas de las señales que representan las vocales y la adaptación a diferentes estilos y dialectos de lenguaje de señas. Además, se requerirá un conjunto de datos amplio y diverso para el entrenamiento del modelo, lo que implica un esfuerzo significativo en la recopilación y etiquetado de datos.

En última instancia, este proyecto tiene el potencial de mejorar la calidad de vida de las personas sordas, alentando la inclusión y la comunicación efectiva en una sociedad más diversa y accesible. El éxito de esta iniciativa podría tener un impacto significativo en la igualdad de oportunidades para las personas con discapacidades auditivas y en la promoción de la diversidad cultural y lingüística.

4. ANTECEDENTES

4.1. Reconocimiento de gestos y lenguaje de señas

Las tendencias recientes en tecnología relacionada con el reconocimiento de gestos y el lenguaje de señas se han centrado en desarrollar sistemas habilitados para Inteligencia Artificial (IA) que puedan reconocer e interpretar gestos y frases en lenguaje de señas [1]. Estos sistemas utilizan guantes sensores y algoritmos de Aprendizaje Profundo para reconocer gestos individuales discretos, números, letras, palabras y frases [2]. Los sistemas propuestos tienen como objetivo reducir la barrera de comunicación entre las personas con discapacidad auditiva o del habla y aquellos que no conocen el lenguaje de señas. Algunos estudios recientes han explorado el uso del reconocimiento de gestos con las manos en entornos desafiantes [3]. Un proyecto de investigación se enfocó en desarrollar un sistema de reconocimiento de lenguaje de señas para comunicarse con personas con discapacidades, logrando una precisión del 96.3% para las 26 letras del alfabeto en inglés [4].

Se han delineado las mejores prácticas para la investigación en tecnología de lenguaje de señas con el fin de superar errores comunes y obstáculos que han obstaculizado el progreso en este campo. Estas prácticas incluyen la participación de personas sordas en el núcleo de la investigación, incluyendo a investigadores que también son sordos, y reconocer los desafíos que conlleva este tipo de investigación [5].

El lenguaje de señas y las diferentes formas de comunicación basada en señas son prominentes para grandes grupos de la sociedad. Con la llegada de los enfoques de Aprendizaje Profundo, el área de reconocimiento de la lengua de signos se ha enfrentado a una mejora significativa en la precisión en los últimos años [6].

4.2. Tecnologías actuales en visión artificial y Redes Neuronales Convolucionales

Las tendencias recientes en tecnología relacionada con la visión artificial y las redes neuronales convolucionales (CNN, por sus siglas en inglés) se han centrado en desarrollar sistemas que puedan reconocer e interpretar datos visuales, especialmente imágenes y videos. Las CNN son un tipo de algoritmo de aprendizaje profundo diseñado específicamente para el procesamiento y reconocimiento de imágenes. Se utilizan ampliamente para reconocer y clasificar imágenes y objetos. Las CNN se aplican más comúnmente para analizar imágenes comunes, aunque sus aplicaciones se han extendido a varios otros campos, como el procesamiento de lenguaje natural, el análisis de imágenes médicas y el análisis de series de tiempo financieras [7].

Los avances recientes en la tecnología de visión artificial se han centrado en desarrollar sistemas que pueden ver tanto en tierra como bajo el agua [8]. Otros desarrollos recientes incluyen la creación de un dispositivo electrónico neuro mórfico de diseño nuevo que dota a la micro robótica de una visión a colores [9]. Además, los investigadores están explorando el uso de la inteligencia artificial generativa, que tiene la capacidad de crear contenido nuevo y original en lugar de simplemente analizar o actuar sobre datos existentes [10].

4.3. Conjuntos de datos existentes para lenguaje de señas y reconocimiento de gestos

Para fines del presente proyecto se decide utilizar de la plataforma Kaggle, un *"Conjunto de datos de imágenes de gestos en lenguaje de señas, 37 gestos únicos con señales con las manos"* [11]. Obtenido del siguiente enlace:

https://www.kaggle.com/datasets/ahmedkhanak1995/sign-language-gesture-images-dataset

El conjunto de datos consta de 37 gestos de signos con las manos diferentes que incluyen gestos del alfabeto AZ, gestos de números del 0 al 9 y también un gesto para el espacio, que significa cómo las personas sordas o mudas representan el espacio entre dos letras o dos palabras mientras se comunican. El conjunto de datos tiene dos partes, es decir, dos carpetas [11]:

Primera: Datos de imágenes de gestos, que consta de imágenes coloreadas de las manos para diferentes gestos. Cada imagen de gesto tiene un tamaño de 50x50 y está, clasificados en el nombre de carpeta especifico, es decir, las carpetas que contienen letras constan de imágenes de gestos de las letras de la A a la Z y las carpetas con dígitos constan de gestos del 0 al 9, la carpeta '_' consta de imágenes del gesto para el espacio. Cada gesto tiene 1500 imágenes, por lo que en total hay 37 gestos, lo que significa que hay 55,500 imágenes para todos los gestos en la primera carpeta [11].

Segunda: datos preprocesados de imágenes de gestos que tienen la misma cantidad de carpetas y la misma cantidad de imágenes, es decir, 55,500. La diferencia aquí es que estas imágenes son imágenes binarias convertidas de umbral para fines de entrenamiento y prueba. La red neuronal convolucional es muy adecuada para este conjunto de datos con fines de entrenamiento de modelos y predicción de gestos [11].

5. ALCANCE DEL ESTUDIO

El proyecto tendrá los siguientes alcances claramente definidos:

- **Reconocimiento de las Cinco Vocales:** El objetivo principal del proyecto radica en el desarrollo de un modelo de Convolutional Neural Network (CNN) capaz de identificar con precisión las cinco vocales fundamentales del alfabeto en lenguaje de señas: A, E, I, O y U.
- **Conjunto de Datos Exclusivo:** Para garantizar la coherencia y la calidad de los datos, el proyecto se basará exclusivamente en el dataset que poseemos, el cual contiene imágenes representativas de las cinco vocales en lenguaje de señas. Estas imágenes ofrecen una amplia variedad de ejemplos, capturados en diferentes condiciones de iluminación y con fondos diversos, y son realizados por distintas personas.
- **Preprocesamiento de Datos:** Se llevará a cabo un riguroso proceso de preprocesamiento de las imágenes del lenguaje de señas. Esto implica la aplicación de técnicas destinadas a mejorar la calidad de las imágenes, lo que contribuirá a obtener resultados más precisos en el reconocimiento.
- **Arquitectura de CNN Personalizada:** Se diseñará y entrenará una arquitectura de CNN específicamente adaptada a la tarea de reconocimiento de gestos en lenguaje de señas. Este proceso abarcará decisiones fundamentales como la selección de la cantidad de capas, filtros y funciones de activación, con el fin de optimizar el desempeño del modelo.
- **Entrenamiento y Validación Rigurosos:** El conjunto de datos se dividirá cuidadosamente en tres partes: entrenamiento, validación y prueba. Esta división permitirá entrenar el modelo de CNN de manera efectiva y, posteriormente, evaluar su rendimiento de manera exhaustiva. Esto garantizará que el modelo esté listo para enfrentar la tarea de reconocimiento de gestos en lenguaje de señas con precisión y confiabilidad.

6. RECOPILACION DE DATOS

Para este proyecto, hemos seleccionado como fuente de datos el "Conjunto de Datos de Imágenes de Gestos en Lenguaje de Señas" disponible en la página de Kaggle a través del siguiente enlace:

https://www.kaggle.com/datasets/ahmedkhanak1995/sign-language-gesture-images-dataset

Este conjunto de datos abarca 3000 imágenes para cada uno de los 37 caracteres representativos en el lenguaje de señas, que incluyen las letras del alfabeto, los números del 0 al 9 y un espacio en blanco. Es importante destacar que, para este proyecto, nos enfocaremos exclusivamente en las imágenes correspondientes a las cinco vocales: A, E, I, O y U. Los datos están disponibles para su descarga y aprovechamiento, estas imágenes están en color RGB en formato jpg.

Características clave del dataset:

Etiquetas Descriptivas: Cada imagen en el dataset está cuidadosamente etiquetada según la vocal que representa, incluyendo "A," "E," "I," "O," o "U." Estas etiquetas son fundamentales para llevar a cabo un entrenamiento supervisado efectivo de nuestra Convolutional Neural Network (CNN).

Variedad de Sujetos: Las imágenes provienen de diversos individuos que ejecutan los gestos en una amplia gama de contextos y ángulos. Esta diversidad en la fuente de datos contribuirá significativamente a la capacidad del modelo para generalizar de manera efectiva.

Ruido y Variabilidad del Mundo Real: Este dataset incluye ejemplos que reflejan desafíos del mundo real, como gestos con ruido, variaciones de iluminación, fondos diversos y distorsiones. Esto garantiza que nuestra CNN esté preparada para enfrentar situaciones del mundo real.

Resolución y Tamaño Estandarizado: Todas las imágenes se presentan en una resolución uniforme y tienen el mismo tamaño, específicamente 200 x 200 píxeles. Esta uniformidad facilita enormemente el proceso de procesamiento y convolución en nuestra CNN.

Tamaño del Dataset: El conjunto de datos consta de un total de 15,000 imágenes, con 3,000 ejemplos disponibles para cada vocal. En términos de almacenamiento, ocupa un espacio de 218.8 megabytes en disco.

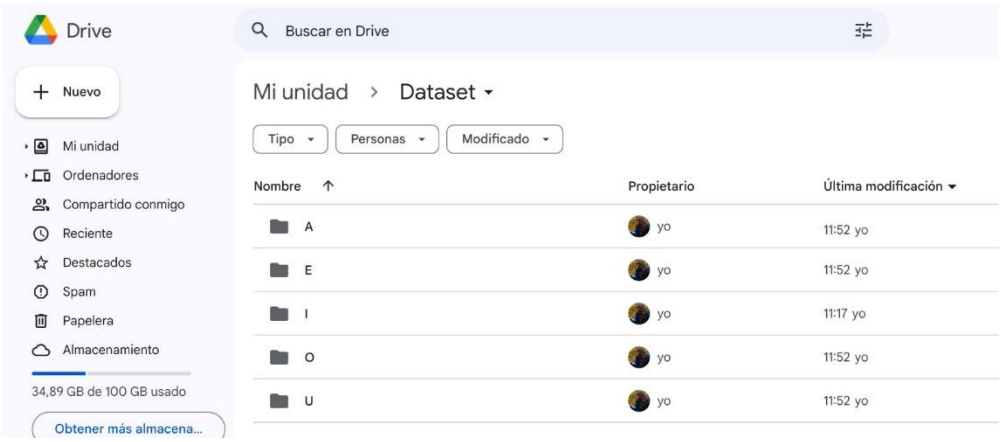

Ilustración 1. Almacenamiento del Dataset

Estas características esenciales hacen que este conjunto de datos sea un recurso valioso y robusto para entrenar y evaluar nuestra red neuronal convolucional, asegurando que esté preparada para abordar una variedad de desafíos del mundo real en el reconocimiento de gestos en lenguaje de señas.

Ejemplo de imágenes de señas de vocales:

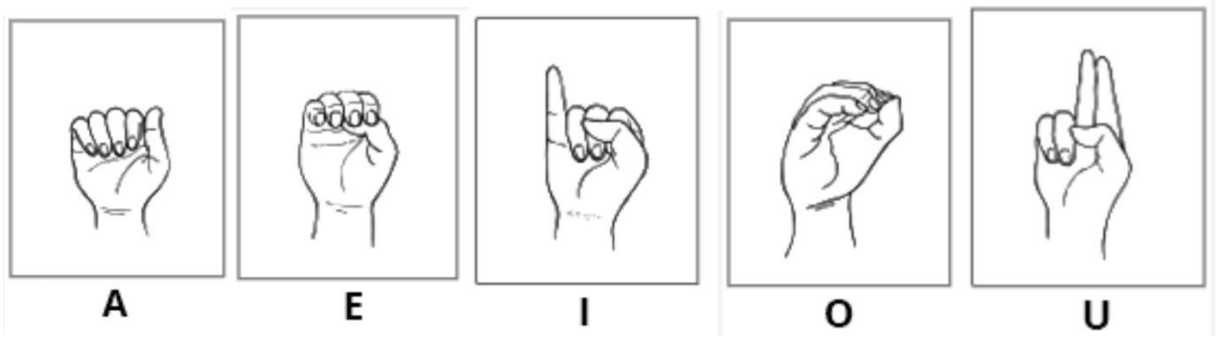

Ilustración 2. Elemplode de vocales en Lenguaje de Señas

7. ENTRENAMIENTO DEL MODELO

7.1. Carga y Preprocesamiento de los datos

A continuación, se presenta el proceso de carga y preprocesamiento de las imágenes de señas para su uso en el modelo de aprendizaje automático. En este punto **X** y **Y** son listas vacías que se utilizarán para almacenar los datos de entrada y salida de la red neuronal. En este caso, X se utilizará para las imágenes de lenguaje de señas preprocesadas, y Y contendrá las etiquetas correspondientes que representan las vocales a las que pertenecen las imágenes. Posteriormente, se crea otra lista vacía llamada *image_paths* para almacenar las rutas relativas de las imágenes en el sistema de archivos.

Tomando en cuenta que las imágenes se almacenan en cinco carpetas con los nombres A, E, I, O, U, nombres que se encuentran almacenados en la lista **folders**; por cada carpeta se construye la ruta completa de la carpeta y se obtiene la lista de nombres de archivos en la carpeta.

```python
X, Y = [], []
image_paths = []

for folder in folders:
    folder_path = path + folder
    images = os.listdir(folder_path)

    for image_filename in images:
        image_path = os.path.join(folder_path, image_filename)
        img = cv2.imread(image_path)
        img = cv2.resize(img, (50, 50))
        img = img / 255.0
        X.append(img)
        Y.append(folder)
        relative_image_path = os.path.relpath(image_path, '/content/drive/My Drive/Dataset')
        image_paths.append(relative_image_path)

X = np.array(X)
Y = np.array(Y)
```

Ilustración 3. Preprocesamiento de los datos

Se leen las imagenes utilizando OpenCV (`cv2.imread`) y se redimensionan a un tamaño de 50x50 píxeles utilizando `cv2.resize`. Posteriormente, se normaliza los valores de píxeles de la imagen dividiendo cada píxel por 255.0, lo que escala los valores de píxeles al rango [0, 1] y se agrega la imagen preprocesada (`img`) a la lista X y la etiqueta de la carpeta actual (`folder`) a la lista Y. Por último, se agrega la ruta de la imagen a la lista `image_paths`.

La normalización de píxeles es una práctica importante en el procesamiento de imágenes y el aprendizaje automático porque mejora la estabilidad, la velocidad de convergencia y la capacidad de generalización de los modelos, al tiempo que facilita el ajuste de hiperparámetros y la compatibilidad con funciones de activación.

Finalmente, las listas X y Y se convierten en matrices NumPy, lo que facilita su uso en modelos de aprendizaje automático. X contiene las imágenes preprocesadas y Y contiene las etiquetas de clase asociadas con esas imágenes. Estas matrices se pueden utilizar como entrada y salida para entrenar un modelo de aprendizaje automático.

```
[[[[0.83137255 0.09411765 0.11372549]
   [0.63921569 0.18039216 0.20392157]
   [0.63921569 0.17254902 0.18823529]
   ...
   [0.6627451  0.18823529 0.22745098]
   [0.7254902  0.23137255 0.25490196]
   [0.84705882 0.25882353 0.27843137]]

  [[0.76470588 0.3372549  0.33333333]
   [0.54901961 0.6745098  0.72156863]
   [0.5254902  0.68235294 0.7254902 ]
   ...
   [0.38431373 0.4627451  0.54509804]
   [0.41568627 0.48235294 0.57254902]
   [0.57254902 0.43529412 0.50196078]]

  [[0.6745098  0.17647059 0.16078431]
   [0.34117647 0.36470588 0.31764706]
   [0.36470588 0.35686275 0.35294118]
   ...
   [0.27843137 0.37254902 0.44705882]
   [0.29411765 0.38039216 0.48235294]
   [0.52941176 0.38431373 0.46666667]]

  ...

  [[0.63921569 0.15294118 0.14901961]
   [0.29019608 0.32156863 0.31764706]
   [0.29803922 0.3254902  0.30980392]
```

```
array(['A', 'A', 'A', ..., 'U', 'U', 'U'], dtype='<U1')
```

Ilustración 4. Lista de imagenes preprocesadas X *Ilustración 5. Lista Y de etiquetas clasificadas de las imágenes de X*

A continuación, se muestra cinco imágenes aleatorias de la lista con sus respectivas etiqueas y rutas con el fin de verificar los resultados del preprocesamiento.

Ilustración 6. Imagenes aleatorias del Dataset

En la lista Y mapeamos las vocales A, E, I, O, U con los números 0, 1, 2, 3, 4 y 5 con el fin de trabajar con datos enteros.

7.2. Implementación de la Red Neuronal Convolucional

Para la implementación de una arquitectura de red neuronal convolucional (CNN) se ha utilizado la biblioteca Keras, que es una interfaz de alto nivel para TensorFlow. Una CNN se utiliza típicamente para tareas de procesamiento de imágenes.

Se ha creado un modelo secuencial en Keras, utilizando una pila lineal de capas de red neuronal, lo que significa que las capas se agregan secuencialmente una tras otra.

i. La primera capa convolucional (`Conv2D`) tiene 32 filtros (o neuronas), cada uno de tamaño 3x3, y utiliza la función de activación ReLU (Rectified Linear Unit). La capa espera imágenes de entrada con un tamaño de (50, 50, 3), lo que significa que se espera que las imágenes tengan una resolución de 50x50 píxeles y 3 canales de color (RGB).

ii. Se agrega una capa de agrupación máxima (`MaxPooling2D`) después de la primera capa convolucional. Esta capa reduce la resolución espacial de las características extraídas por la capa convolucional mediante el muestreo máximo en regiones de 2x2 píxeles. Esto ayuda a reducir la cantidad de parámetros y a aumentar la invariancia a las pequeñas variaciones en la posición de las características en la imagen.

iii. Se agrega una segunda capa convolucional similar a la primera, pero esta vez con 64 filtros en lugar de 32.

iv. Se agrega otra capa de agrupación máxima después de la segunda capa convolucional.

v. Se agrega una tercera capa convolucional similar a las anteriores.

vi. Se agrega una capa de aplanamiento (`Flatten`) para convertir la salida de las capas convolucionales en un vector unidimensional. Esto es necesario antes de conectar la red a capas completamente conectadas.

vii. Se agrega una capa densa (`Dense`) con 64 neuronas y función de activación ReLU.

viii. Finalmente, se agrega una capa densa de salida con 5 neuronas (una para cada vocal: A, E, I, O, U) y función de activación softmax. La función softmax se utiliza comúnmente en problemas de clasificación para calcular las probabilidades de pertenencia a cada clase.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 48, 48, 32)        896

 max_pooling2d (MaxPooling2  (None, 24, 24, 32)        0
 D)

 conv2d_1 (Conv2D)           (None, 22, 22, 64)        18496

 max_pooling2d_1 (MaxPoolin  (None, 11, 11, 64)        0
 g2D)

 conv2d_2 (Conv2D)           (None, 9, 9, 64)          36928

 flatten (Flatten)           (None, 5184)              0

 dense (Dense)               (None, 64)                331840

 dense_1 (Dense)             (None, 5)                 325

=================================================================
Total params: 388485 (1.48 MB)
Trainable params: 388485 (1.48 MB)
Non-trainable params: 0 (0.00 Byte)
```

Ilustración 7. Arquitectura de la CNN

8. VALIDACIÓN Y PRUEBAS

8.1. Conjuntos de validación y prueba

Separamos nuestro dataset en dos para entrenamiento y pruebas.

```
print("Forma de X:", X.shape)
print("Forma de Y:", Y.shape)
```

```
Forma de X: (15159, 50, 50, 3)
Forma de Y: (15159, 5)
```

```
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.25)
```

```
print("Forma de X_train:", X_train.shape)
print("Forma de X_test:", X_test.shape)
print("Forma de y_train:", y_train.shape)
print("Forma de y_test:", y_test.shape)
```

```
Forma de X_train: (11369, 50, 50, 3)
Forma de X_test: (3790, 50, 50, 3)
Forma de y_train: (11369, 5)
Forma de y_test: (3790, 5)
```

Ilustración 8. Creación de los conjuntos de validación y pruebas

8.2. Compilación y Entrenamiento del modelo

Compilamos el modelo de red neuronal en Keras. La función `compile` se utiliza para configurar cómo se llevará a cabo el proceso de entrenamiento del modelo. A continuación, se explican los argumentos utilizados en la función `compile`:

- `optimizer='adam'`: Este argumento especifica el optimizador que se utilizará durante el entrenamiento del modelo. En este caso, se utiliza el optimizador Adam. Adam es un algoritmo de optimización popular que ajusta automáticamente la tasa de aprendizaje durante el entrenamiento para adaptarse a las características de los datos y suele funcionar bien en una variedad de problemas de aprendizaje automático.
- `loss='categorical_crossentropy'`: Este argumento especifica la función de pérdida que se utilizará para medir la discrepancia entre las predicciones del modelo y las etiquetas reales durante el entrenamiento. En este caso, se utiliza la función de pérdida de entropía cruzada categórica (categorical crossentropy). Esta función de pérdida es comúnmente utilizada en problemas de clasificación multiclase cuando las etiquetas están codificadas en formato one-hot (cada muestra pertenece a exactamente una clase). La entropía cruzada categórica mide la diferencia entre las probabilidades predichas por el modelo y las probabilidades reales (one-hot encoded) de las clases.
- `metrics=['accuracy']`: Este argumento especifica las métricas que se utilizarán para evaluar el rendimiento del modelo durante el entrenamiento y la evaluación. En este caso, se utiliza la métrica de precisión (accuracy), que mide la fracción de muestras clasificadas correctamente por el modelo.

```
1 # Compilar el modelo
2 modelo.compile(optimizer='adam',
3                loss='categorical_crossentropy',
4                metrics=['accuracy'])
5
```

Ilustración 9. Compilación del Modelo

Una vez que se ha compilado el modelo con estos argumentos, el modelo está listo para ser entrenado utilizando datos de entrada y etiquetas de salida. Durante el entrenamiento, el optimizador minimizará la función de pérdida especificada (entropía cruzada categórica en este caso) ajustando los pesos y sesgos de la red neuronal para hacer que las predicciones se aproximen mejor a las etiquetas reales. La métrica de precisión se utilizará para evaluar el rendimiento del modelo en el conjunto de entrenamiento y en el conjunto de prueba después de cada época de entrenamiento.

Se realizó el entrenamiento del modelo de red neuronal en Keras, con un número de épocas o iteraciones completas a través de todo el conjunto de datos de entrenamiento que se realizarán durante el proceso. En este caso, se realizarán 5 épocas de entrenamiento, lo que significa que el modelo verá todo el conjunto de entrenamiento cinco veces en total.

```
1
2 # Entrenamiento del modelo con el conjunto de datos de entrenamiento
3 fit= modelo.fit(X_train, y_train,
4           epochs=5,
5           validation_data=(X_test, y_test))
```

Ilustración 10. Entrenamiento del modelo

Se proporciona un conjunto de datos de validación que no se utiliza para entrenar el modelo, pero se utiliza para evaluar su rendimiento después de cada época de entrenamiento. Esto permite monitorear cómo se generaliza el modelo a datos que no ha visto durante el entrenamiento y ayuda a detectar el sobreajuste (overfitting).

```
Epoch 1/5
356/356 [==============================] - 13s 8ms/step - loss: 0.7127 - accuracy: 0.7075 - val_loss: 0.1755 - val_accuracy: 0.9462
Epoch 2/5
356/356 [==============================] - 3s 7ms/step - loss: 0.1157 - accuracy: 0.9623 - val_loss: 0.0756 - val_accuracy: 0.9741
Epoch 3/5
356/356 [==============================] - 2s 7ms/step - loss: 0.0469 - accuracy: 0.9860 - val_loss: 0.0225 - val_accuracy: 0.9934
Epoch 4/5
356/356 [==============================] - 2s 6ms/step - loss: 0.0366 - accuracy: 0.9884 - val_loss: 0.0235 - val_accuracy: 0.9931
Epoch 5/5
356/356 [==============================] - 2s 6ms/step - loss: 0.0364 - accuracy: 0.9891 - val_loss: 0.0212 - val_accuracy: 0.9942
```

Ilustración 11. Iteraciones del modelo

8.3. Evaluación del rendimiento del modelo con datos reales.

Luego de entrenar el modelo se evalúa en el conjunto de prueba (datos de prueba) que se proporcionan como entrada para calcular métricas de rendimiento del modelo. Las métricas de rendimiento incluyen la pérdida (loss) y cualquier métrica adicional especificada durante la compilación del modelo, como la precisión (accuracy). El modelo entrenado en un archivo con el nombre 'modelo_vocales.h5'. El formato de archivo .h5 es un formato comúnmente utilizado para almacenar modelos de Keras. Guardar el modelo es importante porque permite cargar y utilizar el modelo entrenado en el futuro sin necesidad de volver a entrenarlo desde cero.

```
1  #Evaluar el modelo en el conjunto de prueba
2 resultado = modelo.evaluate(X_test, y_test)
3
4 # Guardar el modelo entrenado en un archivo
5 modelo.save('modelo_vocales.h5')
```

```
119/119 [==============================] - 0s 3ms/step - loss: 0.0212 - accuracy: 0.9942
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3000: UserWarning: Y
  saving_api.save_model(
```

Ilustración 12. Precisión del Modelo

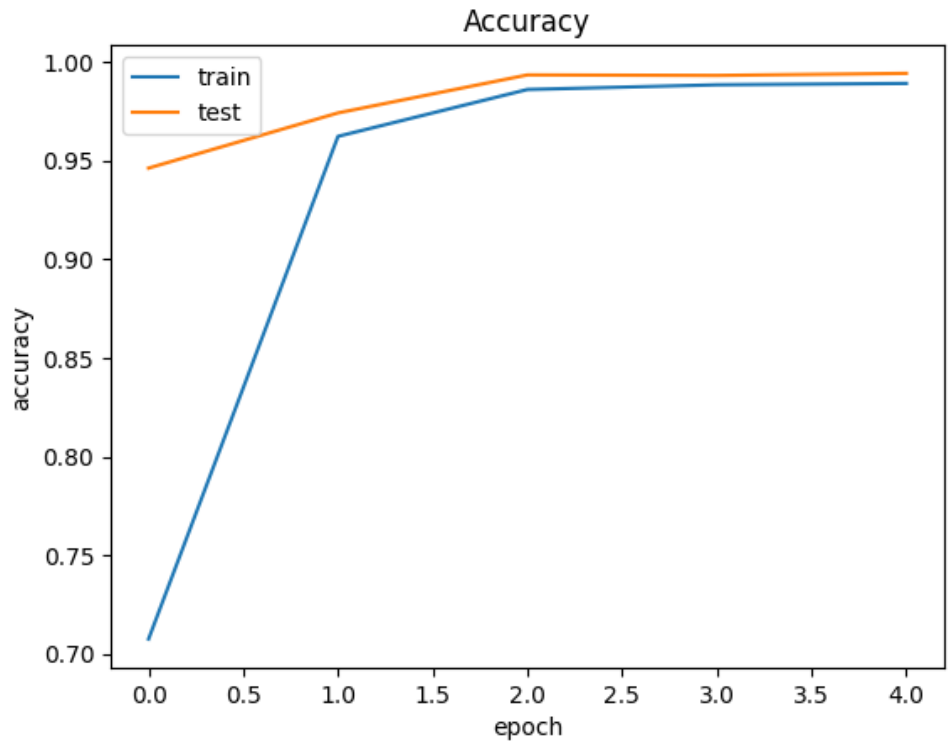

Ilustración 13. Grafica de Precisión por Iteración

9. EVALUACIÓN Y RESULTADOS

9.1. Evaluación de métricas de desempeño.

Utilizamos la función `classification_report` para generar un informe de clasificación que muestra métricas de rendimiento detalladas para nuestro problema de clasificación. Estas métricas incluyen la precisión, el recall, la puntuación F1 y otras métricas relacionadas con la clasificación. El informe proporciona una visión completa de cómo el modelo está funcionando en términos de clasificación en diferentes clases.

El informe de clasificación generado suele tener la siguiente estructura:

- **Precisión**: La precisión mide la proporción de verdaderos positivos (instancias clasificadas correctamente como positivas) entre todas las instancias clasificadas como positivas por el modelo. Es útil cuando se quiere minimizar los falsos positivos.

- **Recall**: El recall mide la proporción de verdaderos positivos entre todas las instancias reales que son positivas. Es útil cuando se quiere minimizar los falsos negativos.

- **F1-Score:** El puntaje F1 es la media armónica de precisión y recall. Es una métrica que equilibra tanto falsos positivos como falsos negativos.

- **Soporte:** El soporte es el número de muestras verdaderas de cada clase en el conjunto de prueba.

- **Promedio**: El informe de clasificación proporciona resultados para cada vocal por separado y calcula promedios para todas las métricas. Puede haber diferentes tipos de promedios, como el promedio ponderado (weighted) o el promedio micro/macro, dependiendo del argumento utilizado en la función.

```
1 print(classification_report(y_pred.round(), y_test))
```

	precision	recall	f1-score	support
0	0.98	1.00	0.99	734
1	1.00	0.98	0.99	760
2	1.00	1.00	1.00	767
3	1.00	1.00	1.00	769
4	1.00	1.00	1.00	760
micro avg	0.99	0.99	0.99	3790
macro avg	0.99	0.99	0.99	3790
weighted avg	0.99	0.99	0.99	3790
samples avg	0.99	0.99	0.99	3790

Ilustración 14. Métricas de desempeño

9.2. Análisis de eficiencia y velocidad de respuesta.

Utilizamos el modelo para realizar predicciones en un conjunto de nuevas imágenes representadas por `nuevas_X`. `nuevas_X` contiene las características o datos de entrada de nuevas imágenes. El método `predict` del modelo toma estas características como entrada y produce predicciones para cada imagen. Las predicciones generalmente son valores continuos que representan las probabilidades de pertenencia a diferentes clases. Una vez que se obtienen las predicciones, se calcula las etiquetas predichas.

```
predicciones = loaded_model.predict(nuevas_X)

# Las predicciones son probabilidades, puedes obtener la etiqueta predicha utilizando argmax
etiquetas_predichas = np.argmax(predicciones, axis=1)

# Ahora tienes las etiquetas predichas para las nuevas imágenes
print(etiquetas_predichas)
```

```
2/2 [==============================] - 0s 17ms/step
```

Ilustración 15. Velocidad de respuesta

Como resultado de estas predicciones obtenemos un tiempo por predicción de 17 milisegundos.

10. CONCLUSIONES Y RECOMENDACIONES

- **Identificación de Signos de Vocales en Lenguaje de Señas:** Se realizó una investigación para identificar y comprender los signos específicos que representan cada vocal en el lenguaje de señas de una región de habla hispana.

- **Recopilación de Conjunto de Datos:** Se recopiló un conjunto de datos que incluye una amplia variedad de signos de lenguaje de señas correspondientes a las vocales. Este conjunto de datos es esencial para el entrenamiento y evaluación del modelo.
- **Desarrollo de Arquitectura de CNN:** Se desarrolló una arquitectura de Red Neuronal Convolucional (CNN) adecuada, diseñada para procesar imágenes de gestos de lenguaje de señas y extraer características relevantes que faciliten la identificación de vocales.
- **Entrenamiento del Modelo:** El modelo de CNN fue entrenado utilizando el conjunto de datos recopilado. Se aplicaron técnicas de aprendizaje profundo para mejorar la precisión y la robustez del sistema de reconocimiento de vocales.
- **Implementación de Algoritmos de Clasificación:** Se implementaron algoritmos de clasificación que permiten asignar las vocales correspondientes a los gestos identificados por el modelo.
- **Medición de la Tasa de Identificación y Eficiencia en la Comunicación:** Se midió la tasa de identificación de vocales correctas logradas por el sistema y su eficiencia en la comunicación. Además, se recopilaron comentarios y retroalimentación de los usuarios para mejorar aún más el sistema.

El presente proyecto ha logrado cumplir los objetivos planteados, sin embargo, se recomienda la implementación de visión artificial para el reconocimiento de señas de personas que utilizan el lenguaje de señas, asimismo es importante el uso del resto de las letras de alfabeto y dígitos del sistema decimal.

11. RECURSOS Y REQUISITOS

Para el desarrollo de este proyecto, se hará uso de las siguientes bibliotecas y herramientas de Python:

- **TensorFlow**: Es una biblioteca de código abierto desarrollada por Google que se utiliza ampliamente para construir y entrenar modelos de aprendizaje automático, incluidas redes neuronales. En el contexto del aprendizaje profundo, TensorFlow se utiliza como backend para Keras, una biblioteca de alto nivel que proporciona una interfaz fácil de usar para construir y entrenar modelos de redes neuronales. Sin embargo, también es posible utilizar otros backends de Keras, como Theano o CNTK. Estos recursos son fundamentales para la construcción y entrenamiento de modelos de redes neuronales.
- **OpenCV** (Open Source Computer Vision Library) es una biblioteca de código abierto que se utiliza para el procesamiento de imágenes y visión por computadora. Proporciona una amplia variedad de herramientas y algoritmos para realizar tareas como detección de objetos, seguimiento de objetos, segmentación de imágenes y más. En el contexto de proyectos que involucran imágenes o video, OpenCV es esencial para realizar operaciones de procesamiento de imágenes y manipulación de datos de imagen.
- **Matplotlib:** es una biblioteca de gráficos en 2D que se utiliza para crear visualizaciones y gráficos en Python. Es especialmente útil cuando se trabaja con datos y se necesita visualizar resultados, como gráficos de pérdida durante el entrenamiento de modelos, mostrar imágenes de entrada o resultados de predicciones, y crear visualizaciones para el análisis de datos. Matplotlib es altamente personalizable y ofrece una variedad de opciones de trazado.

- **NumPy**: es una biblioteca fundamental en Python para el procesamiento numérico y científico de datos. Proporciona estructuras de datos como matrices y funciones para realizar operaciones matemáticas eficientes en ellas. NumPy es especialmente útil en el contexto del aprendizaje automático y la visión por computadora, ya que permite realizar cálculos numéricos de manera eficiente en matrices de datos, como imágenes o características extraídas de imágenes.

12. CRONOGRAMA

- 1 a 16 de septiembre de 2023: Elaboración de Perfil de Proyecto
- 17 de septiembre de 2023: Entrega de Perfil de Proyecto
- 18 a 28 de septiembre de 2023: Desarrollo de Proyecto
- 29 de septiembre 2023: Elaboración de Informes y Documentación

REFERENCIAS

[1] Papastratis I, Chatzikonstantinou C, Konstantinidis D, Dimitropoulos K, Daras P. Artificial Intelligence Technologies for Sign Language. Sensors (Basel). 2021 Aug 30;21(17):5843. doi: 10.3390/s21175843. PMID: 34502733; PMCID: PMC8434597.

[2] Wen, F., Zhang, Z., He, T. *et al.* AI enabled sign language recognition and VR space bidirectional communication using triboelectric smart glove. *Nat Commun* **12**, 5378 (2021). https://doi.org/10.1038/s41467-021-25637-w

[3] Chang V, Eniola RO, Golightly L, Xu QA. An Exploration into Human-Computer Interaction: Hand Gesture Recognition Management in a Challenging Environment. SN Comput Sci. 2023;4(5):441. doi: 10.1007/s42979-023-01751-y. Epub 2023 Jun 12. PMID: 37334142; PMCID: PMC10258789.

[4] Yulius Obi, Kent Samuel Claudio, Vetri Marvel Budiman, Said Achmad, Aditya Kurniawan, Sign language recognition system for communicating to people with disabilities, Procedia Computer Science, Volume 216, 2023, Pages 13-20, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2022.12.106. (https://www.sciencedirect.com/science/article/pii/S1877050922021846)

[5] Fox, N., Woll, B. & Cormier, K. Best practices for sign language technology research. *Univ Access Inf Soc* (2023). https://doi.org/10.1007/s10209-023-01039-1

[6] Razieh Rastgoo, Kourosh Kiani, Sergio Escalera, Sign Language Recognition: A Deep Survey, Expert Systems with Applications, Volume 164, 2021, 113794, ISSN 0957-4174, https://doi.org/10.1016/j.eswa.2020.113794. (https://www.sciencedirect.com/science/article/pii/S095741742030614X)

[7] Venkatesan, R., & Li, B. (2017). *Convolutional neural networks in visual computing: a concise guide*. CRC Press.

[8] MIT News, 4 de agosto de 2022, *First artificial vision system for both land and water* URL: https://news.mit.edu/2022/first-artificial-vision-system-for-both-land-and-water-0804 Fecha de consulta: 27 de septiembre de 2023

[9] Georgia State University. (2022, April 19). Researchers take step toward developing 'electric eye'. *ScienceDaily*. Retrieved September 27, 2023 from www.sciencedaily.com/releases/2022/04/220419092342.htm

[10] AI Accelerator Institute, 5 de Junio de 2023, *Top 5 Computer Vision Trends in 2023,* URL: https://www.aiacceleratorinstitute.com/top-5-computer-vision-trends-in-2023/ Fecha de consulta: 27 de septiembre de 2023

[11] Kaggle, *Sign Language Gesture Images Dataset,* URL: https://www.kaggle.com/datasets/ahmedkhanak1995/sign-language-gesture-images-dataset Fecha de consulta: 27 de septiembre de 2023