

UNIVERSIDAD: LA SALLE

LENGUAJE DE SIGNOS A TEXTO

Trabajo Final de Grado

Tutoras:

Núria Valls Canudas y Elisabet Golobardes Ribé

Autora:

Mar Galiana Fernández

Índice

1. Introducción	2
2. Estado del arte	3
2.1. Fundamentos de la enfermedad	3
2.1.1. Prestación clínica	3
2.1.2. Problemas actuales	4
2.2. Aproximaciones actuales	4
2.2.1. Proyectos realizados	4
2.2.2. Técnicas	5
2.2.3. Algoritmos de inteligencia artificial	5
2.2.4. Datos	5
3. Fundamentos / Aspectos teóricos	6
4. Propuesta	7
5. Experimentación	8
5.1. Dataset	8
5.2. Redes Neuronales	9
5.3. Decision Tree	15
6. Costes del proyecto	20
7. Conclusiones y líneas futuras	21

1. Introducción

2. Estado del arte

El análisis del estado del arte, que se realizará en este proyecto, hace referencia a todas las tecnologías y ayudas que disponen las personas sordomudas a la hora de comunicarse con una persona que no sepa el lenguaje de signos.

Esta comunicación se dificulta, no solo por el hecho de que la mayoría de la población no tiene los conocimientos necesarios para poder comunicarse en un lenguaje no verbal; sino por el hecho de que existen unos 140 lenguajes de signos, con sus respectivos alfabetos y gesto, según la organización Ethnologue [1]. Además, la mayoría de las personas que nacieron sin la capacidad de oír no saben leer.

2.1. Fundamentos de la enfermedad

2.1.1. Prestación clínica

Los trastornos en el habla afectan a las habilidades comunicativas y pueden ser desarrolladas por personas de cualquier edad. Hay diferentes tipos, pero los más comunes son: [2]

- Tartamudeo: este trastorno provoca una interrupción en el flujo del habla, que puede ser a partir de repeticiones, bloqueos y/o prolongaciones. [3]
- Apraxia: en la cual la persona no es capaz de realizar acciones cuando se solicitan aunque los músculos requeridos funcionen. Suele ser debido a daños cerebrales, como: un tumor, demencia, una enfermedad neurodegenerativa, lesiones... [4]
- Disartria: en la cual los músculos que permiten el habla se debilitan o se paralizan impidiendo poder pronunciar palabras. La causa suele ser debido a daños cerebrales, tales como: lesiones, tumores, Parkinson, enfermedad degenerativa... [5]

Los síntomas de este trastorno dependen de la gravedad de este, pero en general suelen: repetir o prolongar sonidos, generar sonidos distorsionados, hablar con voz ronca, reorganizar sílabas, agregar sonidos o sílabas a las palabras...[3]

Las posibles causas son [2][3]: daño cerebral debido a un accidente cerebrovascular, cuerdas vocales dañadas, una enfermedad degenerativa como Huntington o Parkinson, cáncer de garganta o de boca, autismo, síndrome de Down, pérdida de la audición...

Hay ciertas características que pueden incrementar las probabilidades de sufrirlo, como pueden ser: [3]

- Ser un hombre.
- Nacer prematuramente.
- Nacer con un peso inferior a lo recomendado.
- Tener antecedentes familiares con este trastorno.
- Tener problemas en partes del cuerpo tales como: la nariz, el oído o la garganta.

TODO: falta la prestación clínica de los sordos

2.1.2. Problemas actuales

- Utilización de teléfonos móviles.
- Las pocas personas que hablan el lenguaje de los signos.
- COVID: las mascarillas impiden leer los labios.

2.2. Aproximaciones actuales

2.2.1. Proyectos realizados

Como se ha comentado, las personas sordo-mudas se enfrentan a diversos problemas, diariamente, ya que muchos de nuestros hábitos no están habilitados para ellos. Es por esto por lo que se han desarrollado diversas tecnologías para poder ayudar a ser una sociedad más inclusiva.

La primera se denomina *Vocalizer To Mute (V2M)*. Es una aplicación que tiene como objetivo facilitar la comunicación entre las personas no hablantes del lenguaje de signo y los sordomudos, especialmente los niños que lo sufren. A partir de la voz de una persona se analiza el mensaje que está transmitiendo y lo representa en el lenguaje de signos con la ayuda de un avatar en 3D. El reconocimiento de voz utiliza la técnica de MFCC (*Mel Frequency Cepstral Coefficients*), la cual permite obtener los componentes más significativos de la señal de audio para poder identificar el contenido relevante, obviando los componentes que aporta información. Se utiliza HTK (*Hidden Markov Model ToolKit*) para poder convertir las señales de los sensores en frases, a partir del modelo HMM la .

The text output obtained from sensor-based system is converted into speech by using the popular speech synthesis technique of hidden Markov model (HMM). The HMM-based-text-to-speech synthesizer (HTS) was attached to the system for converting the text obtained from hand gestures of people into speech

El kit de herramientas HTK se utiliza para convertir estos vectores acústicos en palabras u oraciones reconocibles mediante el uso de un diccionario de pronunciación y un modelo de lenguaje. La aplicación es capaz de reconocer muestras de habla sordomuda de alfabetos en inglés (A - Z), dígitos en inglés (0 a 9) y 15 oraciones comunes que se usan en la vida cotidiana, es decir, buenos días, hola, buena suerte, gracias. , etc. Proporciona servicio de mensajes tanto para sordos como para personas normales. Los sordomudos pueden utilizar un teclado de lenguaje de señas personalizado para redactar el mensaje. La aplicación también puede convertir el mensaje recibido en lenguaje de señas en texto para una persona normal. La aplicación propuesta también se probó en 15 niños de entre 7 y 13 años. La precisión de la aplicación propuesta es del 97,9

2.2.2. Técnicas

2.2.3. Algoritmos de inteligencia artificial

2.2.4. Datos

3. Fundamentos / Aspectos teóricos

4. Propuesta

Explicación de las diferentes estrategias.

Para la estrategia AccuracyDecisionTree hace falta instalar graphviz para poder mostrar (plot) el modelo de árbol de decisión. En el caso de un mac se tiene que hacer con: brew install graphviz

conda install graphviz python-graphviz

<https://www.mikulskibartosz.name/how-to-plot-the-decision-trees-from-xgboost-classifier/>

<https://drive.google.com/file/d/0B0c0MbnP6Nn-eUNRRkVOOGpkbFk/view?resourcekey=0-nVw3WhovKW5FPvPM5GPHfg>

5. Experimentación

El objetivo de este proyecto ha sido poder interpretar el lenguaje de signos y mostrar su significado en formato texto, a partir de imágenes. Para poder alcanzar este objetivo se han hecho diversas pruebas y experimentos.

5.1. Dataset

En la primera versión se ha utilizado un dataset de la comunidad de datos Kaggle. [6] Consiste en 37 gestos diferentes que representan el abecedario inglés, con las letras de la 'A' a la 'Z', los números del 0 al 9 y el carácter espacio. Cada gesto dispone de 1500 imágenes de 50x50 píxeles. Cada imagen está repetida, ya que hay dos versiones de la misma, una ya preprocesada con la intención de mejorar el entrenamiento y las pruebas de las redes neuronales, y otra versión con las imágenes originales en color. Se han utilizado las dos versiones para poder comparar sus resultados.

A continuación, se muestran diez imágenes que hacen referencia a diez gestos obtenidos del dataset de la segunda versión mencionada.

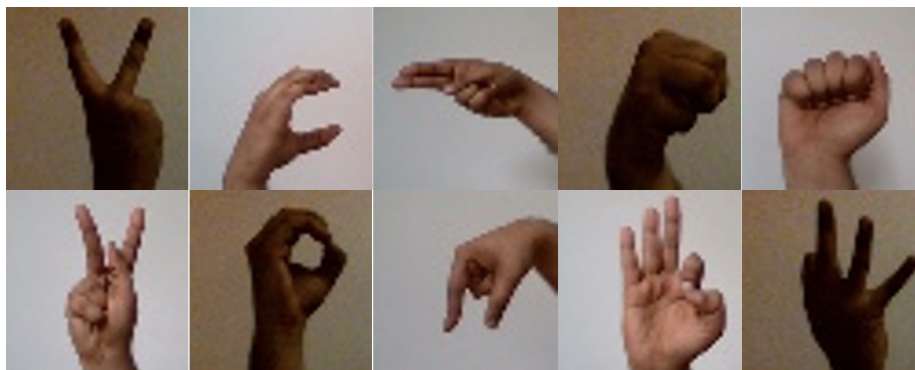


Figura 1: Imágenes obtenidas del dataset *Sign Language Gesture Images Dataset* [6] en color.

Como se puede contemplar en la figura anterior, todas las imágenes han sido tomadas en un fondo de un mismo color, centrándose únicamente en la mano que realiza el gesto. La tonalidad de las imágenes cambia dependiendo del gesto, pero todas las imágenes que hacen referencia al mismo, contienen una tonalidad similar.

A continuación, se muestran otras diez imágenes que hacen referencia a los mismos gestos que en la figura anterior, pero estas han sido obtenidas de la primera versión mencionada.



Figura 2: Imágenes obtenidas del dataset *Sign Language Gesture Images Dataset* [6] preprocesadas con la intención de mejorar el entrenamiento de las redes neuronales.

En total, este dataset dispone de unas 55.500 imágenes, englobando las dos versiones.

5.2. Redes Neuronales

Una vez obtenido el dataset, se decidió empezar utilizando redes neuronales, ya que, como se ha podido apreciar en el estado del arte, es una de las soluciones más recomendadas y utilizadas para el reconocimiento de imágenes.

Se analizaron diferentes tipos de redes neuronales, como se especifica en los fundamentos. Finalmente se decidió empezar por la implementación de una básica y de una convolucional, con el objetivo de poder comparar la precisión de cada una de ellas.

La red neuronal básica se implementó con una estructura de dos capas. Los datos de entrada, que son las imágenes pertenecientes al dataset ya especificado, son previamente procesados, para que sean de una dimensión de 150*150 píxeles. La última capa tienen una dimensión de 38 elementos.

FALTA EXPLICAR BASIC NN

En la figura que hay a continuación, se muestra el resultado del entrenamiento cuando se utiliza el dataset mencionado con las imágenes a color, pero antes de proceder con el entrenamiento son procesadas en blanco y negro, para disminuir el coste de memoria.

```
Model: "sequential"
-----
Layer (type)                Output Shape                Param #
-----
dense (Dense)                (None, 100)                 2250100
-----
dense_1 (Dense)              (None, 38)                  3838
-----
Total params: 2,253,938
Trainable params: 2,253,938
Non-trainable params: 0
-----
Epoch 1/10
304/304 [=====] - 28s 20ms/step - loss: 3.2980 - accuracy: 0.1572 - val_loss: 1.9880 - val_accuracy: 0.6661
Epoch 2/10
304/304 [=====] - 5s 18ms/step - loss: 1.7224 - accuracy: 0.6878 - val_loss: 1.1555 - val_accuracy: 0.8127
Epoch 3/10
304/304 [=====] - 5s 18ms/step - loss: 1.0269 - accuracy: 0.8228 - val_loss: 0.7742 - val_accuracy: 0.8527
Epoch 4/10
304/304 [=====] - 6s 18ms/step - loss: 0.6952 - accuracy: 0.8701 - val_loss: 0.5717 - val_accuracy: 0.8846
Epoch 5/10
304/304 [=====] - 5s 18ms/step - loss: 0.5093 - accuracy: 0.9025 - val_loss: 0.4581 - val_accuracy: 0.8943
Epoch 6/10
304/304 [=====] - 6s 18ms/step - loss: 0.4034 - accuracy: 0.9202 - val_loss: 0.3776 - val_accuracy: 0.9111
Epoch 7/10
304/304 [=====] - 5s 18ms/step - loss: 0.3270 - accuracy: 0.9351 - val_loss: 0.3146 - val_accuracy: 0.9299
Epoch 8/10
304/304 [=====] - 5s 18ms/step - loss: 0.2750 - accuracy: 0.9462 - val_loss: 0.2726 - val_accuracy: 0.9417
Epoch 9/10
304/304 [=====] - 5s 18ms/step - loss: 0.2344 - accuracy: 0.9549 - val_loss: 0.2399 - val_accuracy: 0.9505
Epoch 10/10
304/304 [=====] - 5s 18ms/step - loss: 0.2035 - accuracy: 0.9635 - val_loss: 0.2083 - val_accuracy: 0.9571
```

Figura 3: Resultado de la ejecución de la función *summary* con la red neuronal básica.

La red neuronal convolucional fue implementada a partir de cinco capas. La primera tiene dos dimensiones, los inputs son introducidos es una matriz de 150x150 píxeles. La última capa es de una única dimensión de 38 elementos.

FALTA EXPLICAR CNN

En la figura que se muestra a continuación, contiene el resultado del entrenamiento de la red neuronal convolucional que se acaba de especificar. El dataset utilizado es el que contiene las imágenes a color, pero igual que se ha hecho con la red neuronal básica, las imágenes han sido procesadas en blanco y negro, para disminuir el coste de memoria.

```

Model: "sequential"
-----
Layer (type)                Output Shape                Param #
-----
conv2d (Conv2D)             (None, 148, 148, 25)       250
-----
max_pooling2d (MaxPooling2D) (None, 148, 148, 25)       0
-----
flatten (Flatten)           (None, 547600)             0
-----
dense (Dense)                (None, 100)                54760100
-----
dense_1 (Dense)             (None, 38)                 3838
-----
Total params: 54,764,188
Trainable params: 54,764,188
Non-trainable params: 0
-----
Epoch 1/10
304/304 [=====] - 319s 1s/step - loss: 2.5943 - accuracy: 0.3136 - val_loss: 0.5930 - val_accuracy: 0.8491
Epoch 2/10
304/304 [=====] - 319s 1s/step - loss: 0.4330 - accuracy: 0.8983 - val_loss: 0.2760 - val_accuracy: 0.9010
Epoch 3/10
304/304 [=====] - 325s 1s/step - loss: 0.1851 - accuracy: 0.9567 - val_loss: 0.1389 - val_accuracy: 0.9611
Epoch 4/10
304/304 [=====] - 426s 1s/step - loss: 0.1037 - accuracy: 0.9755 - val_loss: 0.0895 - val_accuracy: 0.9808
Epoch 5/10
304/304 [=====] - 399s 1s/step - loss: 0.0639 - accuracy: 0.9863 - val_loss: 0.0563 - val_accuracy: 0.9880
Epoch 6/10
304/304 [=====] - 369s 1s/step - loss: 0.0375 - accuracy: 0.9934 - val_loss: 0.0423 - val_accuracy: 0.9895
Epoch 7/10
304/304 [=====] - 369s 1s/step - loss: 0.0266 - accuracy: 0.9961 - val_loss: 0.0450 - val_accuracy: 0.9864
Epoch 8/10
304/304 [=====] - 370s 1s/step - loss: 0.0192 - accuracy: 0.9967 - val_loss: 0.0307 - val_accuracy: 0.9940
Epoch 9/10
304/304 [=====] - 363s 1s/step - loss: 0.0157 - accuracy: 0.9972 - val_loss: 0.0171 - val_accuracy: 0.9974
Epoch 10/10
304/304 [=====] - 349s 1s/step - loss: 0.0090 - accuracy: 0.9990 - val_loss: 0.0143 - val_accuracy: 0.9970

```

Figura 4: Resultado de la ejecución de la función *summary* en la red neuronal convolucional.

Se han implementado tres nuevas estrategias para poder entrenar las redes neuronales especificadas. Estas estrategias permiten guardar los modelos que se van utilizando facilitando la ejecución con diferentes datasets y modelos de redes neuronales, no teniendo que ejecutar de nuevo el procesamiento de las imágenes y el entrenamiento a cada cambio que se realice en el código.

El inconveniente de tener el proyecto separado en estrategias es que se genera un aumento en el coste de memoria, ya que se necesita guardar los cambios realizados en archivos para poder utilizarlos en la siguiente ejecución.

Las estrategias implementadas son:

- **SaveDatabase:** tiene como objetivo procesar el dataset, transformándolo al formato adecuado: cambiando el tamaño y el color de cada imagen. Una vez se hayan hecho las modificaciones pertinentes, se guarda el modelo en un archivo denominado pickle, con extensión *.pkl*.

De cada dataset se generarán dos pickles, uno con las imágenes de training y otro con las de testing. Con el dataset con el que estamos trabajando actualmente, en la versión de las imágenes a color, el pickle de testing

tiene un tamaño de 3 GB y el de training de 6,99 GB, 9,99 GB en total.

En la cuadro que se muestra a continuación se puede apreciar la comparación de los tamaños de las dos versiones del dataset cuando se guardan en un directorio, tal y como se descargaron de Kaggle [6], a cuando se guardan en un formato pickle.

	Tamaño directorio	Tamaño pickle
Dataset en color	106,7 MB	9,99 GB
Dataset preprocesado	81,6 MB	9.99 GB

Cuadro 1: Comparativa del tamaño del dataset cuando se guardan las imágenes en un directorio a cuando se utiliza un pickle.

Como se puede apreciar en el cuadro anterior, el archivo pickle tiene un tamaño mayor que el propio dataset, pero aún así es beneficioso ya que su lectura es mucho más rápida y nos permite no tener que procesar los datos en cada ejecución. En futuros experimentos se intentará reducir este tamaño con la intención de poder disminuir el coste de memoria.

FALTA COMPARATIVA TIEMPO DE EJECUCIÓN

- **TrainNeuralNetwork:** esta estrategia tiene como objetivo entrenar la red neuronal y guardar el modelo en un archivo con extensión *h5*. Para poder ejecutarla se necesita especificar el tipo de red neuronal que se quiere utilizar, actualmente hay dos posibilidades: la red neuronal básica y la convolucional. Otro parámetro que también se debe especificar es el pickle con el que se entrenará la red neuronal.

El modelo se guardará en el archivo *h5* con tal de poder utilizarlo en otras estrategias, como la que se explicará a continuación, sin tener que volver a entrenarla para poder trabajar con este modelo.

El tamaño de este archivo depende del tipo de red neuronal utilizada, en este caso el modelo de la básica tiene un tamaño de 27,1 MB, y la convolucional de 657,2 MB, independientemente del pickle utilizado. Esta información se muestra de una forma más gráfica en el siguiente cuadro:

	Tamaño modelo red neuronal básica	Tamaño modelo red neuronal convolucional
Dataset en color	27,1 MB	657,2 MB
Dataset preprocesado	27,1 MB	657,2 MB

Cuadro 2: Tamaño del archivo *h5* por los dos tipos de redes neuronales con los dos datasets que se están utilizando actualmente.

El tiempo de ejecución de esta estrategia varia mucho dependiendo del tipo de red neuronal que se esté entrenando y del pickle seleccionado. La red neuronal convolucional puede llegar a tardar alrededor de una hora más que la básica, como se muestra en la tabla que hay a continuación:

	Tiempo entreno red neuronal básica	Tiempo entreno red neuronal convolucional
Dataset en color	05:54 minutos	55:45 minutos
Dataset preprocesado	03:29 minutos	01:00:16 hora

Cuadro 3: Tiempo de ejecución de la estrategia de *TrainingNeuralNetwork* dependiendo del dataset y del tipo de red neuronal.

- **AccuracyNeuralNetwork:** Una vez entrenada la red neuronal, se habrá creado un archivo *h5* donde se habrá guardado el modelo entrenado. Esta estrategia procesa el modelo guardado e intenta hacer las predicciones con la parte del dataset definida para el testing, una vez tiene el resultado lo analiza para poder comprobar el número de aciertos. El resultado se mostrará por el terminal.

En la siguiente tabla se muestra una comparativa del *accuracy* entre las diferentes versiones del dataset: la original y la preprocesada con las dos redes neuronales implementadas.

	Accuracy red neuronal básica	Accuracy red neuronal convolucional
Dataset en color	72.27 %	34.46 %
Dataset preprocesado	100 %	99.96 %

Cuadro 4: Comparativa del *accuracy* de los dos tipos de redes neuronales dado dos modelos con diferente procesado.

Como se aprecia en la tabla 4, la red neuronal básica tiene una *accuracy* mas elevada en los dos Dataset, con lo que se han deducido dos posibilidades:

- No se están introduciendo suficientes datos en la red neuronal convolucional para poder hacer un buen entrenamiento, ya que este tipo de redes necesitan muchos datos para poder aprovechar todo su potencial.
- La configuración con la que se han implementado la red neuronal convolucional no son las más adecuadas para este dataset.

En cuanto al tiempo de ejecución de esta estrategia, si que se encuentra diferencia entre el tipo de red neuronal, pero su duración no llega al minuto, como se muestra la tabla que hay a continuación:

	Tiempo accuracy red neuronal básica	Tiempo accuracy red neuronal convolucional
Dataset en color	11 segundos	48 segundos
Dataset preprocesado	15 segundos	43 segundos

Cuadro 5: Tiempo de ejecución de la estrategia de *AccuracyNeuralNetwork* dependiendo del dataset y del tipo de red neuronal.

En futuros experimentos se comprobará si las suposiciones que se han mencionado son acertadas o no, con la intención de mejorar la *accuracy* de las dos redes neuronales.

Una vez analizado cada una de estas estrategias, podemos comparar el coste global, tanto de memoria como de tiempo. En la siguiente tabla se muestra la suma de los costes de memoria que han habido en las tres estrategias, dependiendo del dataset y del tipo de red neuronal.

	Coste memoria red neuronal básica	Coste memoria red neuronal convolucional
Dataset en color	10,02 GB	10,65 GB
Dataset preprocesado	10,02 GB	10,65 GB

Cuadro 6: Comparativa del coste de memoria en las tres estrategias entre los dos tipos de redes neuronales, dado dos modelos con diferente procesado.

El coste de memoria ha sido calculado a partir de la suma de los archivos que se generan en cada estrategia. En concreto, la suma del tamaño del pickle y del archivo que contiene el modelo entrenado de la red neuronal. Como conclusión, se puede observar como el dataset no supone una diferenciación en el coste de memoria, ya que los pickles tienen el mismo tamaño con los dos procesados. Lo que provoca la diferencia entre los dos tipos de red neural, es el archivo *h5* que

se genera una vez ha sido entrenada, que tiene un tamaño mayor en la convolucional, como queda reflejado en la tabla 6.

En la siguiente tabla se muestra el coste de tiempo de las tres estrategias, sumando el tiempo de ejecución de cada una de ellas, dependiendo del dataset y de la red neuronal utilizada.

	Coste tiempo red neuronal básica	Coste tiempo red neuronal convolucional
Dataset en color	06:05 minutos	56:33 minutos
Dataset preprocesado	03:44 minutos	01:00:59 hora

Cuadro 7: Comparativa del coste de tiempo en las tres estrategias entre los dos tipos de redes neuronales, dado dos modelos con diferente procesado.

FALTA SUMAR EL TIEMPO DE LOS PICKELS

En la tabla 7 se puede observar como hay una diferencia abismal entre el coste del tiempo de ejecución de la red neuronal básica a la convolucional, prácticamente la diferencia es de una hora. Con el coste de tiempo sucede lo mismo que con el coste de memoria, es indiferente el dataset utilizado. La razón de esto, es que al final hay el mismo número de datos, únicamente cambia el tipo de procesado, el dataset que originalmente es en color, ha sido transformado para poder trabajar en tonalidades grises, con lo que no hay un coste adicional del color.

En cuanto a la conclusión a la implementación de las redes neuronales, se ha observado, en la tabla 4, como la red neuronal básica ha dado mejores resultados que la convolucional, y como el procesado del dataset ha significado una mejora inimaginable en los dos tipos de redes. Esta información se tendrá en cuenta cuando se vaya a utilizar un dataset diferente al actual.

Durante la explicación de cada estrategia se han contemplado diferentes mejoras que se llevarán a cabo en futuras experimentaciones, con la intención de poder obtener el mejor resultado con el mínimo coste posible de las redes neuronales.

5.3. Decision Tree

Segunda experimentación: árbol de decisión –¿accuracy del 100 con los dos datasets: optimizer y gray.

Plot decision tree:


```
In [27]: python3 Src/main.py --accuracyDecisionTree sign_gesture_gray_150x150px

[INFO]: Strategy selected: --accuracyDecisionTree
[INFO]: Arguments entered: sign_gesture_gray_150x150px
/Users/margaliana/Documents/Salle/TFG/SignGestureDetection/Assets/Dataset/Pickels/sign_gesture_gray_150x150px/sign_gesture_gray_150x150px_train.pkl
/Users/margaliana/Documents/Salle/TFG/SignGestureDetection/Assets/Dataset/Pickels/sign_gesture_gray_150x150px/sign_gesture_gray_150x150px_test.pkl
[INFO]: Accuracy is: 100.00%
[INFO]: Strategy executed successfully
[INFO]: Execution finished
```

Figura 5: Resultado de la ejecución de la estrategia *AccuracyDecisionTree* utilizando el dataset no optimizado en blanco y negro.

```
In [25]: python3 Src/main.py --accuracyDecisionTree sign_gesture_optimizer_150x150px

[INFO]: Strategy selected: --accuracyDecisionTree
[INFO]: Arguments entered: sign_gesture_optimizer_150x150px
/Users/margaliana/Documents/Salle/TFG/SignGestureDetection/Assets/Dataset/Pickels/sign_gesture_optimizer_150x150px/sign_gesture_optimizer_150x150px_train.pkl
/Users/margaliana/Documents/Salle/TFG/SignGestureDetection/Assets/Dataset/Pickels/sign_gesture_optimizer_150x150px/sign_gesture_optimizer_150x150px_test.pkl
[INFO]: Accuracy is: 100.00%
[INFO]: Strategy executed successfully
[INFO]: Execution finished
```

Figura 6: Resultado de la ejecución de la estrategia *AccuracyDecisionTree* utilizando el dataset optimizado.

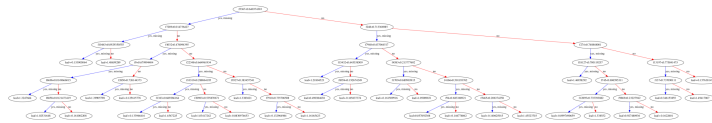


Figura 7: Modelo de árbol de decisión al ejecutar la estrategia *AccuracyDecisionTree* utilizando el dataset no optimizado en blanco y negro.



Figura 8: Modelo de árbol de decisión al ejecutar la estrategia *AccuracyDecisionTree* utilizando el dataset optimizado.

USO DE MAS DE UN DATASET

Se han evaluado varios datasets: <https://www.kaggle.com/kareemalaa74/dataset-asl-test-and-train> <https://www.kaggle.com/grassknotted/asl-alphabet> <https://www.kaggle.com/lucasvieirademi> <https://www.kaggle.com/ammarnassanahajali/american-sign-language-letters>

Finalmente se ha escogido el dataset: <https://www.kaggle.com/kareemalaa74/dataset-asl-test-and-train>

motivos: - También incluía números, - Imágenes hechas desde ángulos diferentes a los del dataset actual - Diferentes fondos, en el dataset actual todos los fondos son blancos, en este se han hecho las imágenes en diferentes sitios.

Decision Tree: Accuracy:

```
[INFO]: Strategy selected: --accuracyDecisionTree
[INFO]: Arguments entered:
    * Decision Tree model file: asl_alphabet_gray_150x150px_model.pickle.dat
[INFO]: Accuracy is: 100.00%
[INFO]: Strategy executed successfully
[INFO]: Execution finished
[INFO]: The duration of the execution has been 00:00:12 [hh:mm:ss]
```

Figura 9: Modelo de árbol de decisión al ejecutar la estrategia *AccuracyDecisionTree* utilizando el nuevo dataset en blanco y negro.

Plot decision tree:

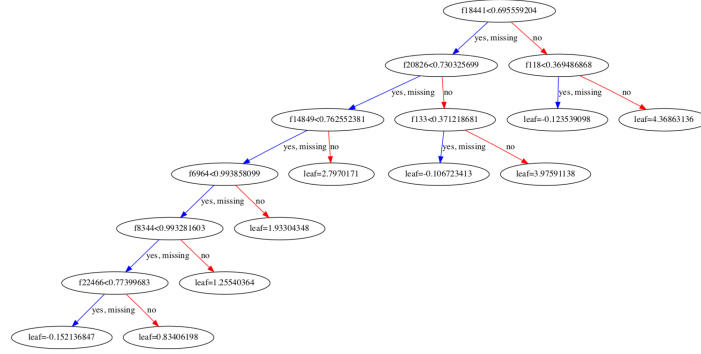


Figura 10: Modelo de árbol de decisión al ejecutar la estrategia *AccuracyDecisionTree* utilizando el nuevo dataset optimizado.

CNN:

TRAIN:

```
In [69]: python3 Src/main.py --trainNeuralNetwork cnn_asl_alphabet_gray_150x150px sign_gesture_gray_150x150px
```

```
[INFO]: Strategy selected: --trainNeuralNetwork
[INFO]: Arguments entered:
* Neural Network type: cnn
* Pickels selected: asl_alphabet_gray_150x150px, sign_gesture_gray_150x150px
Model: "sequential"
```

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 148, 148, 25)	250
max_pooling2d (MaxPooling2D)	(None, 148, 148, 25)	0
flatten (Flatten)	(None, 547600)	0
dense (Dense)	(None, 100)	54760100
dense_1 (Dense)	(None, 39)	3939

```

Total params: 54,764,289
Trainable params: 54,764,289
Non-trainable params: 0

Epoch 1/10
462/462 [=====] - 544s 1s/step - loss: 2.0944 - accuracy: 0.5792 - val_loss: 0.7822 - val_ac
curacy: 0.7888
Epoch 2/10
462/462 [=====] - 510s 1s/step - loss: 0.5132 - accuracy: 0.8591 - val_loss: 0.4022 - val_ac
curacy: 0.8843
Epoch 3/10
462/462 [=====] - 504s 1s/step - loss: 0.2410 - accuracy: 0.9309 - val_loss: 0.2058 - val_ac
curacy: 0.9421
Epoch 4/10
462/462 [=====] - 504s 1s/step - loss: 0.1382 - accuracy: 0.9636 - val_loss: 0.1427 - val_ac
curacy: 0.9604
Epoch 5/10
462/462 [=====] - 501s 1s/step - loss: 0.0799 - accuracy: 0.9809 - val_loss: 0.1115 - val_ac
curacy: 0.9696
Epoch 6/10
462/462 [=====] - 508s 1s/step - loss: 0.0411 - accuracy: 0.9917 - val_loss: 0.0950 - val_ac
curacy: 0.9754
Epoch 7/10
462/462 [=====] - 501s 1s/step - loss: 0.0492 - accuracy: 0.9895 - val_loss: 0.0499 - val_ac
curacy: 0.9902
Epoch 8/10
462/462 [=====] - 509s 1s/step - loss: 0.0162 - accuracy: 0.9979 - val_loss: 0.0430 - val_ac
curacy: 0.9916
Epoch 9/10
462/462 [=====] - 506s 1s/step - loss: 0.0255 - accuracy: 0.9947 - val_loss: 0.0479 - val_ac
curacy: 0.9895
Epoch 10/10
462/462 [=====] - 504s 1s/step - loss: 0.0110 - accuracy: 0.9982 - val_loss: 0.0378 - val_ac
curacy: 0.9930
[INFO]: A new decision tree model has been created with the name of: cnn_asl_alphabet_gray_150x150px-sign_gesture_gra
y_150x150px_model.h5
In the path: /Users/margaliana/Documents/Salle/TFG/SignGestureDetection/Assets/ModelStructures/NeuralNetworkModel/
This is the name that will be needed in the other strategies if you want to work with this model.
[INFO]: Strategy executed successfully
[INFO]: Execution finished
[INFO]: The duration of the execution has been 01:35:02 [hh:mm:ss]
```

Figura 11: Modelo de árbol de decisión al ejecutar la estrategia *TrainNeuralNetwork* utilizando el nuevo dataset en blanco y negro.

Accuracy:

```
In [10]: python3 Src/main.py --accuracyNeuralNetwork cnn cnn_asl_alphabet_gray_150x150px-sign_gesture_gray_150x150px_model.h5
```

```
[INFO]: Strategy selected: --accuracyNeuralNetwork
[INFO]: Arguments entered:
* Neural Network type: cnn
* Neural Network model file: cnn_asl_alphabet_gray_150x150px-sign_gesture_gray_150x150px_model.h5
[INFO]: Accuracy is: 99.30%
[INFO]: Strategy executed successfully
[INFO]: Execution finished
[INFO]: The duration of the execution has been 00:01:31 [hh:mm:ss]
```

Figura 12: Modelo de árbol de decisión al ejecutar la estrategia *AccuracyNeuralNetwork* utilizando el nuevo dataset en blanco y negro.

6. Costes del proyecto

7. Conclusiones y líneas futuras

Referencias

- [1] Alexey Karpov, Irina Kipyatkova, and Milos Zelezny. Automatic technologies for processing spoken sign languages. *Procedia Computer Science*, 81:201–207, 2016.
- [2] Sara Minnis. Speech disorders. <https://www.healthline.com/health/speech-disorders>, September 2019.
- [3] Jamie Eske. What are speech disorders. <https://www.medicalnewstoday.com/articles/324764>, March 2019.
- [4] Mazziotta JC Daroff RB, Jankovic J and Pomeroy SL. Apraxia. <https://medlineplus.gov/spanish/ency/article/007472.htm>, June 2020.
- [5] Mazziotta JC Daroff RB, Jankovic J and Pomeroy SL. Disarthria. <https://medlineplus.gov/spanish/ency/article/007470.htm>, August 2020.
- [6] Ahmed Khan. Sign language gesture images dataset. <https://www.kaggle.com/ahmedkhanak1995/sign-language-gesture-images-dataset>, 2019.

Anexo