Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional

Unidad Zacatenco
Departamento de Computación

Traductor del lenguaje de señas mexicano a texto

tesis que presenta
Ing. Gil Alberto Díaz Balderas

para obtener el grado de
Maestro en Ciencias en Computación

director de la tesis
Dr. J. Guadalupe Rodríguez García

Ciudad de México Diciembre, 2016

II

Resumen

Para muchas personas que cuentan con alguna discapacidad, que les impide comunicarse de manera oral con los demás, el lenguaje de señas es la principal herramienta de comunicación. Es por esta razón que en el presente trabajo se propone realizar el reconocimiento de gestos del lenguaje de señas mexicano.

Hace un par de años se desarrolló un sensor llamado *leap motion* que permite hacer el rastreo tridimensional de la mano completa, dando información específica de puntos relevantes tales como las falanges, puntas de los dedos, tamaño de los dedos, centro de la palma, orientación con respecto a los ejes del sensor, velocidad, etc. Se pretende utilizar *leap motion* como dispositivo de entrada de datos, ya que provee información detallada relevante para llevar a cabo el reconocimiento de los gestos.

También se han utilizado diversos métodos para hacer el reconocimiento de los gestos de los lenguajes de señas tales como modelos ocultos de Markov, máquinas de soporte vectorial, redes neuronales artificiales, etc. Debido a la eficacia en la implementación de las redes neuronales en otro tipo de proyectos de clasificación, se propone utilizar una red neuronal como método de clasificación para realizar el reconocimiento de gestos.

En el presente trabajo se propone una herramienta de software capaz de realizar el reconocimiento de gestos estáticos del lenguaje de señas mexicano, mediante la implementación de una red neuronal y la utilización del sensor *leap motion* como dispositivo de entrada de datos.

Abstract

For many people who have a disability, which prevents them from communicating verbally with others, the sign language is the main communication tool. For this reason, in the present work is proposed to implement a mechanism for hand gesture recognition of mexican sign language.

A couple of years ago, a sensor called *leap motion* was developed, it allows three-dimensional tracking of the entire hand, giving specific information on relevant points such as phalanx, fingertips, direction, center of the palm, orientation with respect to the axes of the sensor, speed, etc. It is intended to use *leap motion* as a data input device, as it provides relevant detailed information to perform gesture recognition.

Several methods have also been used to recognize hand gestures of sign languages such as hidden Markov models, support vector machines, artificial neural networks, and so on. Due to the efficiency in the implementation of neural networks in other types of classification projects, it is proposed to use a neural network as a method of classification to perform the recognition of the gestures.

In the present work, a software tool capable of performing the recognition of static gestures of the mexican sign language is proposed, through the implementation of a neural network and the use of the leap motion sensor as a data input device.

Dedicado a mi hija Hilda.

Agradecimientos

Al Centro de Investigacion y de Estudios Avanzados del Instituto Politecnico Nacional (CINVESTAV), particularmente al Departamento de Computacion por las facilidades brindadas para realizar este posgrado y por el apoyo económico para presentar este proyecto en el Encuentro Nacional en Computación (ENC) 2016, realizado en la ciudad de Chihuahua, Chihuahua.

Al Consejo Nacional de Ciencia y Tecnologia (CONACYT) por el apoyo economico durante dos años.

A mi director de tesis, el Dr. José Guadalupe Rodríguez García, por todo su apoyo, paciencia y consejos.

A mis padres que forjaron los cimientos de mi persona, sin ellos nada de esto hubiera podido ocurrir.

A mi tía Elia Díaz Guadarrama, a mi primo Raúl Escobar y a su familia, por recibirme en su casa cuando llegué a la CDMX, sin su ayuda hubiera sido muy complicado realizar este posgrado.

A mi nueva familia, que son un gran apoyo en todo lo que hago.

A mi familia y amigos, que siempre estuvieron al pendiente de mi.

Por último pero no menos importante a mis compañeros del departamento, por todos los buenos momentos, risas, locuras y un gran etc.

Índice general

Índice de tablas

Índice de figuras

Capítulo 1

Introducción

En México en el año 2010 se censaron más de 400,000 personas con alguna discapacidad para hablar o comunicarse de acuerdo con el Instituto Nacional de Estadística y Geografía (INEGI)[1]. La cantidad de personas con dichas discapacidades dista mucho de disminuir, más bien se acrecenta. Es por ello que deseamos proporcionar una herramienta que ayude a estas personas con sus tareas diarias y permita mejores condiciones de equidad y por consiguiente una mejor calidad de vida.

El uso de la tecnología para ayudar a personas con capacidades diferentes ha sido un tema recurrente en los últimos años, ahora con la incorporación de nuevos dispositivos a la vida cotidiana, se han abierto nuevas oportunidades para generar herramientas que ayuden a dichas personas en sus tareas diarias.

Para muchas personas que cuentan con alguna discapacidad, que les impide comunicarse de manera oral con los demás, el lenguaje de señas es la principal herramienta de comunicación. El principal problema es que pocas personas, que no tienen dicha discapacidad, saben utilizar el lenguaje de señas. Otro problema es que muchas personas además de no poder comunicarse de manera oral con otros, tampoco tienen conocimiento de cómo leer y escribir [3].

El lenguaje señas no es universal, así como existen diferentes lenguajes orales, también existen diversos lenguajes de señas. A pesar de la creencia de

[1]Instituto Nacional de Estadística y Geografía, http://www.inegi.org.mx, Octubre 2015.

muchas personas, los lenguajes de señas no son compatibles entre sí [3]. Por dicha razón las personas que viven en nuestro país utilizan el lenguaje de señas mexicano.

Desde hace algunos años se han desarrollado aplicaciones que traducen lenguaje de señas a texto y a voz, mediante la utilización de diversos dispositivos que ayudan al reconocimiento de gestos [5]. También se han utilizado diversos métodos para hacer la extracción de datos de los dispositivos de entrada y para hacer el reconocimiento de los gestos de los lenguajes de señas [7].

Hace un par de años se desarrolló un sensor llamado *leap motion* que permite hacer el rastreo tridimensional de la mano completa, dando información específica de puntos relevantes tales como las falanges, puntas de los dedos, tamaño de los dedos, centro de la palma, orientación con respecto a los ejes del sensor, velocidad, etc [3]. Lo cual ha permitido su uso en la manipulación de robots [1, 2, 3, 4], control de la pc [5, 6], juegos [7, 8, 9, 10], reconocimiento de lenguaje de señas [11, 12], aplicaciones de realidad virtual [13, 14, 15, 16], entre otros.

1.1. Planteamiento del problema

El problema a resolver se centra en como obtener los gestos del lenguaje mexicano de señas mediante el uso del dispositivo llamado *leap motion*. Para ello primero se debe analizar y clasificar la información obtenida por el dispositivo de entrada. Si la información obtenida no es suficiente para hacer el reconocimiento del gesto, se debe calcular nueva información que brinde una mejor descripción de las características de las manos, tal como ángulos entre falanges, distancia entre dedos, etc.

Una vez obtenida toda la información requerida, se debe hacer el procesamiento de la misma, mediante la utilización de alguna técnica de reconocimiento de patrones, para determinar qué seña está siendo captada por el sensor. Una vez reconocido el gesto, se debe generar la salida del sistema, en este caso texto.

Por último se debe hacer un análisis de los resultados obtenidos por la herramienta propuesta, acerca de su eficiencia y de su rendimiento. Además de hacer una comparativa con proyectos similares.

En la actualidad no existe una herramienta que traduzca del lenguaje mexicano de señas a texto o voz mediante la utilización de *leap motion* como dispositivo de entrada, es por ello que se optó por desarrollar el presente trabajo.

1.2. Hipótesis

Mediante el uso de *leap motion* como dispositivo de entrada y la utilización de una red neuronal profunda, es posible hacer la traducción del lenguaje mexicano de señas a texto.

1.3. Objetivos generales y específicos del proyecto

1.3.1. Objetivo General

Proporcionar un traductor de señas a texto que permita la interacción de una persona con limitaciones para comunicarse con su entorno mediante la utilización de *leap motion*, como dispositivo de entrada de datos.

1.3.2. Objetivos Específicos

1. Estudiar el modo de captura de datos proporcionados el sensor *leap motion*.
2. Definir los datos que serán útiles para la detección de la forma de las manos y sus posibles movimientos.
3. Generar nueva información, a partir de los datos obtenidos por *leap motion*, que ayude con la tarea de identificar la forma de las manos.
4. Identificar la forma de las manos utilizando una red neuronal profunda.
5. Traducir la formas de la manos a texto.
6. Validar la respuesta de le herramienta mediante pruebas con personas experimentadas en el uso del lenguaje mexicano de señas.

1.4. Metodología

1. Capturar datos del dispositivo de entrada, en este caso *leap motion*.

2. Procesar los datos de entrada para reconocer el patrón capturado.
3. Generar la respuesta en texto, correspondiente al patrón identificado.
4. Verificar que la respuesta de le herramienta sea congruente al patrón capturado.
5. Realizar pruebas de funcionamiento en el laboratorio.
6. Implementar mejoras a deficiencias encontradas en las pruebas.
7. Probar la herramienta con personas experimentadas en el uso del lenguaje mexicano de señas.
8. Identificar y corregir fallas.

Capítulo 2

Estado del arte

En la actualidad existen diversos trabajos en relación con la captura de señas. El rápido avance tecnológico ha ofrecido nuevos sensores con mejores capacidades y mejor precisión para capturar las señas. En el presente trabajo nos enfocamos a los trabajos relacionados con el reconocimiento de gestos manuales.

2.1. Reconocimiento de gestos con *leap motion*

2.1.1. Utilizando redes neuronales

A. S. Elons y Menna Ahmed [17], proponen un sistema donde se utilizan los datos generados por *leap motion* para entrenar un red neuronal de tipo perceptrón multicapa, la cual consta de 3 capas, la capa de entrada, la capa oculta y la capa de salida. La capa de entrada representa los datos de entrada, obtenidos por el sensor *leap motion*, que para efectos del estudio, se utilizó una lista de los 10 dedos de las manos con sus respectivas direcciones.

La capa oculta es la responsable del proceso de aprendizaje. Por último la capa de salida es igual al número de gestos que se utilizaron para probar el sistema.

Las pruebas realizadas en este trabajo se centraron sobre 50 señas del lenguaje arábigo de señas, las cuales fueron obtenidas de 4 personas diferentes. El conjunto de señas realizadas por 2 personas fueron utilizadas como el conjunto para entrenar el perceptrón multicapa y el conjunto de señas de las 2 personas restantes fueron utilizadas para hacer las pruebas de reconocimiento.

La implementación del perceptrón multicapa se hizo de la siguiente manera: la capa de entrada consta de 10 neuronas (por que se obtiene un vector de características por cada dedo de las 2 manos), la capa oculta consta de 30 neuronas y la capa de salida consta de 50 neuronas, el grado de conocimiento es de 0.001 y el factor momento es 0.5. La estrucura de la arquitectura planteada se puede observar en la Figura 2.1.

Ya que *leap motion* captura las posiciones y movimientos de los dedos, para proba el sistema se tomaron en cuenta 2 tipos de conjuntos de características diferentes, un conjunto contiene la posición de los dedos y el otro conjunto contiene las distancias entre los dedos.

El sistema obtiene un mayor índice de reconocimiento de gestos utilizando la distancia entre los dedos para entrenar el perceptrón multicapa, ya que las distancias entre los dedos proporcionan características invariantes a la posición de la mano con respecto al sensor.

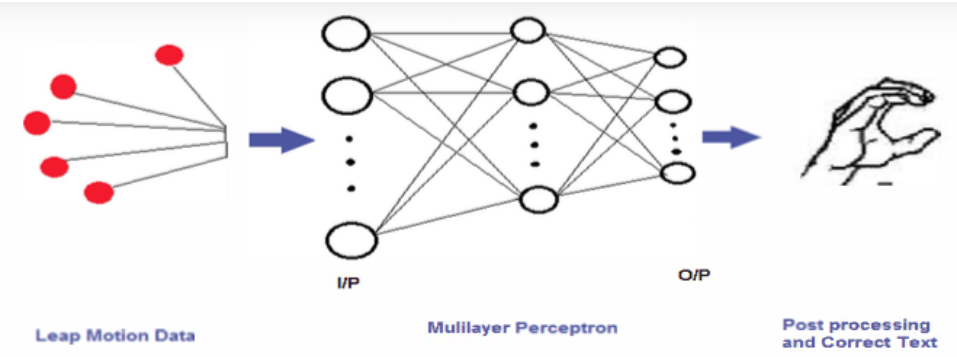

Figura 2.1 Estructura de la red neuronal propuesta en el proyecto.

M. Mohandes y S. Aliyu [18], desarrollaron un sistema para reconocer las 28 señas del alfabeto arábigo, las imágenes de las señas utilizadas en este proyecto se mustran en la Figura 2.2. Dichas señas son estáticas y se ejecutan con una sola mano. Sólo utilizaron datos obtenidos por el sensor para generar sus vectores de características. El dispositivo de entrada utilizado fue el sensor *leap motion*.

Figura 2.2 Lenguaje de señas arábigo.

Para entrenar y probar el sistema utilizaron 10 muestras de cada seña y cada muestra consta de 10 cuadros, haciendo un total de 100 cuadros por cada seña. En total se tuvieron 2800 cuadros que fueron importados a MATLAB para

Para realizar la clasificación de los gestos, se utilizaron 2 técnicas diferentes, una es el clasificador bayesiano y la otra es una red neuronal de tipo perceptrón multicapa entrenada con retropropagación.

En el caso del clasificador bayesiano la efectividad para reconocer los gestos fue de 98.3 % y para el perceptrón multicapa la efectividad fue de 99.1 %. Los problemas encontrados para no alcanzar mejores porcentajes de efectividad se debieron a que en ocasiones los dedos no son visibles en su totalidad por la camara del sensor.

Filomena Soares y Carlos Moreira [12] describen un estudio preliminar y desarrollo de un videojuego destinado a aprender el alfabeto en lengua de signos portuguesa, a través de gestos. El sensor *leap motion* transforma el gesto en datos y lo transmite al navegador a través de la API para dicho sensor escrita en Javascript.

Una vez obtenidos los datos del sensor, se utiliza el marco de trabajo *leap*

trainer[1] para analizar los datos y para aprender gestos utilizando su sistema de reconocimiento que implementa el reconocimiento de gestos basado en una red neuronal. El sistema de reconocimiento convierte un gesto en una matriz, que luego se comparará con las matrices de gestos grabados de un conjunto de entrenamiento. Las redes neuronales necesitan que los datos tengan una dimensión uniforme, esto es garantizado por *brain.js*, una biblioteca implementada en *leap trainer*, que trunca todas las matrices de datos de entrenamiento a la longitud de la más corta.

El número de iteraciones se establece en 20,000. El número de nodos depende del número de capas. Las capas se definen en función del número de dedos "activo" de la mano y su posición. Así, la red neuronal es ajustable con cada gesto. El sistema fue probado con cinco gestos diferentes. Se tarda alrededor de un segundo para identificar una letra con una precisión alrededor del 95 %.

2.1.2. Utilizando otras técnicas de reconocimiento

Ching-Hua Chuan y Eric Regina [19], desarrollaron un sistema piloto basado en 26 señas del lenguaje americano de señas, donde utilizan diversos datos de entrada proporcionados por el sensor *leap motion*, los cuales son: fuerza de agarre (determina si la mano está abierta o cerrada), vector normal, velocidad y posición de la palma, fuerza de atenazar (qué tan abierto está cada dedo con respecto al pulgar de la misma mano), y de cada dedo obtienen su dirección y su tamaño, así como la posición y velocidad de cada una de las falanges.

Utilizando aprendizaje de máquina se derivaron nuevos datos a partir de los obtenidos por el sensor. Los nuevos datos son: distancia promedio, que es calculada como la suma de la distancia entre la punta del dedo en tramas adyacentes, propagación promedio de la palma, que es estimada basado en la distancia entre las puntas de los dedos adyacentes y la tri-propagación promedio, que es el área entre 2 puntas de dedos adyacentes y el punto medio de las posiciones de los metacarpianos.

Por cada dedo se derivaron 4 características las cuales son: distancia ex-

[1]LEAP trainer, http://leap.quitebeyond.de/leap-trainer-gesture-learning-recognition-framework/, Octubre 2016.

tendida (la máxima distancia de todos los puntos del dedo desde el centro de la palma), proyección inclinación-punta (es la proyección del vector inclinación-punta sobre el vector normal de la palma), ordenX (es el orden del dedo sobre el plano x-z con respecto los demás dedos), ángulo (es el ángulo entre las direcciones de los dedos en el plano x-z).

Para el reconocimiento del alfabeto utilizaron el método k-vecino más cercano con diferentes valores de k y máquina de soporte vectorial con diferentes funciones núcleo. Para las pruebas se obtuvieron 4 conjuntos de señas que fueron realizados por 2 personas, una de ellas sordomuda.

Los resultados utilizando el método k-vecino más cercano muestran el mayor porcentaje de reconocimiento en promedio de 72.78 % con un valor de k=7. Los resultados varian significativamente entre los 4 conjuntos de prueba, además los mejores grados de reconocimiento de obtienen con valores de k entre 1 y 150, a partir de valores mayores a 150 para k, el grado de reconocimiento decae notablemente.

Los mejores resultados con la máquina de soporte vectorial se obtienen al implementar la función Gaussiana de base radial como función de núcleo, alcanzando un 79.83 % de efectividad.

Además se realizó un análisis de las señas que no fueron totalmente reconocidas por los métodos anteriormente descritos, donde se señala que las fallas en la clasificación se deben a problemas relacionados en la captura de las señas por parte del sensor *leap motion*, ya que las señas que fueron clasificadas erróneamente tienen mucha similitud con otras señas y las diferencias no fueron percibidas al momento de ser capturadas por el sensor.

Giulio Marin y Fabio Dominio [20], utilizaron *leap motion* y *kinect* para hacer el reconocimiento de gestos estáticos del lenguaje de señas americano. Con *leap motion*, utilizaron la posición de los dedos, la posición de la palma y la orientacón de la mano como conjunto de datos obtenidos directamente de *leap motion*. Además de los datos que se extraen del sensor, ellos calcularon nuevos conjuntos de datos los cuales son: ángulo de cada dedo (el ángulo se mide con respecto a la orientación de la mano), distancia entre dedos (se mide la distancia entre la punta del dedo y el centro de la palma) y la elevación de los dedos (corresponde a la distancia entre la punta del dedo y el plano de la región de la palma).

Con respecto a *Kinect* se extrajeron dos conjuntos de datos, uno que representa las distancias entre los puntos de la mano y el centro de la palma, el cual es obtenido mediante un conjunto de histogramas. El otro conjunto de datos representa la curvatura del contorno de la mano, que es obtenido por un descriptor basado en un operador integral multi-escala.

Todos los conjuntos de datos se resumen en 5 vectores de características útiles para el reconocimiento de gestos. Para dicha tarea utilizaron un clasificador de tipo máquina de soporte vectorial multi-clase, basado en el enfoque uno contra uno, con una función no lineal gaussiana de base radial como función de núcleo. El diagrama de la arquitectura aquí descrita se puede observar en la Figura 2.3.

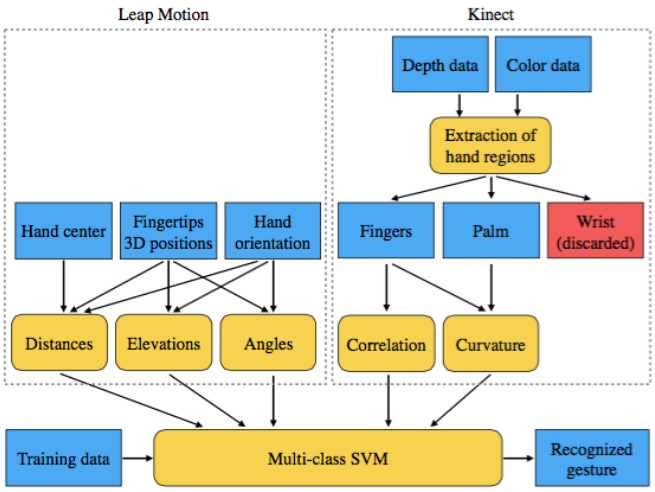

Figura 2.3 Diagrama de la arquitectura propuesta.

Para las pruebas se utilizó un conjunto de 10 gestos, los datos se tomaron de 14 personas diferentes. Cada persona repitió 10 veces cada gesto para tener un total de 1400 diferentes muestras de datos

Realizaron experimentos utilizando los dispositivos por separado y también utilizando la concatenación de los conjuntos de datos extraídos de cada dispositivo. En el caso de *leap motion*, utilizando sólo la distancia entre dedos, obtuvieron un reconocimiento del 76 %, utilizando los ángulos de los dedos un 74.2 %, utilizando la elevación de los dedos un 73 % y al utilizar las 3 características obtuvieron un 81 % de efectividad.

En el caso de *Kinect* utilizando las distancias entre los puntos de la mano y la palma se obtuvo un 65 % de efectividad. Para el caso de la curvatura del contorno de la mano se obtuvo un 87.3 % de efectividad. Utilizando los dos se obtuvo un 89.7 %. Al utilizar todos los datos de los dos sensores se obtuvo un 91.3 % de efectividad.

Yanmei Chen y Zeyu Ding [21], realizaron un trabajo para el reconomiento rápido de gestos del lenguaje arábigo de señas. Como dispositivo de entrada de datos utilizaron *leap motion*. Primero hicieron el muestreo de los gestos y extrajeron las características de los gestos para formar vectores de características. Además realizaron un preprocesamiento de las características, ya que el clasificador sólo admite vectores de características del mismo tamaño.

Para la clasificación de los gestos utilizaron una máquina de soporte vectorial y modelos ocultos de Markov. Como uno de los objetivos del reconocimiento rápido de gestos es reconocer el gesto antes de que éste sea completado por el usuario, se realizaron pruebas con 50 %, 70 %, 80 % y 100 % del total de las muestras tomadas del gesto. La arquitectura propuesta se puede observar en la Figura 2.4

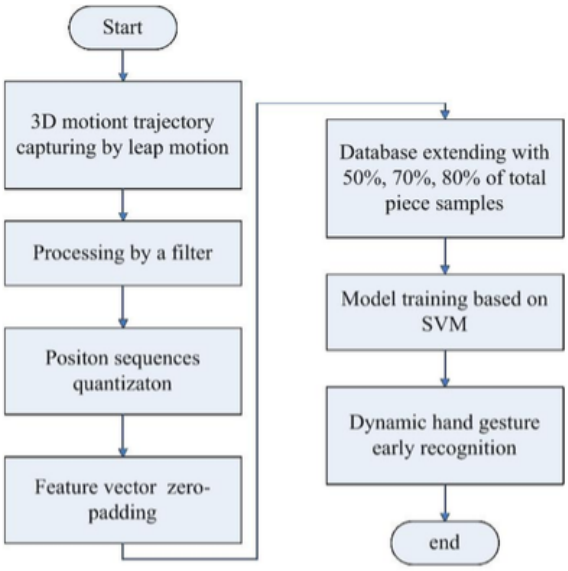

Figura 2.4 Diagrama del proceso de identificación de señas.

Los resultados obtenidos reflejaron que la herramienta utilizando máquina de soporte vectorial obtiene índices de reconocimiento superiores al 90 % y además realiza el reconocimiento de los gestos en tiempos menores a medio segundo, siendo estos resultados muy superiores a los obtenidos utilizando modelo oculto de Markov, sobre todo en el tiempo de respuesta.

Poonam Chavan y Tushar Ghorpade [22] proponen un sistema que implica el reconocimiento del lenguaje de señas indio usando el sensor de movimiento *leap motion*. Todos los atributos relacionados con los dedos y palma que pueden ser obtenidos con el API de *leap motion* se almacenan en una base de datos. Después de obtener los datos del sensor se utiliza un sistema de mapeo de los valores de los atributos obtenidos, de acuerdo con el resultado calculado, cada muestra capturada por el sensor se etiqueta como gesto o muestra sin gestos.

Cada muestra que no contiene gestos es eliminada, las muestras que si contienen un gesto, son procesadas por un clasificador de bosque aleatorio. Una vez entrenado el clasificador, éste se alimentará con datos de muestras individuales.

Sobre la base de mps (muestras por segundo) se calculará un promedio estadístico. Este promedio se utilizará para reducir aún más la desviación del clasificador. Por ejemplo, durante el uso del lenguaje de señas, un signo puede mantenerse durante 3 segundos y si el mps es 13, entonces vamos a adquirir 39 muestras para un signo. Se creará una lista de 39 muestras y se alimentará al clasificador una a una para determinar el signo. El signo será entonces el que tenga mayor promedio. Una vez determinado el signo será mostrado en una pantalla el texto correspondiente al signo. El sistema puede ser observado en la Figura 2.5.

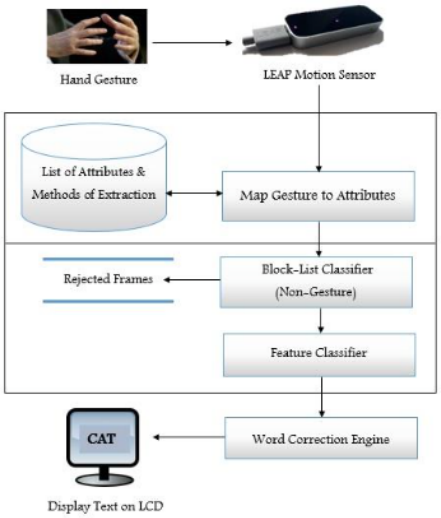

Figura 2.5 Sistema de traducción propuesto.

Jayash Kumar y R. Gupta [23] en su trabajo proponen rastrear y reconocer gestos que representan números que van de 0 a 9 con la ayuda del dispositivo *leap motion*. Las características relevantes a los dígitos numéricos fueron derivadas y se aplicó el método de comparación de plantillas geométricas para reconocer dígitos que van de 0 a 9. *leap motion* redujo la complejidad de identificar el objeto apuntador, ya que es administrado internamente por su API.

El método propuesto consta de 3 fases: La primer fase es la detección de gestos, se refiere a la tarea de localizar patrones en un flujo continuo de señales de entrada, es decir, localizar el punto de inicio y el punto final de un patrón de gesto significativo, se considera como una de las tareas más difíciles en el reconocimiento de los gestos [24].

La segunda fase es la identificación características y registro de gestos, después de identificar el punto inicial del gesto, se registran gestos numéricos. Para grabar estos gestos, las muestras individuales se extraen del gesto dinámico. Para cada trama, se extraen características relacionadas con dispositivos apuntadores (mano y dedo) que incluyen posición, velocidad, posición de la punta, normal, longitud, etc. Los gestos continuos se registran fotograma a fotograma hasta que se identifica el gesto numérico final.

Por último la tecera fase es el entrenamiento y clasificación, una vez que se graba el gesto y se extraen las características correspondientes de mano, dedo y palma, se pasa a través de la fase de entrenamiento, donde las características se guardan en almacenamiento temporal. Con el fin de clasificar un gesto numeral aleatorio, las características extraídas se comparan con las características disponibles en el almacenamiento temporal del modelo y, por consiguiente, el gesto se clasifica. En este trabajo, se ha utilizado el método Geometric Template Matching (GeTeM) para el entrenamiento y reconocimiento de gestos dinámicos.

Para fines de entrenamiento se consideraron 50 muestras de gestos totales (5 gestos por numero). Las pruebas se realizaron utilizando 500 gestos al azar (50 gestos por numero) y se observó una tasa de clasificación promedio de 70.2 %. Un diagrama de la arquitectura implementada puede ser observada en la Figura 2.6

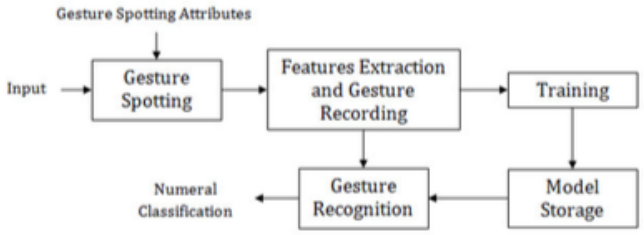

Figura 2.6 Diagrama de la arquitectura propuesta.

Kai-Yin Fok y Nuwan Ganganath [25] proponen un sistema de reconocimiento de gestos del lenguage americano de señas, que está conformado por tres niveles: sensores, fusión y aplicación (ver Figura 2.7).

En el nivel del sensor, los sensores que se usan en el sistema devuelven los datos al centro de fusión sin comunicarse entre sí. En el nivel de fusión, los conjuntos de datos primero se alinean y luego se fusionan en un único conjunto de datos utilizando un algoritmo de fusión sensor-a-sensor. En el nivel de aplicación, los datos fusionados se mapean a un gesto usando modelos ocultos de Markov entrenados con h estados.

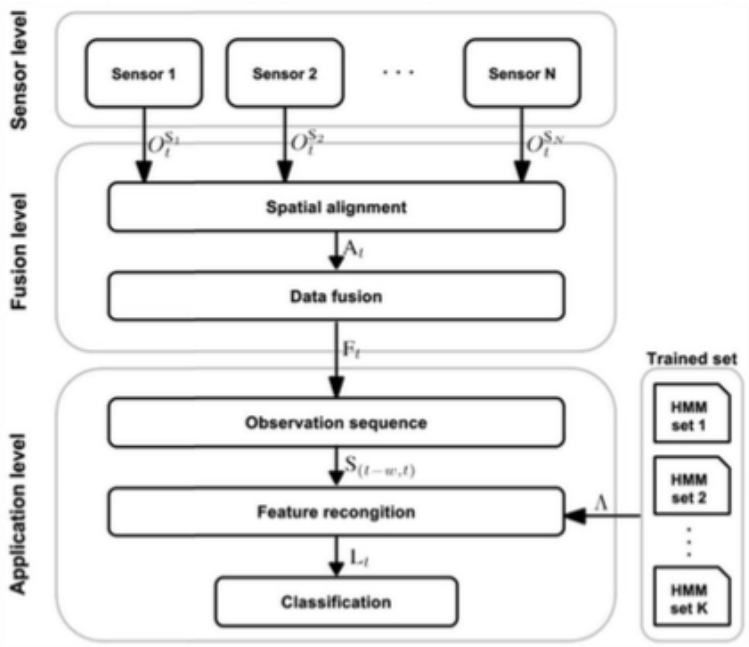

Figura 2.7 Arquitectura de tres niveles.

Con el fin de analizar y evaluar el sistema propuesto, una aplicación se construye utilizando lenguaje de programación Java. Un total de 17720 muestras fueron capturadas, mismas que representan gestos de dígitos de 0 a 9 en el lenguaje amerciano de señas, los datos se recogieron de 8 sujetos. De cada seña, la mitad de las muestras se utilizaron para fines de entrenamiento, mientras que la otra mitad se utilizaró para evaluar el sistema propuesto. Los resultados muestran que utilizando un sensor *leap motion* se obtiene un 84.68 % de efectividad, al utilizar 2 sensores *leap motion* se alcanza un 93.14 % de efectividad.

2.2. Reconocimiento de gestos con kinect

S. Aliyu y M. Mohandes [26] desarrollaron un sistema para reconocer 20 palabras del lenguage arábigo de señas, utilizaron el dispositivo Microsoft Kinect (MK) como dispositivo de entrada de datos, el sistema propuesto consta de las siguientes etapas: el dispositivo MK, recopilación de datos, segmentación para aislar puntos de interés, extracción de características y clasificación.

En la etapa de adquisición de datos se obtuvieron videos RGB y de profundidad, el sensor Kinect que fue programado para capturar 30 fotogramas. Se recolectaron diez videos de muestra por cada seña, tanto para RGB como para profundidad, con un total de 200 muestras. En la etapa de segmentación, cada video fue transformado a una secuencia de imágenes, donde de cada video se obtienen 30 imagenes, mismas que fueron segmentadas para aislar la región de la mano, para después realizar la extracción de las características relevantes.

Para la extracción de características utilizaron el análisis discriminante lineal (LDA), se obtuvo un vector de características de dimensión 5280. Se hicieron pruebas del sistema utilizando LDA como clasificador, usando el 30 % de los datos para las pruebas y el resto para el entrenamiento, se obtuvo una exactitud promedio del 99.8 %.

2.3. Reconocimiento de gestos con cámaras

Prashant G. Ahire y Kshitija B. Tilekar [27], realizaron un traductor bidireccional que transforma voz a señas y viceversa, utilizando el Lenguaje Indio de Señas. Para hacer la conversión de voz a señas utilizaron el API de Google Voz a Texto (STT) por sus siglas en inglés, donde una vez identificada la palabra escrita, se busca en una base de datos que contiene las animaciones de señas y se muestra como salida dicha animación.

En cuanto a la traducción de señas a voz, se utiliza como dispositivo de entrada una cámara web. Primero descomponen el vídeo en una secuencia de imágenes, la segunda parte del proceso es remover el ruido de las imágenes obtenidas utilizando una técnica basada en histogramas. Después se buscan regiones de interés utilizando el método de crecimiento de región. Ya teniendo el área de interés se procede a hacer un mapeo del gesto obtenido

por la cámara sobre los que se encuentran en la base de datos, esto se logra utilizando una aproximación basada en correlación. Una vez reconocido el gesto se procede a transformar el texto a voz mediante el API de Google Texto a Voz (TTS) por sus siglas en inglés, misma que genera el audio correspondiente.

Prerna Gupta y Garima Joshi [28], llevaron a cabo un trabajo donde hacen el reconocimiento de gestos del lenguaje indio de señas. Utilizaron videos de los gestos como datos de entrada del sistema, los videos fueron hechos por 7 voluntarios, además se utilizó un fondo negro y cada brazo estaba cubierto por una manga negra que deja visible sólo la mano de quien hace el gesto.

Primero hicieron un preprocesamiento de las imágenes obtenidas de los videos, para extraer el contorno de la mano. Despué utilizaron 2 técnicas de rastreo, una de ella fue cambio de significado y la otra fue el filtro Kalman. Los resultados de los experimentos arrojaron que utilizando el filtro Kalman, como herramienta de rastreo, se obtienen mejores indicadores de desempeño tales como: menor tiempo de respuesta, mayor precisión, menor afectación por cambios de iluminación, y buen reconocimiento de los gestos.

Neha Singh y Neha Baranwal [29], desarrollaron un marco de trabajo para el reconocimiento de gestos del lenguaje indio de señas. Se utilizaron videos de los gestos como datos de entrada. El primer paso fue dividir los videos en una secuencia de imágenes y convertirlas en imágenes en escala de grises. Después se aplicó una transformada wavelet discreta con un filtro Haar para reducir la dimensionalidad de los datos.

Para la extracción de las características utilizaron los coeficientes cepstrales en las frecuencias de Mel y para la clasificación utilizaron una máquina de soporte vectorial y el k-vecino más cercano. Realizaron pruebas utilizando 42 gestos del lenguaje indio de señas, donde 23 eran gestos estáticos y 19 eran gestos dinámicos. La técnica que propusieron genera mejor índice de reconocimiento de gestos utilizando la máquina de soporte vectorial. Además con el uso de los coeficientes cepstrales en las frecuencias de Mel se reduce el tamaño de los datos a ser procesados, dando lugar a una disminución en el tiempo de procesamiento.

2.4. Precisión de *leap motion*

Jože Guna y Grega Jakus [30], realizaron pruebas para analizar el comportamiento, confiabilidad y precisión del sensor *leap motion*. Utilizaron 2 escenarios principales, uno estático y otro dinámico. Además *leap motion* fue instalado de manera fija sobre una mesa de 60 x 60 cm y se utilizó un equipo de alta presición de rastreo óptico que consta de 8 cámaras de alta velocidad Oqus 3+ y el software Qualisys Track Manager (version 2.8), que fueron usados como sistema de referencia .

Las condiciones del ambiente se controlaron de tal forma que todos los experimentos se realizaron con una temperatura constante de 22 C y con una intensidad de iluminación de aproximadamente 500 lux. El sistema de referencia fue configurado para que su frecuencia de muestreo fuera de 500 Hz.

En la parte estática utilizaron un modelo plástico de una mano para simular una mano real (ver Figura 2.8), con la finalidad de mantener un sólo gesto y poder moverlo dentro de 37 ubicaciones dentro del rango de lectura del sensor. Con el fin de analizar en que zonas existe un mejor desempeño del sensor. Por cada ubicación se tomaron al menos 4,000 muestras, en total se obtuvieron 214,546 que cubren las 37 ubicaciones seleccionadas.

Figura 2.8 Modelo plástico de una mano utilizado en el experimento.

Para la parte dinámica utilizaron una herramienta en forma de v para simular 2 dedos que puedan ser rastreados por el sensor y que además permitieran

mantener una distancia constante de los 2 dedos. Dicha herramienta fue desplazada con una velocidad aproximada de 100 mm/s. Se tomaron lecturas de un total de 119,360 posiciones válidas con una densidad de muestreo de 1.2 muestras por cm^3.

Los resultados de los experimentos realizados, reflejaron que en el escenario estático los mejores reconocimientos de la mano se observan a menores distancias con respecto al sensor. Y a medida que la mano se aleja del sensor empiezan a observarse variaciones en las lecturas. En el escenario dinámico se encontraron inconsistencias en el desempeño del sensor debido a la variación en la frecuencia de muestreo, lo que complica los métodos de pos procesamiento.

Frank Weichert y Daniel Bachmann [31], también realizaron pruebas de presición y confiabilidad del sensor *leap motion*. Utilizaron un robot industrial llamado Kuka Robot KR 125/3 (ver Figura 2.8), con precisón repetible de menos de 0.2 mm, ya que la amplitud de temblor de una persona joven es de 0.4 mm $\pm$ 0.2 mm.

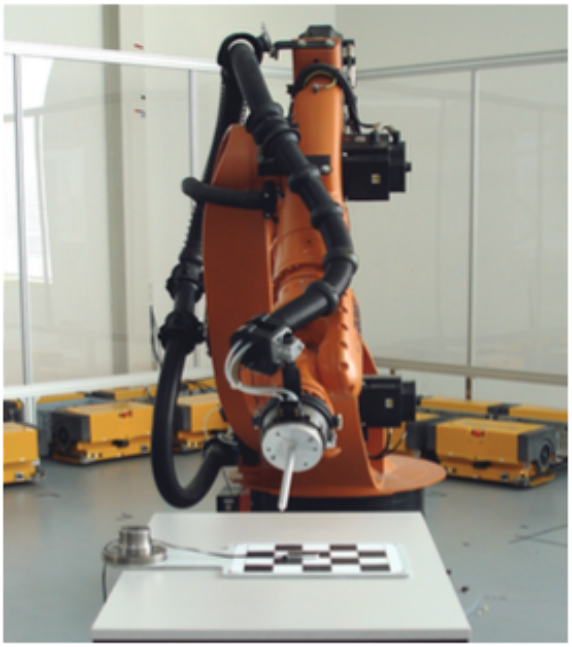

Figura 2.9 Kuka Robot KR 125/3.

Se realizaron pruebas en escenarios estáticos y dinámicos, donde el robot utiliza una pluma que es el objeto que rastrea el sensor. También se utilizaron plumas con diferentes diámetros (3mm, 4mm, 5mm, 6mm, 8mm y 10mm) para medir su posible influencia en las mediciones. Las condiciones del ambiente se controlaron de tal forma que todos los experimentos se realizaron con una temperatura constante de 23 C y con una intensidad de iluminación de aproximadamente 250 lux.

Las pruebas para el escenario estáticos se realizaron con 5,000 mediciones para cada dimensión de diámetro de la pluma. Las pruebas dinámicas se realizaron desplazando la pluma en rutas lineales de 200 mm a través de los planos *xy, xz, yz*. También se realizó una ruta que describe la función seno a través del plano *xy*.

Los resultados de las pruebas en el escenario estático revelan que la desviación de los puntos medidos con los puntos obtenidos de las lecturas del sensor es menor de 0.2 mm. Para el escenario dinámico encontraron que la desviación de la precisión del sensor es de menos de 2.5 mm. Cabe mencionar que este análisis fue realizado acorde al estándar ISO 9283, que es utilizado para robots industriales.

Capítulo 3

Marco Teórico

3.1. Redes Neuronales Artificiales

Las redes neuronales artificiales son un paradigma de programación bio-inspirado ya que modela la capacidad de procesamiento de información del sistema nervioso, el cual permite a una computadora aprender a partir de datos de observación [32, 33]. Para comprender cómo funcionan las redes neuronales artificiales, primero se deben considerar algunas propiedades esenciales de las redes neuronales biológicas desde el punto de vista del procesamiento de información.

Los sistemas nerviosos de los animales están compuestos por miles de millones de celulas interconectadas, cada una de ellas es una estructura muy compleja que procesa señales entrantres en muy diversas maneras [32]. En comparación con las puertas electrónicas, las neuronas son mucho más lentas ya que tienen tiempos de conmutación del orden de milisegundos en contraparte las puertas electrónicas tienen tiempos del orden de los nanosegundos. Sin embargo el cerebro puede resolver problemas que hasta el momento las computadoras no han podido resolver de manera eficiente [32].

Hoy en día, los mecanismos para la producción y el transporte de señales de una neurona a otra, son fenómenos fisiológicos bien entendidos, pero cómo estos sistemas individuales cooperan para formar sistemas complejos y masivamente paralelos capaces de hazañas increíbles de procesamiento de información aún no ha sido completamente descifrado [32].

El sistema nervioso de un animal es una estructura de procesamiento de

información . Las entradas sensoriales, es decir, las señales del medio ambiente, se codifican y se procesan para evocar la respuesta apropiada. Las redes neuronales biológicas son sólo una de las muchas soluciones posibles al problema de procesamiento de la información. La principal diferencia entre las redes neuronales y sistemas informáticos convencionales es el paralelismo masivo y redundancia que explotan con el fin de hacer frente a la falta de fiabilidad de las unidades de computación individuales. Por otra parte, las redes neuronales biológicas son sistemas de auto-organizados y cada neurona individual es también una estructura de auto-organizada delicada capaz de procesar información de muchas maneras diferentes [32].

En las redes neuronales biológicas la información se almacena en los puntos de contacto entre diferentes neuronas, las denominadas sinapsis.

Los modelos clásicos de las redes neuronales artificiales se basan en una simple descripción de cómo están estructuradas las neuronas que conforman el sistema nervioso de los animales [34]. Las redes neuronales artificiales tienen diversos elementos que hacen posible que realicen la tarea de aprender, mismos que se presentan a continuación.

3.1.1. Neurona

Es la unidad de procesamiento de las redes neuronales artificiales [35]. El primer modelo de neurona fue desarrolado por McCulloch-Pitts en 1943. Años más tarde Frank Rossenblatt desarrolló un tipo de neurona al que llamó perceptrón, la cual es la base para entender modelos de neuronas más modernas que se utilizan en el presente trabajo [33].

Un perceptrón puede tener muchas entradas binarias, $x_1, x_2, ..., x_n$, y produce una sola salida binaria (ver Figura 3.1):

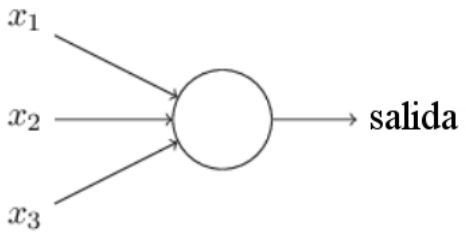

Figura 3.1 Perceptrón

Rosenblatt propuso introducir pesos, $w_1, w_2, ..., w_n$, que son números reales que expresan la importancia de las entradas para el procesamiento de la respuesta de la neurona. La salida de la neurona, 0 o 1, está determinada si la suma de las entradas multiplicadas por su respectivo peso, $\sum_j w_j x_j$, es menor o mayor que algún valor límite. Así como los pesos, el valor límite es un número real, el cual es un parámetro de la neurona [35].

$$salida = \begin{cases} 0 & \text{si } \sum_j w_j x_j \leq \text{ valor límite} \\ 1 & \text{si } \sum_j w_j x_j > \text{ valor límite} \end{cases}$$

Cada neurona realiza un trabajo relativamente simple que es recibir la señal de entrada de cada uno de sus vecinos o de fuentes externas y usar dichos valores para generar una señal de salida que se propagará hacia otras neuronas. El sistema es inherentemente paralelo en el sentido de que muchas neuronas pueden realizar sus operaciones al mismo tiempo [35].

Debido a las limitaciones y deficiencias que presentan los perceptrones, se definieron nuevos modelos de neuronas que pudieran generar representaciones más eficientes. Un modelo de neurona muy utilizado es la neurona sigmoide, la cual tiene la característica de soportar pequeños cambios en los pesos y en los valores límite, de tal forma que permitan ajustar de una manera más fina la salida de neurona.

El cambio más significativo en la neurona sigmoide es que tanto las entradas como la salida de la neurona son números reales entre 0 y 1, en lugar de ser binarias como en el caso del perceptrón. Otro cambio importante de las neuronas sigmoides es que el valor límite se cambia al otro lado de la desigualdad y se reemplaza por lo que se conoce como el *bias del perceptrón*, bias $\equiv -$valor límite. Además podemos simplificar la expresión $\sum_j w_j x_j$ haciendo uso de la notación del producto punto $w \bullet x \equiv \sum_j w_j x_j$ donde w y x son vectores y sus componentes son pesos y entradas respectivamente [36]. Usando el concepto de bias en lugar del valor límite, la regla de salida de la neurona sigmoide queda de la siguiente manera:

$$salida = \begin{cases} 0 & \text{si } w \bullet x + \text{bias} \leq 0 \\ 1 & \text{si } w \bullet x + \text{bias} > 0 \end{cases}$$

Se puede pensar en el bias como una medida de qué tan fácil es conseguir que la neurona tenga como salida un 1. Para una neurona con un bias muy

grande, es muy fácil obtener como salida un 1. Pero si el bias es muy negativo, entonces es difícil para la neurona dar como salida un 1.

3.1.2. Función de activación

La función de activación de una neurona define la salida de ésta dada una entrada o un conjunto de entradas. En el caso de las neuronas sigmoides su salida está determinada por la función sigmoide $\sigma(w \bullet x + \text{bias})$ que se define de la siguiente manera [33]:

$$\sigma(\mathrm{z}) = \frac{1}{1 + \mathrm{e}^{-\mathrm{z}}}$$

donde $z = (w \bullet x + \text{bias})$.

Para entender la similitud con el modelo de perceptrón, se toma $z \equiv (w \bullet x + \text{bias})$ que es un número positivo muy grande, entonces $e^{-\mathrm{z}} \approx 0$, lo que nos da como resultado que $\sigma(\mathrm{z}) \approx 1$. En otras palabras, cuando $z = (w \bullet x + \text{bias})$ es muy grande y positivo, la salida de la neurona sigmoide es de aproximadamente 1, tal como lo habría sido para un perceptrón.

Por el contrario cuando $z \equiv (w \bullet x + \text{bias})$ es muy negativo, entonces $e^{-\mathrm{z}} \to \infty$, lo que nos da como resultado que $\sigma(\mathrm{z}) \approx 0$. Entonces cuando $z = (w \bullet x + \text{bias})$ es muy negativo, el comportamiento de una neurona sigmoide también se aproxima mucho a un perceptrón. Es sólo cuando $z = (w \bullet x + \text{bias})$ es de tamaño modesto que hay mucha desviación del modelo de perceptrón [37]. Se puede apreciar la forma de la función sigmoide en la Figura 3.2.

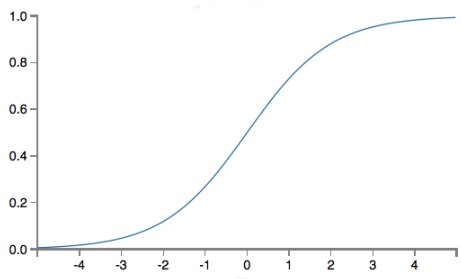

Figura 3.2 Forma de la función sigmoide.

Una de las características que debe cumplir la función de activación utilizada en la red neuronal, es que dicha función sea diferenciable para poder aplicar

los algoritmos de aprendizaje descritos en la sección 3.1.4. La derivada de la función sigmoide es de la siguiente manera:

$$\frac{d\sigma(\mathrm{z})}{dz} = \sigma(\mathrm{z})(1 - \sigma(\mathrm{z}))$$

3.1.3. Estructura de una red neuronal

Una red neuronal artificial está conformada por una serie de capas que son un arreglo de neuronas, las cuales se interconectan con las neuronas de capas anteriores y posteriores, por lo general las capas pueden dividirse en tres tipos diferentes de acuerdo a su posición dentro de la estructura de la red neuronal. Estos tres tipos de capas son: capas de entrada, capas de salida y capas ocultas [38].

La capa de entrada se refiere a aquella capa que contiene a las neuronas que son alimentadas por una entidad externa a la red neuronal. Esta capa tiene distintas características, una de ellas es que las neuronas no llevan a cabo ningún tipo de procesamiento en absoluto, sino más bien son las unidades especiales que se encargan de alimentar a las demás capas de la red neuronal con los valores obtenidos del exterior de la red neuronal [37, 39].

La capa intermedia se llama una capa oculta, ya que las neuronas en esta capa no son ni entradas ni salidas. El término "oculta"quizá suena un poco misterioso pero realmente no significa nada más que "no es una entrada o una salida". La red de la Figura 3.3 tiene una sola capa oculta, pero algunas redes tienen múltiples capas ocultas [34].

La capa más a la derecha o de salida contiene las neuronas de salida, las cuales en su conjunto representan la salida o respuesta de la red neuronal en su conjunto.

Tales redes de capas múltiples se denominan a veces perceptrones multicapa, a pesar de estar formado por neuronas sigmoides, no de perceptrones.

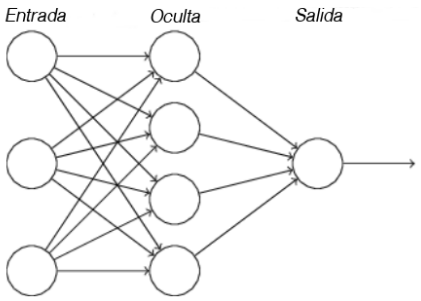

Figura 3.3 Estructura de un perceptrón multicapa.

3.1.4. Algoritmos de aprendizaje de una red neuronal

Supongamos que tenemos una red neuronal, que nos gustaría utilizar para aprender a resolver algún problema de clasificación. Y nos gustaría que la red pudiera aprender pesos y bias de modo que la salida de la red clasifica correctamente los elementos de dicho problema. Para ver cómo el aprendizaje podría funcionar , supongamos que hacemos un pequeño cambio en un cierto peso (o en algún bias) en la red . Lo que nos gustaría es que este pequeño cambio en el peso, sólo corresponda a un pequeño cambio en la salida de la red [36] ver Figura 3.4.

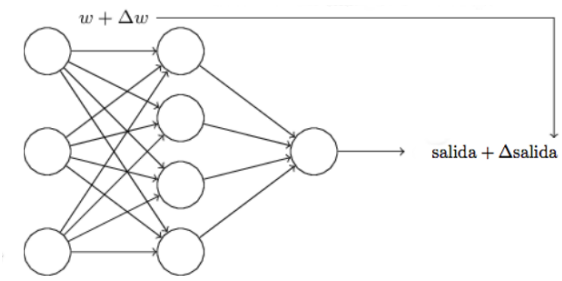

Figura 3.4 Un cambio pequeño en un peso debe corresponder a un cambio pequeño en la salida

Si fuera cierto que un pequeño cambio en un peso (o en algún bias) hace que la salida de la red sólo tenga una pequeña variación, entonces podríamos utilizar este hecho para modificar los pesos y bias para conseguir que la red neuronal se comporte de la manera que queremos. Por ejemplo, supongamos que la red está clasificando erróneamente una imagen como un 8 cuando debería ser un 9.

Con las afirmaciones anteriores es evidente la manera en que se puede hacer un pequeño cambio en los pesos y bias para que el resultado de la red se sitúe un poco más cerca de la clasificación de la imagen como un 9. Y luego nos repetimos esto, haciendo el cambio de los pesos y bias una y otra vez para producir con cada iteración una mejor salida. Se dice entonces que la red neuronal estaría aprendiendo [33].

Los algoritmos de aprendizaje se pueden dividir en métodos supervisados y no supervisados. El aprendizaje supervisado se refiere a un método en el que se recogen y se presentan a la red neuronal algunos vectores de entrada. La salida calculada por la red es observada y la desviación de la respuesta esperada es medida. Los pesos se corrigen de acuerdo con la magnitud del error en el modo definido por el algoritmo de aprendizaje. Este tipo de aprendizaje también se llama aprendizaje con un maestro, ya que un proceso de control conoce la respuesta correcta para el conjunto de vectores de entrada seleccionados.

El aprendizaje no supervisado se utiliza cuando, para una determinada entrada, la salida numérica exacta de una red que se debe producir es desconocido.

Gradiente descendiente

Para esta sección usaremos la notación x para denotar una entrada de entrenamiento. Será conveniente considerar cada entrada de entrenamiento x como un vector. Vamos a denotar la correspondiente salida deseada por $y = y(x)$, donde y es un vector que representa el número de neuronas en la capa de salida. Lo que nos interesa es conocer un algoritmo que nos permita encontrar los pesos y bias de modo que la salida de la red se aproxima $y(x)$ para todo x entradas de entrenamiento [33]. Para cuantificar qué tan bien nos estamos acercando a este objetivo definimos la función de costo:

$$C(w, b) \equiv \frac{1}{2n} \sum_{x} \parallel y(x) - a \parallel^2$$

Donde w denota el conjunto de todos los pesos en la red, b todos los bias, n es el número total de entradas de entrenamiento, a es el vector de las salidas de la red cuando x es la entrada, y la suma es sobre todas las entradas de entrenamiento, x. Por supuesto, la salida a depende de x, w y b. La función expuesta se denomina función de costo cuadrático, algunas veces también se conoce como el error cuadrático medio, ECM.

Cuando el costo $C(w, b)$ se hace pequeño, es decir, $C(w, b) \equiv 0$, es precisamente cuando $y(x)$ es aproximadamente igual a la salida, a, para todas las entradas de entrenamiento, x. Así que nuestro algoritmo de entrenamiento ha hecho un buen trabajo si se encontraron pesos y bias de manera que $C(w, b) \equiv 0$. Por el contrario, no está haciéndolo bien cuando $C(w, b)$ es grande, lo que significaría que $y(x)$ no está cerca de la salida a. Así que el objetivo de nuestro algoritmo de entrenamiento será minimizar el costo $C(w, b)$ como una función de pesos y bias. En otras palabras, queremos encontrar un conjunto de pesos y bias que hacen que el costo sea lo más pequeño posible [33].

Lo que nos gustaría es encontrar donde $C(w, b)$ alcanza su mínimo global. Una forma de atacar el problema es utilizar el cálculo para tratar de encontrar el mínimo global analíticamente. Podríamos calcular derivadas y luego tratar de utilizarlos para encontrar lugares donde $C(w, b)$ es un valor extremo. La función $C(w, b)$ puede ser una función complicada de muchas variables lo cual complicaría de manera significativa el objetivo de encontrar un mínimo global, por lo que el cálculo no funciona de manera eficiente [35].

Afortunadamente, hay una analogía que sugiere un algoritmo que funciona bastante bien, el cual se basa en calcular el gradiente de la función $C(w, b)$ el cual se denota como ∇C e indica la dirección de máximo crecimiento de la función $C(w, b)$, pero como necesitamos la dirección en la cual la función $C(w, b)$ muestra el máximo decrecimiento, debemos multiplicar el ∇C por un factor de aprendizaje, denotada como η, la cual indica el tamaño del paso hacia el mínimo, como η es un número real positivo, se multiplica el ∇C por el negativo de η [34]. Entonces el desplazamiento hacia el mínimo global Δv queda definido de la siguiente manera:

$$\Delta v = -\eta \nabla C$$

Una variante del gradiente descendiente es el gradiente descendiente estocástico, la diferencia entre los dos algoritmos es que mientras el gradiente descendiente utiliza todo el conjunto de datos de entrenamiento, el gradiente descendiente estocástico utiliza sólo una muestra del conjunto de datos de entrenamiento para acelerar el proceso de parendizaje de la red neuronal [33].

Algoritmo de backpropagation

El algoritmo *backpropagation* es el metodo de entrenamiento más utilizado en redes con conexión hacia delante. Es un método de aprendizaje supervisado de gradiente descendente, en el que se distinguen claramente dos fases: primero se aplica un patrón de entrada, el cual se propaga por las distintas capas que componen la red hasta producir la salida de la misma.

Esta salida se compara con la salida deseada y se calcula el error cometido por cada neurona de salida. Estos errores se transmiten hacia atrás, partiendo de la capa de salida, hacia todas las neuronas de las capas intermedias. Cada neurona recibe un error que es proporcional a su contribución sobre el error total de la red. Basándose en el error recibido, se ajustan los errores de los pesos de cada neurona, de igual forma se actualizan los valores de los bias de cada neurona [39].

3.2. *Leap Motion*

El sensor *leap motion* es un dispositivo periférico USB pequeño, que está diseñado para ser colocado en un escritorio físico. También se puede montar en un casco de realidad virtual. Dicho dispositivo consta de 2 cámaras infrarrojas y 3 LEDs infrarrojos, que permite hacer el rastreo tridimensional completo de ambas manos, dando información específica de puntos relevantes tales como las falanges, puntas de los dedos, tamaño de los dedos, centro de la palma, orientación con respecto a los ejes del sensor, velocidad, etc [3]. Lo cual ha permitido su uso en la manipulación de robots, control de la pc, juegos, reconocimiento de lenguaje de señas, sólo por mencionar algunas aplicaciones[1].

De acuerdo con las especificaciones del fabricante *leap motion* puede dar seguimiento de las manos a una velocidad de hasta 200 cuadros por segundo, además de proporcionar un campo de interacción tridimensional de 150 grados y 8 pies cúbicos. Una representación gráfica del sensor puede ser apreciada en la Figura 3.5 .

El alcance efectivo del rastreo de objetos por parte de *leap motion* se extiende desde aproximadamente 25 a 600 milímetros por encima del dispo-

[1]Leap Motion, https://www.leapmotion.com/product/desktop, Julio 2016.

Figura 3.5 Sensor *leap motion*.

sitivo (1 pulgada a 2 pies). La detección y seguimiento de los objetos en el área descrita se realiza de mejor manera cuando el sensor tiene una visión clara y de alto contraste de la silueta del objeto rastreado. El software *leap motion* combina los datos obtenidos del sensor con un modelo interno de la mano humana para ayudar a hacer frente a condiciones desafiantes de seguimiento.

El sistema *leap motion* emplea un sistema de coordenadas cartesianas donde la orientación positiva es definida por la regla de la mano derecha. El origen se centra en la parte superior del sensor *leap motion*. Los ejes x y z se encuentran en el plano horizontal, donde el eje x corre paralelo al borde largo del dispositivo, el eje y es vertical, con valores positivos creciente hacia arriba (en contraste con la orientación hacia abajo de la mayoría de los sistemas de coordenadas que utilizan los gráficos por computadora) y el eje z tiene valores positivos crecientes hacia el usuario. En la Figura 3.6 se puede el sistema de coordenadas descrito.

3.2.1. Modelo de mano utilizado por leap motion

El modelo de la mano proporciona información acerca de la identidad, posición y otras características de una mano detectada, el brazo al que se une la mano, y las listas de los dedos asociados con la mano. Las manos están representados por la clase de Hand.

El software leap motion utiliza un modelo interno de una mano humana para brindar un seguimiento predictivo incluso cuando partes de una mano no sean visibles. El modelo de la mano siempre proporciona posiciones para cinco dedos, aunque se debe tomar en consideracio que el rastreo es óptimo

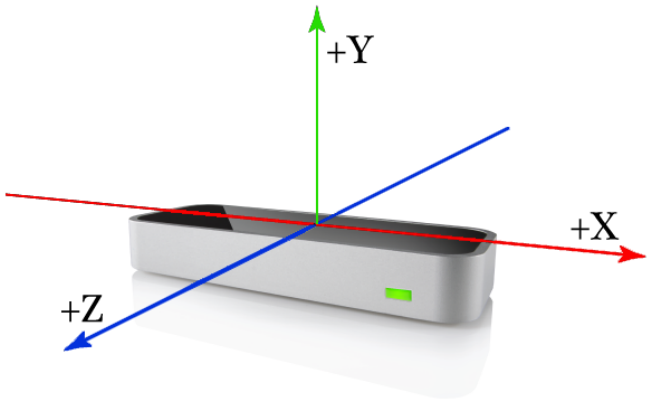

Figura 3.6 Sistema de coordenadas del sensor *leap motion*.

sólo cuando la silueta de una mano y todos sus dedos son claramente visibles. El software utiliza las partes visibles de la mano, su modelo interno y las observaciones pasadas para calcular las posiciones más probables de las partes que no están claramente visibles. Además se debe tomar en cuenta que los movimientos sutiles de los dedos contraídos contra la palma o las partes de los dedos que estén en una posición donde no puedan ser identificados por los sensores de *leap motion*, generalmente no podrán ser detectados de manera correcta.

El rastreo de los dedos por parte de *leap motion* proporciona información sobre cada dedo en una mano. Si todo o parte de un dedo no es visible, las características de los dedos se estiman basándose en observaciones recientes y el modelo anatómico de la mano. Los dedos están representados por la clase de Finger.

Un objeto de tipo *Finger* proporciona una lista de objetos de tipo *Bone* que corresponde a los huesos que conforman el dedo, cada objeto *Bone* describe la posición y orientación del hueso del dedo. El modelo anatómico del dedo utilizado por *leap motion* considera que todos los dedos contienen cuatro huesos ordenados desde la base hasta la punta. En la Figura 3.7 se presenta el modelo anatómico de la mano utilizado por *leap motion*, donde se puede observar que la representación del pulgar no se ajusta exactamente con el sistema de nomenclatura anatómica estándar (mostrado en la sección 3.3). Un pulgar real tiene un hueso menos que los otros dedos. Sin embargo, para facilidad de programación, el modelo del pulgar utilizado por *leap motion*

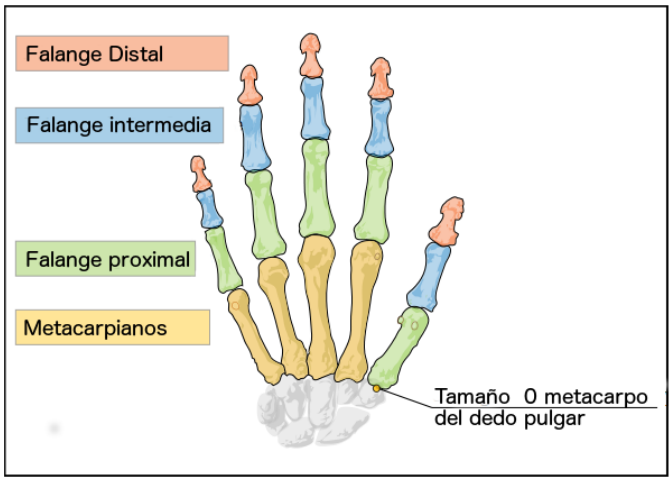

Figura 3.7 Morfología de las manos utilizada por *leap motion*.

incluye un hueso metacarpiano de longitud cero de modo que el pulgar tiene el mismo número de huesos en los mismos índices que los otros dedos. Como resultado el hueso metacarpiano del pulgar se etiqueta como una falange proximal y la falange proximal se etiqueta como la falange intermedia en el modelo de dedos de *leap motion*.

3.3. Morfología de las manos

En la Figura 3.8 se muestra como las manos constan de 5 dedos los cuales están constituidos por 3 falanges: distal, media y proximal (excepto el dedo pulgar que no tiene falange media), además cada dedo cuenta con un hueso metacarpiano. Los huesos se describen de la siguiente manera:

- Metacarpiano.- Es el hueso que conecta el dedo a la muñeca.
- Falange proximal.- Es el hueso en la base del dedo, conectado a la palma.
- Falange intermedia.- Es el hueso medio del dedo situado entre las falanges proximal y distal.
- Falange distal.- Es el hueso terminal en el extremo del dedo.

Para formar señas estas falanges deben ser puestas en diferentes posiciones, las cuales se caracterizan por formar angulos entre las falanges que

componen al dedo, de igual manera las separaciones entre los dedos forman ángulos.

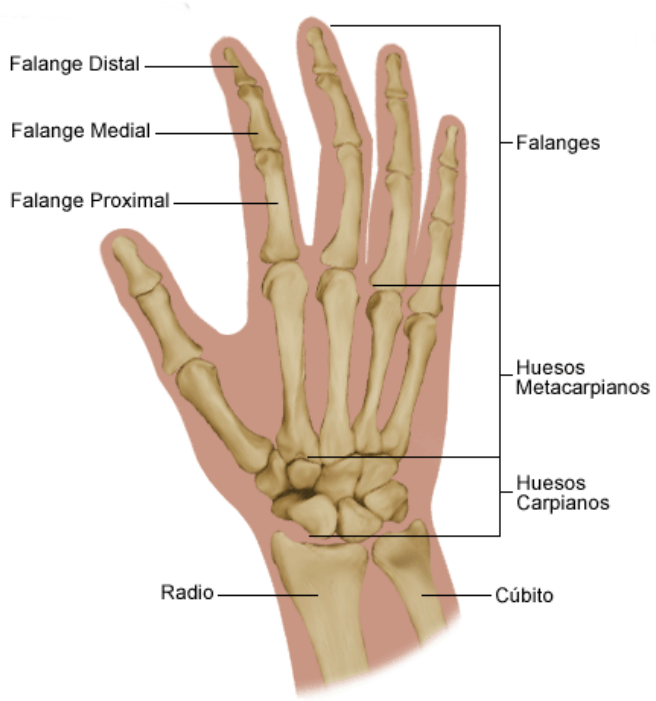

Figura 3.8 Morfología de las manos.

Capítulo 4

Propuesta de Solución

4.1. Arquitectura del sistema

Para realizar la traducción del lenguaje mexicano de señas se propone una arquitectura que consta de los siguientes componentes (ver Figura 4.1):

- Sensor.
- Reconocimiento del gesto.
- Respuesta del sistema.

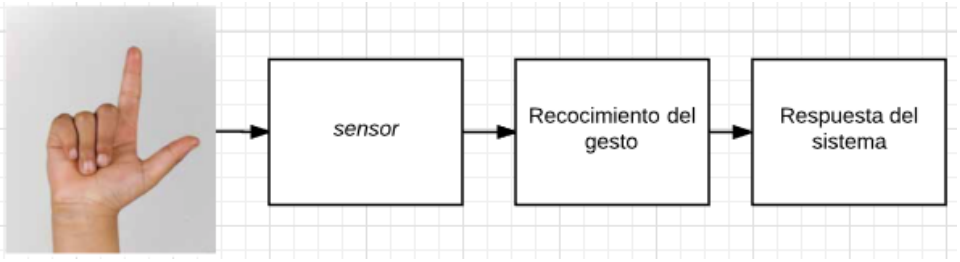

Figura 4.1 Arquitectura del sistema.

4.1.1. Sensor

En la arquitectura propuesta se contempla un sensor que provea información de la mano y de la posición de los dedos al formar una seña. Para dicho trabajo se propone utilizar el sensor *leap motion*, ya que es capaz de proporcionar información específica de la mano y de los dedos, misma que

es crucial al momento de hacer el reconocimiento de señas. El conjunto de datos que es objeto de estudio se describe en la sección 4.2.

Existen 2 tipos de extracción de datos que se deben realizar en el sistema, el primero es cuando los datos que se extraen son utilizados para realizar el entrenamiento del componente de reconocimiento y el otro es cuando el sistema se encuentra listo para realizar el reconocimiento de las señas en tiempo real.

Cuando la información obtenida del sensor es utilizada para generar los conjuntos de datos para entrenar el componente de reconocimiento, dicha información se debe almacenar en archivos para su posterior utilización. Cuando la información del sensor es utilizada por la herramienta que hace la traducción de señas, se envía directamente al componente de reconocimiento (ver Figura 4.2).

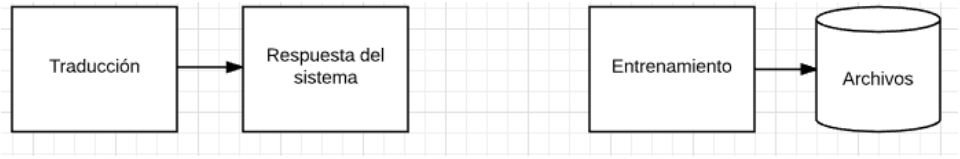

Figura 4.2 Datos obtenidos de *leap motion*.

4.1.2. Reconocimiento del gesto

En este componente el elemento principal es una red neuronal de tipo perceptrón multicapa que consta de 3 capas, una capa de entrada, una capa oculta y una capa de salida (ver sección 3.1.3), que tiene la labor de reconocer el gesto que se realiza frente a *leap motion*. En este componente se realizan diversas acciones tales como limpieza y partición de datos, un preprocesamiento de la información obtenida y el entrenamiento de la red neuronal (ver Figura 4.3).

Figura 4.3 Acciones realizadas en el componente de reconocimiento de gestos.

Limpieza y partición de datos

En esta sección se deben realizar las transformaciones de los datos obtenidos por el sensor leap motion, que son destinados para el entrenamiento de la red neuronal, tales como limpieza, formato y tamaño de la información.

Se requiere una limpieza de datos, ya que se utilizan archivos donde se almacenan los datos que son extraídos de *leap motion*, como dicho sensor soporta diversos tipos de datos, cuando se hace el almacenamiento de la información en archivos, se llegan a introducir caracteres que interfieren en el entrenamiento de la red neuronal, por ello se realiza una acción de limpieza de datos.

Además de la limpieza de los datos, se debe transformar la información al formato que soporta la red neuronal, el cuál es en forma de tuplas, por ello se debe cambiar el formato de texto plano a tuplas para que pueda realizarce el entrenamiento de la red neuronal. También el tamaño de las tuplas debe ser el mismo, ya que ese es otro requerimiento del entrenamiento de la red neuronal, para ello se debe garantizar que todos las tuplas tienen el mismo tamaño. Los tamaños de las tuplas está definido por el número de características que se toman en cuenta para realizar el reconocimiento de las señas, estos tamaños son explicados en el capítulo 6.

Una vez que toda la información cuenta con el formato y dimensión correcta, se debe hacer una partición de los datos, ya que para el entrenamiento de la red neuronal se debe contar con 2 conjuntos de datos, un conjunto de datos de entrenamiento, que es con el cual se realiza el proceso de aprendizaje de la red neurona, y un conjunto de datos de prueba, con el cual se realiza la verificación del proceso de aprendizaje de la red neuronal.

Preprocesamiento

En la etapa del preprocesamiento se determina la posición de la palma de la mano, para poder decidir a que red neuronal se debe enviar la información que se extrae de *leap motion*, ya que el sistema consta de 3 redes neuronales las cuales son descritas a detalle en el capítulo 6.

Los datos que son utilizados para realizar la identificación de la posición de la palma de la mano, se toma dirección que corresponde al vector normal a la palma de la mano y se calculo el ángulo que se forma con los ejes x y y, con ello se obtiene la información necesaria para determinar la posición de la palma con respecto al sensor *leap motion*.

Entrenamiento de la red neuronal

Para realizar el entrenamiento de la red neuronal, se debe contar con los conjuntos de datos de entrenamiento y de prueba, para poder empezar a realizar los experimentos de entrenamiento, mismos que se detallan en el capítulo 6.

Para obtener una red neuronal que tenga buen porcentaje de efectividad de reconocimiento se deben realizar diversas variaciones en los parámetros de entrenamiento tales como el número de neuronas en la capa oculta, el factor de aprendizaje y el tamaño del subconjunto de datos de entrenamiento utilizado. Una vez que se obtiene un entrenamiento con un buen porcentaje de efectividad, se debe almacenar la red neuronal que resulta del entrenamiento para su posterior uso en la herramienta de traducción.

El almacenamiento de la red neuronal debe contener una descripción exhaustiva de todos los componentes de la misma, tales como la estrucutura de las 3 capas que componen al perceptrón multicapa (entrada, oculta y salida) y los valores (peso y *bias*) de todas las neuronas de todas las capas. Dicho almacenamiento de la red neuronal se realiza en un archivo de texto plano.

4.1.3. Respuesta del sistema

Ya que se tiene la información relevante que se va extraer del sensor *leap motion* y una red neuronal entrenada con un alto porcentaje de efectividad para reconocer señas, se puede utilizar una herramienta de software que sea capaz de transformar una seña en texto.

La herramienta de software primero debe contener el elemento de extracción de datos que se explicó en la sección 4.1.1, el cual permite enviar los datos del sensor *leap motion* directamente a la red neuronal. Además la herramienta de software debe contar con un módulo para cargar una red neuronal que ha sido almacenada en un archivo de texto plano, este módulo se encarga de mapear todos las características de la red neuronal para su posterior uso en la herramienta de reconocimiento de gestos.

Cuando ya se ha cargado la red neuronal en la herramienta, se le deben enviar los datos obtenidos por *leap motion*, una vez procesados los datos en la red neuronal, se obtiene una respuesta que corresponde a la identificación del gesto que se está realizando, dicha respuesta debe ser analizada para que se produzca la respuesta en texto y se muestre al usuario.

4.2. Conjunto de datos para procesamiento

En el presente trabajo se propone utilizar un conjunto de datos que contempla sólo los ángulos que se forman entre los diferentes huesos que conforman las manos, con el fin de obtener un conjunto de características que identifiquen a los gestos del lenguaje mexicano de señas, donde dicho conjunto de características sean independientes del tamaño de los huesos de las manos de las personas que ejecuten dichos gestos.

En la primera parte de la selección de datos, se analizaron algunas características que pueden ser obtenidas por el SDK de *leap motion*, estos datos se seleccionaron con la premisa de que pudieran ser tomados como referencia para generar ángulos que ayuden a describir la posición de las falanges con respecto a la palma de la mano cuando es realizada una seña.

En este primer conjunto de datos se obtuvieron 490 características, la cuáles corresponden a:

- El ángulo entre el vector normal y la dirección de la palma.
- Los ángulos de la palma respecto a los ejes *xyz*.
- Las posiciones del brazo, codo y muñeca con respecto al vector normal de la palma.
- Por cada falange se obtiene:

- El ángulo generado con el vector de dirección de la falange siguiente (excepto falange distal), ver Figura 4.4.

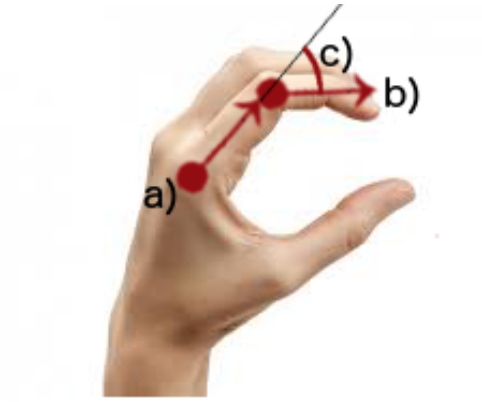

Figura 4.4 a) Vector de dirección de la falange proximal, b) Vector de dirección de la falange media y c) Ángulo formado entre a) y b)

- Los ángulos generados por los vectores base de los 3 ejes *xyz* con respecto a los vectores base de la falange siguiente y con el vector normal de la palma.
- El ángulo formado con la misma falange del dedo siguiente (excepto el dedo meñique), ver Figura 4.5.

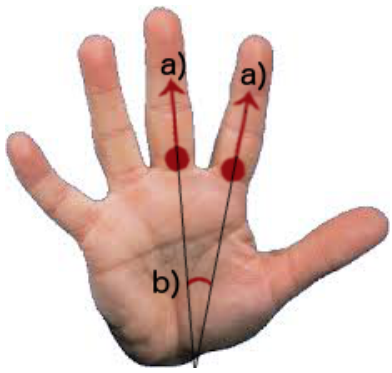

Figura 4.5 a) Vector de dirección de la falange proximal y b) Ángulo formado por los vectores de dirección.

- El ángulo formado con falange y el vector normal de la palma, ver Figura 4.6.

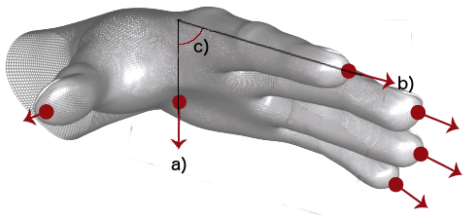

Figura 4.6 a) Vector normal de la palma, b) Vector de dirección de la falange distal del dedo índice y c) Ángulo formado entre a)y b).

El alfabeto del lenguaje de señas consta de 27 letras, de las cuales 21 son estáticas y 6 son dinámicas, en este trabajo abordaremos las 21 letras estáticas, además debe señalarse que las señas corrspondientes a las letras del alfabeto (estáticas y dinámicas) son ejecutadas únicamente con la mano derecha, debido a esto, en este trabajo se toma en cuenta sólo la mano derecha como objeto de rastreo por el sensor.

Las 21 letras estáticas del alfabeto fueron divididas en 3 grupos de acuerdo a la posición completa de la palma de la mano con respecto al sensor, las 3 posiciones de la palma que se tomaron en cuenta son: palma de frente al dispositivo, palma hacia un costado (siendo el dedo meñique el más cercano al dispositvo) y con la palma opuesta al dispositivo. Dichas posiciones se pueden observar en la Figura 4.7

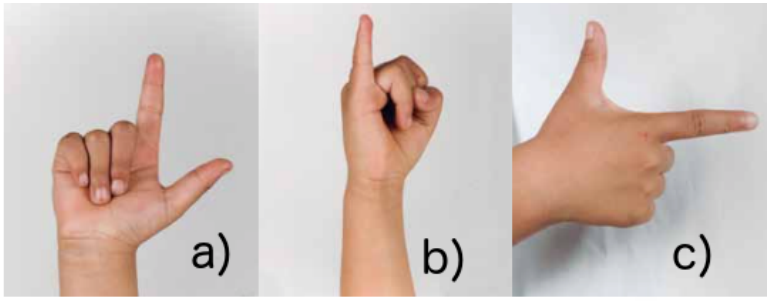

Figura 4.7 a) Posición de la palma de frente al dispositivo, b) Posición de la palma hacia un costado y c) Posición de la palma opuesta al dispositivo.

Dicha división de las letras en 3 grupos se debe a que en los experimentos realizados en el capítulo se observa que a menor número de señas a identificar por una red neuronal, es más fácil obtener un mayor porcentaje de efectividad de reconocimiento.

Cada grupo de señas será clasificado por una red neuronal independiente, para así contar con 3 redes neuronales. La red neuronal que clasifica a las señas con la palma de frente al dispositivo identifica 11 letras las cuales son: *a, b, d, e, l, r, s, t, u, v* y *w*. La red neuronal que clasifica las señas con la palma hacia un costado identifica 5 letras las cuales son: *c, f, i, o* y *p*. La red neuronal que clasifica a las letras con la palma opuesta al dispositivo identifica 5 letras: *y, g, h, m* y *n*.

La aplicación que realiza el reconocimiento de las señas, primero hace un preprocesamiento (ver sección 4.1.2) para identificar la posición de la palma de la mano, una vez obtenida la posición de la palma ya se puede saber a que red neuronal se deberán enviar los datos obtenidos por *leap motion*.

Capítulo 5

Implementación

La implementación de este trabajo se realizó usando el lenguaje de programación python [40, 41], debido a que algunos de los códigos que se tomaron como referencia están escritos en este lenguaje.

5.1. Obtención de datos de *leap motion*

Para obtener los datos del sensor *leap motion* se tomó como referencia el código de ejemplo proporcionado en el SDK de *leap motion*, al cuál se le hicieron las modificaciones pertinentes para que sólo se capturen los datos de los ángulos mencionados en la sección 4.

Leap motion llega a proporcionar hasta 200 cuadros por segundo, como se mencionó en la sección 3.2, donde cada cuadro proporciona los datos sensados. Se puede acceder a dichos datos mediante los métodos definidos por el SDK del dispositivo. En dicho SDK, cada cuadro es representado por un objeto de tipo *Frame* que contiene una lista de entidades correspondientes al objeto rastreado por el sensor. En la Figura 5.1 en la línea 40 se muestra como se obtiene el objeto *Frame* del sensor.

Cada objeto *Frame* contiene la lista de manos sensadas, donde cada mano es representada por un objeto de tipo *Hand*, dicho objeto tiene un conjunto de atributos de los cuales el que nos importa es saber si la mano rastreada es la mano derecha. Un atributo importante de la mano es el vector normal de la palma, el cual sirve como punto de referencia para realizar algunas las mediciones que son fundamentales para el reconocimiento de los gestos manuales. En la Figura 5.1 en la línea 42 se muestra como se obtienen los

objetos de tipo *Hand* que contiene el objeto *Frame*, además en la línea 43 se muestra la condición para determinar si la mano rastreada por el sensor es la mano derecha.

```
38      def on_frame(self, controller):
39          # Get the most recent frame and report some basic information
40          frame = controller.frame()
41          # Get hands
42          for hand in frame.hands:
43              if hand.is_left:
44                  continue
```

Figura 5.1 Obtener objetos de tipo *Frame* y *Hand* del SDK de *leap motion*

Un objeto de tipo *Hand* contiene la lista de los dedos que conforman la mano, dicha lista empieza con el dedo pulgar y termina con el dedo meñique, cada dedo es mapeado a un objeto de tipo *Finger*. Cada falange de los dedos se representa por un objeto de tipo *Bone*, cada objeto *Finger* contiene una lista de objetos *Bone*, la cuál inicia por el hueso metacarpal y termina con el hueso de la falange distal. La característica más importante de este objeto (al menos para este trabajo) es la dirección, la cuál está representada por un objeto de tipo *Vector* con 3 componentes correspondientes a los ejes x, y y z.

```
48              # Get the hand's normal vector and direction
49              normal = hand.palm_normal
50              # Get fingers
51              list_fingers=hand.fingers
52              for index, finger in enumerate(list_fingers, start=0):
53
54                  if (index<4):
55                      nextfinger=operator.itemgetter(index+1)(list_fingers)
56                      for falanx in range(0,4):
57                          currentbone=finger.bone(falanx)
58                          nextbone=nextfinger.bone(falanx)
```

Figura 5.2 Obtener la lista de dedos y sus correspondientes falanges

En la Figura 5.2 en la línea 49 se muestra cómo se obtiene la dirección del vector normal de la palma de mano. En la línea 51 se puede observar cómo se obtiene la lista de los dedos que conforman la mano. Para obtener la lista de las falanges que conforma cada dedo se ejecuta la línea 56.

Para generar los datos del conjunto de prueba, una vez que se verificó que la mano rastreada es la mano derecha, se procede a obtener el valor del vector normal de la palma, después se itera en la lista de los dedos y en cada uno de los dedos se itera en la lista de los huesos para obtener los vectores de dirección.

Una vez obtenido el vector de dirección del hueso, se calcula el ángulo que existe con la misma falange del dedo contiguo (El SDK de *leap motion* proporciona un método llamado *angle_to* el cual proporciona el ángulo entre 2 objetos de tipo *Vector*), excepto cuando se llega al dedo meñique. Además se calcula el ángulo que se forma con el vector normal de la palma y el ángulo formado con la falange contigua, excepto cuando se trata de la falange distal.

```
62  currentbone.direction.angle_to(nextbone.direction)* Leap.RAD_TO_DEG
63
64  currentbone.direction.angle_to(normal) * Leap.RAD_TO_DEG
65
66  currentbone.direction.angle_to(nextfalanx.direction)* Leap.RAD_TO_DEG
67
```

Figura 5.3 Cálculo de los ángulos utilizados

En la Figura 5.3 se puede observar en la línea 62 como se obtiene el ángulo entre las 2 falanges de dedos contiguos, para calcular el ángulo entre una falange y el vector normal de la palma se escribe la línea 64 y en la línea 65 se muestra como se obtiene el ángulo que se forma entre 2 falanges contiguas.

Todos los datos obtenidos se guardan en un archivo de texto (un archivo por cada seña), con el fin de poder analizar los datos posteriormente.

5.2. Limpieza y partición de datos

Una vez que se tienen las muestras de cada seña almacenadas en los archivos de texto, se procede a hacer una limpieza del archivo de texto para que no existan caracteres no deseados. También se hace un conteo del número de muestras que contiene el archivo, se entiende como una muestra al conjunto de los ángulos que se han descrito en la sección 4.

En este proyecto se fijó el número de muestras en 1100 por cada una de las 21 señas estáticas del alfabeto del lenguaje de señas mexicano, de las cuales 770 muestras son tomadas para conformar el conjunto de entrenamiento y 330 muestras forman el conjunto de prueba (70 % y 30 % respectivamente).

Una vez que se hace el conteo de muestras totales contenidas en el archivo, se verifica que mínimo haya 1100 muestras. En caso de que el número de muestras sea menor, se descarta el archivo como candidato a proporcionar los datos que se necesitan. Si existen al menos 1100 muestras en el archivo se procede a generar otro archivo que contiene únicamente los datos del conjunto de entrenamiento (770 muestras). Este archivo tiene como nombre el *id* del gesto del cual se guarda la información (el *id* es numérico y empieza por el 0), seguido del sufijo "training" (ej. 4training.txt). También se genera un archivo con el *id* del gesto y el sufijo "testing", el cual contiene los datos de prueba (330 muestras).

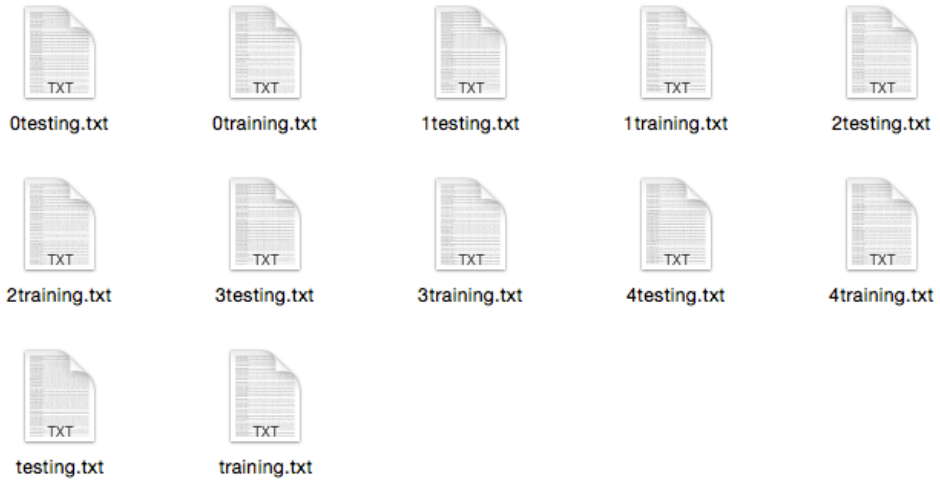

Figura 5.4 Archivos con los conjuntos de datos de entrenamiento y de prueba.

En la Figura 5.4 se muestran los archivos correspondientes a 5 señas del alfabeto del lenguaje mexicano de señas, se puede observar que los archivos que empiezan con el *id* del gesto seguido del tipo de datos que contiene el archivo.

Ya que se obtuvieron todos los datos de los gestos que van a ser reconocidos

por la red neuronal, se procede a generar los archivos que contienen todos los datos de entrenamiento y de prueba. Para esto se genera un script que une todos los datos de entrenamiento y de prueba en 2 archivos "training.txt" y "testing.txt", los cuales serán los encargados de alimentar el proceso de entrenamiento de la red neuronal. En la Figura 5.4 se puede observar que en la parte inferior se encuentran los archivos llamados "training.txt" y "testing.txt".

5.3. Red neuronal

Las 3 redes neuronales descritas en la sección 4 siguen la misma mecánica para realizar su entrenamiento, es por ello que se describe el procedimiento general que se siguió para realizar el entrenamiento de las mismas.

El código de la red neuronal se obtuvo de [33], al cual se le hicieron diversas modificaciones para que se pudiera implementar en el presente proyecto.

5.3.1. Carga de datos de entrenamiento y de prueba

La primer adaptación y la más importante, fue la de escribir un módulo para cargar los datos tanto de entrenamiento como de prueba a la red neuronal. La red neuronal acepta los datos de entrenamiento y de prueba como tuplas, donde cada tupla consta de 2 componentes, el primero es un arreglo unidimensional de tamaño igual al número de características que serán evaluadas por la red neuronal y el segundo es el número correspondiente al *id* del gesto al que corresponden los datos.

Dicho módulo, para cargar los datos recibe como primer parámetro el número de características que conforman la capa de entrada de la red neuronal, el segundo parámetro es el número de muestras totales que conforman el conjunto de entrenamiento, el tercer parámetro es el número de gestos que la red neuronal será capaz de reconocer, el cuarto parámetro es el número total de muestras que conforman el conjunto de pruebas, el quinto parámetro es el nombre del archivo que contiene los datos del conjunto de datos de entrenamiento y el sexto parámetro es el nombre del archivo que contiene el conjunto de datos de prueba.

En la Figura 5.5 se puede observar en la línea 2 como se importa el módulo descrito y en la línea 4 se observa como se generan los conjuntos de datos de entrenamiento y de prueba, mismos que contemplan 19 características

```
2  import datos
3
4  training_data, test_data = datos.load_data_wrapper(
5      19,8470,11,3630,"training19frente.txt","testing19frente.txt")
6
```

Figura 5.5 Utilización del módulo para cargar datos.

en la capa de entrada de la red neurona, un total de 8470 muestras para los datos de entrenamiento, 3630 muestras para los datos de prueba, 11 neuronas de salida y los archivos donde toma los datos en texto plano son "training19frente.txt" y "testing19frente.txt".

Estas adaptaciones se hicieron con el fin de obtener módulos que trabajen con cantidades de datos variables, ya que las 3 redes neuronales que se utilizan en este proyecto son distintas entre sí. En el capítulo 6 se detallarán los cambios en las estructuras de las redes neuronales, mismos que se facilitaron con las adaptaciones aqui mencionadas.

5.3.2. Entrenamiento y almacenamiento de la red neuronal

Para realizar el entrenamiento de las redes neuronales se generó un script que permite utilizar el módulo para cargar los datos de entrenamiento y de prueba descrito en la sección anterior. Además este script facilita la inserción de los valores con los que se desea que la red neuronal opere, tales como su estructura en cada capa, los valores del factor de aprendizaje, el tipo de inicialización de los valores de las neuronas, el tipo de medición del error, el número de épocas y el tamaño de la muestra de los datos de entrenamiento en cada época. En la Figura 5.6 se puede observar como se estructuran los parámetros con los cuales va a ser entrenada la red neuronal.

```
33 for neuronas in range(30,121,10): #numero de neuronas en la capa oculta
34     for lr in range(1,6): # learning rate
35         for mbs in range (10,31,10):
36             for times in range (1,4):
37                 ler=(float(lr)/10.0)
38                 net = network2_prueba.Network(
39                     [19, neuronas, 11], cost=network2_prueba.QuadraticCost)
40                 net.default_weight_initializer()
41                 net.SGD("red.txt",training_data, 30,mbs, ler)
```

Figura 5.6 Script para el entrenamiento de la red neuronal.

Para almacenar la red neuronal se realizaron algunos cambios en el código original obtenido de [33]. Primero se hicieron los cambios para que en el proceso de entrenamiento y validación del entrenamiento se guarde el resultado de la efectividad de reconocimiento de la red neuronal de la primer época de entrenamiento. Dicho valor será el máximo global de efectividad hasta ese momento, también se almacena la estructura de la red neuronal. Al continuar con las demás épocas de entrenamiento si se obtiene un porcentaje de efectividad mayor al registrado como el máximo global, se actualiza el nuevo máximo global de efectividad y se guardar la estructura de la red neuronal.

Para realizar el almacenamiento de la estructura de la red neuronal se adoptó una estructura de directorios, la cual se define de la siguiente manera: el primer directorio indica el tamaño de muestra en cada época, donde se pone el prefijo "muestra" seguido del tamaño (ej. "muestra10"), el segundo directorio indica que factor de aprendizaje fue utilizado, el nombre del directorio empieza por prefijo "lr" seguido por el valor del factor de aprendizaje (ej "lr0.1") y por último se crea el archivo donde el nombre indica el número de neuronas de la capa oculta de la red neuronal seguido por un guión medio y después la efectividad registrada por dicha red (ej. 50-85.32.txt).

```
mejor resultado hasta el momento
/Users/apple/Documents/LiClipse Workspace/Neural/prueba19opuesto/Quadrat
icCost/muestra30/lr0.1
 1541 / 1650 : 93.3939393939
Epoch 8
mejor resultado hasta el momento
/Users/apple/Documents/LiClipse Workspace/Neural/prueba19opuesto/Quadrat
icCost/muestra30/lr0.1
 1615 / 1650 : 97.8787878788
Epoch 9
 1615 / 1650 : 97.8787878788
```

Figura 5.7 Almacenando la red neuronal en el momento de hacer el entrenamiento de la red neuronal.

En la Figura 5.7 se puede observar como se realizar el proceso de almacenamiento de la red neuronal que obtiene el mayor porcentaje de efectividad, ésta es guardada en la estructura de directorios y con el nombre ya especificado.

Con esta nomenclatura que se adoptó si se encuentra un archivo en la direc-

ción ../muestra10/lr0.2/30-93.54.txt se puede deducir que esa red neuronal fue entrenada con un tamaño de muestra del 10 % del conjunto de datos de entrenamiento en cada época, con un factor de aprendizaje de 0.2, que la red neuronal tiene en su capa oculta 30 neuronas y que su efectividad de reconocimiento es del 93.54 %.

Como se puede observar en la Figura 5.8 se almacenan las redes neuro-

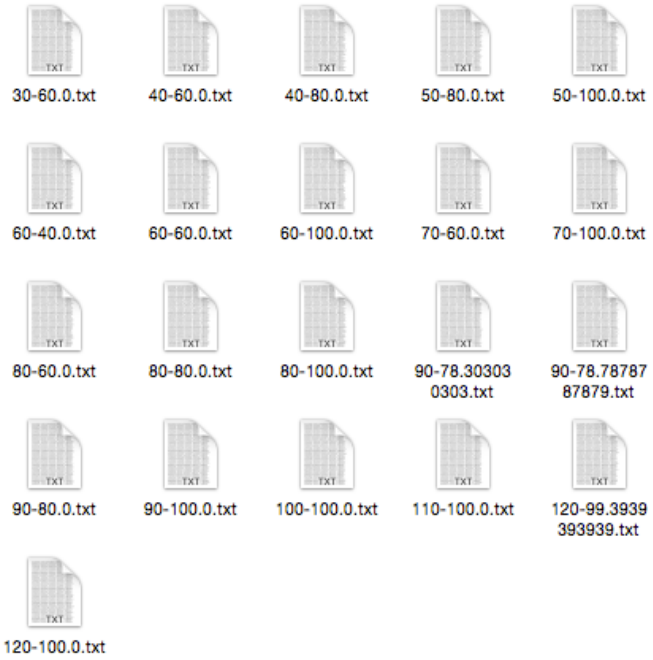

Figura 5.8 Vista de un directorio con redes neuronales almacenadas.

nales con la nomenclatura descrita, lo que permite su fácil identificación.

5.4. Herramienta de traducción del lenguaje de señas a texto

Para realizar el reconocimiento del gesto que se está analizando y generar su respectiva salida en texto, lo primero que se debe hacer es cargar las estructuras de las 3 redes neuronales, las cuales deben estar almacenadas en los archivos generados por el módulo descrito en la sección anterior.

El siguiente paso es identificar la posición de la palma de la mano con respecto al sensor *leap motion*, con el fin de determinar a qué red neuronal se debe enviar la información obtenida.

La acción de determinar la posición de la palma con respecto al sensor *leap motion*, está en función del ángulo formado por el vector normal de la palma y los vectores de los ejes x y y del sensor. Para la posición de frente con el sensor se calcula el ángulo entre el vector normal y el eje y, ya que al estar la palma de frente al dispositivo el ángulo mencionado debe estar cercano a los 0 grados (con un margen de 25 grados). Si por el contrario el ángulo está cercano a los 180 grados (con 25 grados de tolerancia), eso quiere decir que la palma está en posición opuesta al sensor.

Para determinar la posición de la palma de costado al sensor, es necesario observar el ángulo que se forma entre el vector normal de la palma y el eje x, ya que al tener dicha posición el ángulo debe estar cercano a los 0 grados, en esta posición se dejaron 45 grados de tolerancia. En la Figura 5.9 se puede observar el código utilizado para realizar la identificación de la palma.

```
92  angle2=normal.angle_to(left_vector)* Leap.RAD_TO_DEG
93  angle3=normal.angle_to(down_vector)* Leap.RAD_TO_DEG
94  if (angle3<25):
95      print ("frente")
96  if (angle2<45):
97      print ("lado")
98  if (angle3>155):
99      print("opuesto")
```

Figura 5.9 Identifiación de la posición de la palma.

Una vez determinada la posición de la palma se deben enviar los datos capturados por el sensor *leap motion* a la red neuronal que corresponde a la posición de la palma. Estos datos provenientes del sensor se guardan en un arreglo unidimensional del tamaño igual al número de características de la capa de entrada de la red neuronal.

Ya que se envían los datos a la red neuronal, y una vez que ésta devuelve el resultado de la seña que está reconociendo, se debe determinar que letra está siendo realizada por la persona, esto se logra haciendo un mapeo de las letras que corresponden a los *id* que da como resultado cada red neuronal.

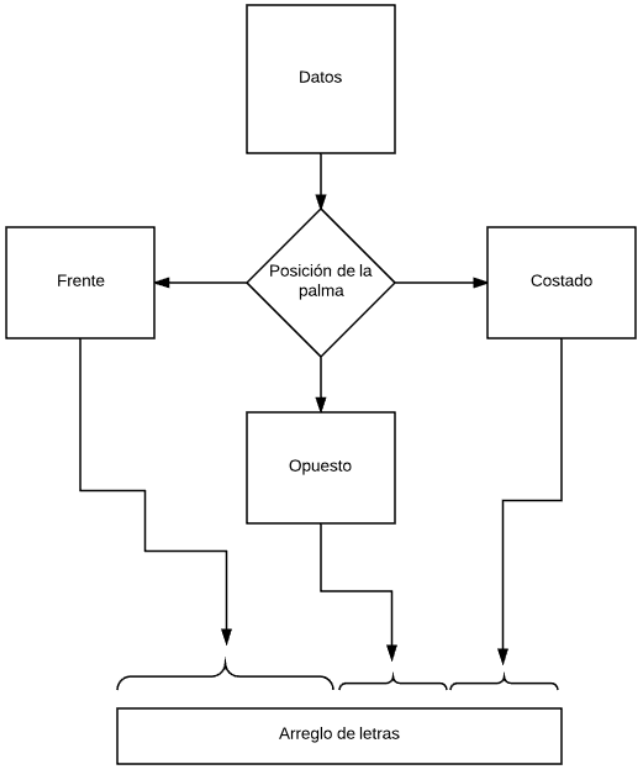

Figura 5.10 Selección de la red neuronal.

En la Figura 5.10 se puede observar como se realiza la selección de la red neuronal a la que se debe enviar los datos obtenidos por *leap motion*, tomando en cuenta la posición de la palma con respecto al sensor.

Si la red neuronal a la que se enviaron los datos es la que reconoce señas con la palma de la mano de frente al sensor, entonces el mapeo se hace directo con el *id* que da como resultado, ej. si devuelve un 8 entonces la letra que se está mostrando al sensor es una U. En caso de que la red neuronal a la que se enviaron los datos es la que reconoce las señas con la palma de la mano hacia un costado, al *id* que de como resultado se le debe agregar 11, ya que en el lugar 11 empiezan las letras que esa red puede reconocer, es decir, si envía como resultado un 2, se le deben agregar 11, lo que da como resultado 13 y ese número corresponde a la letra I.

Por último si la red a la que se le enviaron los datos es la que reconoce las señas con la palma de la mano opuesta al sensor, al *id* que devuelva como resultado se le deben agregar 11 lugares de la primera red neuronal y 5 lugares de la segunda red neuronal, por lo que si el *id* devuelto es un 3, la letra será la que esté en la posición 11+5+3=19, que corresponde a la letra M.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
A	B	D	E	L	R	S	T	U	V	W	C	F	I	O	P	Y	G	H	M	N

Figura 5.11 Vector para mapear las salidas de las redes neuronales.

En la Figura 5.11 se puede observar la estructura del arreglo unidimensional que contiene las 21 letras estáticas del lenguaje mexicano de señas.

Capítulo 6

Pruebas y Resultados

6.1. Selección de datos

Como se describió en la sección 4 los datos utilizados en este proyecto contemplan sólo los ángulos que se forman entre los diferentes huesos que conforman las manos, con el fin de obtener un conjunto de características que identifiquen a los gestos del lenguaje mexicano de señas y dicho conjunto de características sea independiente del tamaño de los huesos de las manos de las personas que ejecuten dichos gestos.

Se tomaron en cuenta los datos obtenidos cuando la mano está completamente abierta y cuando la mano está completamente cerrada (ver Figura 6.1), para poder observar qué características ofrecen una mejor descripción de los huesos de las manos en cada posición. Dichas posiciones se tomaron en cuenta para hacer el análisis de las características, debido a que son los límites, anatómicamente hablando, en los que puede encontrarse una mano.

Una vez que se obtuvieron las datos de las dos posiciones de las manos, se generó una lista en donde por cada característica obtenida se obtuvo el valor mínimo capturado, el valor máximo, la media y la varianza. En este primer conjunto de datos se obtuvieron 490 características, la cuáles son detalladas en la sección 4.2:

Una vez que se obtuvieron los datos de las características ya mencionadas, de las 2 posiciones de la mano tomadas como referencia, se observó que los vectores base no cambian en los datos de las 2 posiciones de las manos. Por lo tanto no ayudan a determinar si la mano estaba en una posición o en la

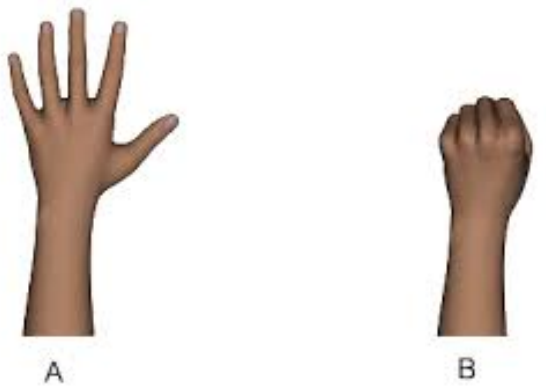

Figura 6.1 Los datos obtenidos con la mano completamente abierta y completamente cerrada

otra, por eso fueron descartados del conjunto de datos. Tampoco los ángulos de la palma con respecto a los ejes *xyz*, las posiciones del brazo, codo y muñeca con respecto al vector normal de la palma, el ángulo entre el vector normal y la dirección de la palma, aportan información significativa para el fin buscado en este proyecto, por ello de igual manera fueron descartados del conjunto de datos.

Una vez eliminadas las características que no aportan información significativa para reconocer las 2 posiciones de las manos utilizadas para hacer el análisis de las características, el conjunto de datos se redujo a 51 características.

6.2. Conjunto de datos de 51 características

6.2.1. Entrenamiento de la red neuronal

Con el conjunto de datos de 51 características (descrito en la sección anterior) se iniciaron las pruebas de entrenamiento de la red neuronal tipo perceptrón multicapa. Dicha red neuronal tiene en su capa de entrada 51 neuronas que corresponde a las 51 características del conjunto de datos. Se hizo una prueba de entrenamiento de una red neuronal que es capaz de reconocer las 21 señas estáticas que contempla este trabajo, se debe mencionar

que las pruebas realizadas en esta sección y en las posteriores, se toman 3 tamaños de muestra del conjunto de datos de entrenamiento (10 %, 20 % y 30 %), ya que se utiliza el algoritmo del gradiente descendiente estocástico (ver 3.1.4). Los resultados obtenidos fueron los siguientes:

Tabla 6.1 Resultados de entrenamiento con una muestra del 10 % del conjunto de datos de entrenamiento

LR \ #N	30	40	50	60	70	80	90	100	110	120
0.1	39.1	49.3	43.1	52.7	54.6	57.5	59.2	60.4	64.3	64.2
0.2	34.1	31.7	38.1	33.4	33.3	32.3	38.7	37.8	44.2	45.3
0.3	19.1	23.8	22.9	23.8	28.6	29.5	23.8	37.7	32.7	29.5
0.4	23.8	19.3	19.0	22.9	22.9	19.2	23.8	21.9	23.8	28.6
0.5	17.0	15.6	19.0	19.0	21.5	23.8	19.0	18.3	19.2	23.8

Como se puede observar en la Tabla 6.1 con ningún parámetro es posible alcanzar el 100 % de efectividad, el máximo porcentaje de efectividad es 64.3 %, que se obtiene con un factor de aprendizaje de 0.1 y con 110 neuronas en la capa oculta.

Tabla 6.2 Resultados de entrenamiento con una muestra del 20 % del conjunto de datos de entrenamiento

LR \ #N	30	40	50	60	70	80	90	100	110	120
0.1	42.5	67.0	56.7	64.8	67.7	66.4	67.2	69.5	70.3	70.1
0.2	38.3	38.0	37.4	52.4	48.2	50.5	52.2	58.6	61.7	60.3
0.3	24.7	23.8	36.0	32.9	39.4	32.9	40.9	43.8	47.7	46.9
0.4	21.3	19.4	28.2	28.6	23.8	35.5	34.8	33.1	32.3	37.0
0.5	19.1	19.0	21.7	25.5	29.4	26.3	28.6	28.6	28.1	32.6

En la Tabla 6.2 se refleja el entrenamiento realizado con el 20 % del conjunto de datos de entrenamiento, el porcentaje máximo alcanzado es de 70.3 %, mismo que se obtiene con un factor de aprendizaje de 0.1 y con 110 neuronas en la capa oculta. En la Tabla 6.3 se puede observar el enternamiento realizado con el 30 % del conjunto de datos de entrenamiento, donde

Tabla 6.3 Resultados de entrenamiento con una muestra del 30 % del conjunto de datos de entrenamiento

LR \ #N	30	40	50	60	70	80	90	100	110	120
0.1	37.8	60.4	67.5	72.8	68.8	70.3	69.3	74.2	70.8	70.9
0.2	29.8	37.9	56.1	50.9	64.1	58.5	66.7	64.0	63.9	63.3
0.3	26.2	31.2	41.9	52.6	46.3	45.9	49.8	54.9	50.9	51.8
0.4	26.3	28.9	40.8	32.3	32.0	45.4	41.8	41.0	51.1	48.8
0.5	18.4	37.3	23.4	28.9	31.9	29.7	44.8	53.8	36.0	38.8

el máximo porcentaje de efectividad es de 74.2 %, el cual se alcanza con un factor de aprendizaje de 0.1 y con 100 neuronas en la capa oculta.

Como los resultados obtenidos al realizar el entrenamiento de la red neuronal con 51 características, para identificar las 21 señas estáticas del alfabeto del lenguaje mexicano de señas, no se acercaron a un porcentaje de efectividad cercano al 100 %, se decidió hacer una prueba reduciendo el número de señas que la red neuronal puede identificar, con el fin de obtener resultados más favorables.

Se iniciaron las pruebas de entrenamiento de la red neuronal, para que pudiera reconocer las primeras 6 letras del alfabeto del lenguaje mexicano de señas (letras de la a a la f) que pueden ser obervadas en la Figura 6.2.

Tabla 6.4 Resultados del entrenamiento para 6 letras del alfabeto con 51 características

%	10 (%)		20 (%)		30 (%)	
LR \ #N	30	40	30	40	30	40
0.1	27.14	36.33	47.89	55.74	48.87	100.0
0.2	26.51		28.74		25.05	
0.3	14.29		28.53		28.37	
0.4	25.95		19.35		23.81	
0.5	14.29		19.05		23.81	

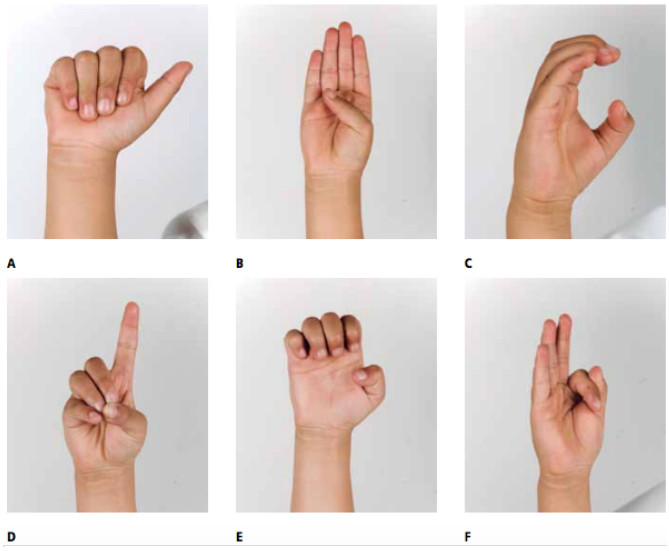

Figura 6.2 6 primeras letras del alfabeto del lenguaje mexicano de señas.

Se puede observar en la Tabla 6.4 el resultado del entrenamiento con muestras de 10 %, 20 % y 30 % del tamaño del conjunto de datos de entrenamiento, con redes neuronales con 30 y 40 neuronas en la capa oculta, como se reconocen 6 señas con este experimento, la capa de salida consta de 6 neuronas. Además se puede observar, en la tabla ya mencionada, que el entrenamiento fue interrumpido al momento de alcanzar el máximo porcentaje posible de efectividad (100 %).

6.2.2. Pruebas de la herramienta

Después de obtener un 100 % en el entrenamiento con 40 neuronas, se procedió a probar el funcionamiento de la red neuronal con la herramienta de software que utiliza a la red neuronal ya entrenada para hacer la traducción de las señas a texto.

Dicha prueba se realizó utilizando al mismo sujeto que proporcionó los datos para hacer el entrenamiento de la red neuronal, de cada una de las 6 letras, que puede reconocer la red neuronal ya mencionada, se tomaron 5 muestras y los resultados se muestran en la Tabla 6.5.

Tabla 6.5 Matriz de confusión de los resultados de las pruebas para 6 letras del alfabeto con 51 características

Letra \ Letra	a	b	c	d	e	f
a	0	0	0	0	0	5
b	0	5	0	0	0	0
c	0	0	5	0	0	0
d	0	0	0	5	0	0
e	0	0	0	0	5	0
f	0	0	0	0	5	0

Los resultados de la pruebas muestran que la red neuronal es capaz de reconocer muy bien las letras *b, c, d, e* pero tiene un desempeño malo al no poder reconocer en ninguna ocasión correctamente las letras *a* y *f*. En general se puede decir que la efectividad de esta red neuronal al momento de hacer la traducción de señas en tiempo real es del 66 %.

6.3. Conjunto de datos de 33 características

Debido a los resultados mencionados en la sección anterior se procedió a generar un nuevo conjunto de datos que pudiera elevar la efectividad de reconocimiento de señas, para lo cual se realizó un análisis de las 51 características que se tomaron en cuenta en el experimento anterior.

Se encontró que al tomar el cuenta los ángulos entre las mismas falanges de dedos contiguos, se estaba generando información redundante que muy probablemente estaba causando ruido al momento de hacer la traducción de las señas.

Para disminuir dicho problema se consideró solo utilizar un ángulo entre falanges de dedos contiguos. La falange con la cual se hace la medición del ángulo es la falange media, se selecciona dicha falange por que en el dedo pulgar la falange proximal es la que aporta mejor información de la posición de dicho dedo y como se explicó en la sección 3.2.1 el modelo de la mano utilizado por *leap motion* en el dedo pulgar, no corresponde al modelo anatómico estándar y la falange proximal del dedo pulgar es considerada como la falange media.

Al eliminar los datos que estaban generando ruido, el nuevo conjunto de datos se redujo a 33 características. Además para este experimento ya se utilizan las 21 señas estáticas del lenguaje mexicano de señas.

6.3.1. Entrenamiento de la red neuronal

Los primeros entrenamientos con el nuevo conjunto de datos de 33 características arrojaron resultados por debajo del 90 % de efectividad, cuando la red neuronal puede reconocer las 21 señas estáticas. Por esta razón se decidió seccionar a las 21 señas en 3 grupos, un grupo se hace cargo de reconocer las señas que se realizan con la palma de frente al sensor, el segundo grupo contiene a las señas que se ejecutan con la palma de costado con respecto al sensor, siendo el dedo meñique el más cercano al sensor y por último un grupo que identifica a las señas que se realizan con la palma opuesta al sensor, tal como se describe en la sección 4.

El grupo de señas que se realizan con la palma de frente es el grupo con mayor número de señas con un total de 11, por ser el grupo más grande y como refieren [33, 35, 32] al tener un mayor número de objetivos que identificar es más difícil acercarse al mínimo global de la red neuronal para garantizar una mejor eficiencia, se toma como referencia éste grupo para iniciar el experimento de entrenamiento.

En este entrenamiento la red neuronal consta de 33 neuronas en la capa de entrada y la capa se salida contiene 11 neuronas. Para obtener el valor óptimo de la capa oculta se realizaron diversos experimentos donde se varía el número de neuronas en la capa oculta, dichos valores van desde 30 neuronas hasta 120, además se consideran valores del facto de aprendizaje desde 0.1 hasta 0.5 y los tamaños de muestra del conjunto de datos de entrenamiento son del 10 %, 20 % y 30 %.

En la Tabla 6.6 se puede observar que con entrenamientos que utilizan el 10 % del conjunto de datos de entrenamiento se obtienen redes que alcanzan el 100 % de efectividad para reconocer las 11 señas, dicho valor lo alcanzan 2 redes, una de ellas tiene 80 neuronas en su capa oculta y fue entrenada con un factor de aprendizaje de 0.1, la otra red tiene 120 neuronas en su capa oculta y fue entrenada con un facto de aprendizaje de 0.1.

Tabla 6.6 Resultados de entrenamiento con una muestra del 10 % del conjunto de datos de entrenamiento

LR \ #N	30	40	50	60	70	80	90	100	110	120
0.1	78.6	88.4	90.9	98.7	99.0	100	98.7	99.5	99.1	100
0.2	54.5	53.3	98.7	79.3	94.1	90.9	96.5	97.7	99.9	90.9
0.3	51.8	54.5	53.3	72.7	78.7	81.8	79.8	72.4	96.8	88.9
0.4	45.4	70.1	54.5	60.0	81.8	79.0	81.8	89.0	78.4	80.5
0.5	45.4	45.4	44.1	45.3	61.6	72.7	69.5	79.0	72.7	63.6

Tabla 6.7 Resultados de entrenamiento con una muestra del 20 % del conjunto de datos de entrenamiento

LR \ #N	30	40	50	60	70	80	90	100	110	120
0.1	97.3	81.8	100	98.1	100	99.8	100	100	100	97.9
0.2	52.0	90.9	87.8	98.5	100	97.5	98.6	100	98.0	100
0.3	62.6	62.3	86.5	79.1	88.9	100	89.3	97.4	97.5	98.7
0.4	60.5	70.0	72.7	61.6	90.5	87.5	87.6	90.9	88.2	98.0
0.5	43.9	70.9	54.5	72.4	62.9	78.8	81.8	81.8	79.1	79.6

En la Tabla 6.7 se puede observar que con entrenamientos que utilizan el 20 % del conjunto de datos de entrenamiento se obtienen redes que alcanzan el 100 % de efectividad para reconocer las 11 señas, dicho valor lo alcanzan 9 redes, 5 de ellas fueron entrenadas con un factor de aprendizaje de 0.1 y cuentan con 50, 70, 90, 100 y 110 neuronas en su capa oculta respectivamente. Otras 3 redes fueron entrenadas con un factor de aprendizaje de 0.2 y contienen 70, 100 y 120 neuronas en su capa oculta, por último una red fue entrenada con un factor de aprendizaje de 0.3 y tiene en su capa oculta 80 neuronas.

En la Tabla 6.8 se puede observar que con entrenamientos que utilizan el 30 % del conjunto de datos de entrenamiento se obtienen redes que alcanzan el 100 % de efectividad para reconocer las 11 señas, dicho valor lo alcanzan 7 redes, 5 de ellas fueron entrenadas con un factor de aprendizaje de 0.1 y cuentan con 40, 90, 100, 110 y 120 neuronas en su capa oculta, otras 2 redes fueron entrenadas con un factor de aprendizaje de 0.2 y tienen en

Tabla 6.8 Resultados de entrenamiento con una muestra del 30 % del conjunto de datos de entrenamiento

LR \ #N	30	40	50	60	70	80	90	100	110	120
0.1	81.8	100	98.9	99.0	98.9	98.5	100	100	100	100
0.2	63.6	90.7	100	90.9	97.7	100	98.1	98.1	98.7	98.1
0.3	80.3	72.7	81.8	79.5	78.9	98.1	81.8	96.7	90.9	98.7
0.4	41.9	69.4	53.4	81.8	79.7	90.5	89.8	89.8	88.1	90.1
0.5	47.0	54.5	63.6	72.7	88.1	89.8	90.9	97.7	96.9	90.9

su capa oculta 50 y 80 neuronas respectivamente.

6.3.2. Pruebas de la herramienta

Después de obtener un 100 % en el entrenamiento de varias redes neuronales, se procedió a probar el funcionamiento de la red neuronal con la herramienta de software se debe utilizar una red neuronal para hacer la traduccón de las señas a texto. Como hubo varias redes neuronales que alcanzaron el 100 % de efectividad en los entrenamientos, se escoge la red neuronal con menor número de neuronas en la capa oculta, debido a que esto representa un menor costo computacional al tener menos conexiones, por esta razón se utiliza la red neuronal con 40 neuronas en la capa oculta.

Dicha prueba se realizó utilizando el mismo sujeto que proporcionó los datos para hacer el entrenamiento de la red neuronal, de cada una de las 11 letras, que puede reconocer la red neuronal ya mencionada, se tomaron 5 muestras y los resultados se muestran en la Tabla 6.9.

Los resultados de la pruebas muestran que la red neuronal es capaz de reconocer muy bien a las letras *a, b, d, e, l, s, u* y *w* pero tiene problemas para reconocer correctamente a las letras *r, t* y *v*. En general se puede decir que la efectividad de esta red neuronal al momento de hacer la traducción de señas en tiempo real es del 87 %.

6.4. Conjunto de datos de 18 características

Como con el conjunto de datos de 33 características se obtuvieron buenos resultados, se realizaron nuevos experimentos con el objetivo de mejorar la

Tabla 6.9 Matriz de confusión de los resultados de las pruebas para 11 letras del alfabeto con 33 características

Letra \ Letra	a	b	d	e	l	r	s	t	u	v	w
a	5	0	0	0	0	0	0	0	0	0	0
b	0	5	0	0	0	0	0	0	0	0	0
d	0	0	5	0	0	0	0	0	0	0	0
e	0	0	0	5	0	0	0	0	0	0	0
l	0	0	0	0	5	0	0	0	0	0	0
r	0	0	0	0	0	3	0	0	2	0	0
s	0	0	0	0	0	0	5	0	0	0	0
t	0	0	0	0	0	0	2	3	0	0	0
u	0	0	0	0	0	0	0	0	5	0	0
v	0	0	0	0	0	0	0	0	2	3	0
w	0	0	0	0	0	0	0	0	0	1	4

eficiencia del reconocimiento de señas y también de disminuir el número de características que toma la red neuronal como datos de entrada. Con el fin de reducir el costo computacional tanto del entrenamiento de la red neuronal como de la herramienta de software que traduce las señas en tiempo real.

Se observó que los ángulos que se forman entre 2 falanges contiguas, también pueden interpretados por el ángulo que se forma entre la falange y el vector normal de la palma. Por ésta razón se idearon 2 experimentos, uno en el cual sólo se toman en cuenta los ángulos formados por 2 falanges contiguas y el otro experimento sólo toma en cuenta los ángulos formados entre las falanges y el vector normal de la palma.

En esta sección se describe el experimento que utiliza como características los ángulos formados entre 2 falanges contiguas, así como los ángulos que se forman entre 2 dedos contiguos utilizando como referencia de dicho ángulo la falange media de los 2 dedos contiguos.

6.4.1. Entrenamiento de la red neuronal

Como se describe en la sección 6.3 se sigue la estructura de seccionar las 21 señas estáticas del lenguaje de señas mexicano, por tal motivo a continuación se muestran los resultados que se obtuvieron al entrenar la red

neuronal que puede identificar 11 señas.

Tabla 6.10 Resultados de entrenamiento con una muestra del 10 % del conjunto de datos de entrenamiento

LR \ #N	30	40	50	60	70	80	90	100	110	120
0.1	90.9	83.5	90.3	83.2	87.8	83.5	83.2	85.9	90.9	85.9
0.2	90.9	89.2	90.9	90.9	85.5	83.5	88.2	88.1	86.0	84.2
0.3	90.9	90.8	88.3	84.1	88.7	84.9	86.0	90.9	85.2	88.0
0.4	90.9	90.9	90.9	88.8	88.4	88.2	88.5	89.7	89.3	89.3
0.5	90.9	90.9	90.9	90.9	90.9	88.7	90.9	85.6	90.5	88.4

Tabla 6.11 Resultados de entrenamiento con una muestra del 20 % del conjunto de datos de entrenamiento

LR \ #N	30	40	50	60	70	80	90	100	110	120
0.1	86.0	90.9	83.5	83.2	90.9	89.1	83.7	84.1	83.9	83.5
0.2	88.4	86.1	90.9	83.6	87.0	83.2	84.1	83.5	89.4	90.9
0.3	86.0	88.2	90.9	86.0	87.8	87.5	90.9	84.0	83.5	89.1
0.4	90.9	89.2	86.2	90.9	88.1	88.7	90.5	90.9	89.0	86.0
0.5	84.5	89.7	85.3	90.9	84.2	87.5	86.0	85.4	88.4	88.1

Tabla 6.12 Resultados de entrenamiento con una muestra del 30 % del conjunto de datos de entrenamiento

LR \ #N	30	40	50	60	70	80	90	100	110	120
0.1	90.9	90.9	83.0	84.1	84.6	83.7	83.4	83.7	87.6	83.0
0.2	89.0	89.8	90.9	90.9	89.7	86.0	84.1	87.2	89.2	88.1
0.3	89.1	83.2	89.3	86.2	83.5	83.9	84.2	83.5	82.9	85.7
0.4	85.9	83.7	83.3	86.0	89.8	90.9	83.7	88.9	89.1	90.9
0.5	87.3	86.2	88.8	83.9	84.4	86.8	88.7	90.9	85.9	87.6

Como se puede observar en las Tablas 6.10, 6.11 y 6.12 no se consigue el 100 % de efectividad en los entrenamientos, es por ello que se descarta el conjunto de datos de 18 características y por la misma razón no se realizan pruebas con la herramienta de traducción.

6.5. Conjunto de datos de 19 características

Este conjunto de datos está formado por los ángulos que se forman entre las falanges y el vector normal de la palma, así como de los ángulos que se forman entre 2 dedos contiguos tomando como referencia la misma falange en cada dedo para calcular el ángulo.

6.5.1. Entrenamiento de la red neuronal

A continuación se muestran los resultados que se obtuvieron al entrenar la red neuronal que puede identificar 11 señas.

En la Tabla 6.13 se puede observar que una sola red alcanza el 100 % de

Tabla 6.13 Resultados de entrenamiento con una muestra del 10 % del conjunto de datos de entrenamiento

LR \ #N	30	40	50	60	70	80	90	100	110	120
0.1	63.2	86.7	90.9	88.2	98.3	90.8	96.8	97.8	100	99.2
0.2	45.4	63.6	68.7	63.6	89.7	88.4	88.9	89.6	90.8	94.9
0.3	36.3	36.3	44.2	63.1	68.9	78.4	87.4	72.7	95.3	77.4
0.4	36.3	51.8	54.5	45.4	71.8	59.7	60.6	63.6	60.9	81.4
0.5	36.3	27.2	45.3	45.2	54.5	51.6	54.5	54.3	70.1	63.2

efectividad, misma que tiene 110 neuronas en su capa oculta y fue entrenada con un factor de aprendizaje de 0.1.

En la Tabla 6.14 se puede observar que con entrenamientos que utilizan el 20 % del conjunto de datos de entrenamiento se obtienen redes que alcanzan el 100 % de efectividad para reconocer las 11 señas. Dicho valor lo

Tabla 6.14 Resultados de entrenamiento con una muestra del 20 % del conjunto de datos de entrenamiento

LR \ #N	30	40	50	60	70	80	90	100	110	120
0.1	54.9	87.7	98.1	86.4	91.3	99.7	100	99.9	99.9	100
0.2	54.2	72.7	72.9	73.8	85.3	90.9	99.0	100	99.9	93.8
0.3	54.3	54.5	54.5	63.6	63.5	86.9	89.4	96.7	99.8	81.8
0.4	45.4	54.4	45.4	63.6	81.8	63.5	90.9	90.5	79.8	78.9
0.5	45.4	54.5	45.4	63.5	72.7	66.8	69.1	72.6	89.4	87.3

alcanzan 3 redes, 2 de ellas fueron entrenadas con un factor de aprendizaje de 0.1 y cuentan con 90 y 120 neuronas en su capa oculta respectivamente; por último una red fue entrenada con un factor de aprendizaje de 0.2 y tiene en su capa oculta 100 neuronas.

Tabla 6.15 Resultados de entrenamiento con una muestra del 30 % del conjunto de datos de entrenamiento

LR \ #N	30	40	50	60	70	80	90	100	110	120
0.1	72.6	78.3	100	99.6	97.8	100	99.3	99.8	100	99.9
0.2	54.5	68.2	81.6	90.2	88.7	98.9	99.4	100	99.9	99.8
0.3	63.6	72.7	70.2	88.2	81.8	99.3	95.9	87.0	97.7	87.4
0.4	47.1	51.1	70.4	72.0	80.8	97.6	72.7	99.7	97.0	96.2
0.5	44.9	39.5	60.1	54.5	81.9	72.6	68.7	81.7	78.6	72.7

En la Tabla 6.15 se puede observar que con entrenamientos que utilizan el 30 % del conjunto de datos de entrenamiento se obtienen redes que alcanzan el 100 % de efectividad para reconocer las 11 señas, dicho valor lo alcanzan 4 redes, 3 de ellas fueron entrenadas con un factor de aprendizaje de 0.1 y cuentan con 50, 80 y 110 neuronas en su capa oculta respectivamente; por último una red fue entrenada con un factor de aprendizaje de 0.2 y tiene en su capa oculta 100 neuronas.

6.5.2. Pruebas de la herramienta

Una vez que se obtuvieron redes neuronales con 100 % de eficiencia para identificar las 11 señas, se procede a realizar el experimento de hacer la traducción de las señas a texto, como en este último experimento es el que considera como definitivo para efectos de este trabajo, se procede a mostrar los resultados que se obtuvieron de 3 personas diferentes al sujeto que proporcionó las señas para generar los datos con los que se entrenó a las redes neuronales.

A continuación se muestra la Tabla 6.16, donde se pueden observar las medidas de la palma y de cada uno de los dedos de cada sujeto que realizó las señas, dichas medidas fueron obtenidas desde el sensor *leap motion* y están representadas en milímetros.

Tabla 6.16 Medidas de los 3 sujetos que realizaron la prueba de la herramienta de traducción

Sujeto / Parte	sujeto1	sujeto2	sujeto3
palma	86.44	90.18	84.46
pulgar	102.04	106.46	99.70
índice	149.89	156.39	146.46
medio	156.93	163.73	153.34
anular	146.02	152.34	142.67
meñique	123.63	128.98	120.80

De acuerdo a los resultados obtenidos en la Tabla 6.17 se puede observar que el sujeto1 no obtuvo buenos resultados cuando trató de hacer los gestos de las letras *e, s, t* y *w*. Donde el promedio de efectividad de traducción de la herramienta es de un 81 %.

Tabla 6.17 Matriz de confusión de los resultados de las pruebas para 11 letras del alfabeto con 19 características, realizadas por el sujeto 1

Letra \ Letra	a	b	d	e	l	r	s	t	u	v	w
a	5	0	0	0	0	0	0	0	0	0	0
b	0	5	0	0	0	0	0	0	0	0	0
d	0	0	5	0	0	0	0	0	0	0	0
e	0	1	0	4	0	0	0	0	0	0	0
l	0	0	0	0	5	0	0	0	0	0	0
r	0	0	0	0	0	5	0	0	0	0	0
s	2	0	0	0	0	0	3	0	0	0	0
t	0	0	0	0	0	0	3	2	0	0	0
u	0	0	0	0	0	0	0	0	5	0	0
v	0	0	0	0	0	0	0	0	0	5	0
w	0	0	0	0	0	0	0	0	0	3	2

En relación a los resultados obtenidos en la Tabla 6.18 se puede observar que el sujeto2 no obtuvo buenos resultados cuando trató de hacer los gestos de las letras *r, t* y *v*. Donde el promedio de efectividad de traducción de la herramienta es de un 89 %.

Tabla 6.18 Matriz de confusión de los resultados de las pruebas para 11 letras del alfabeto con 19 características, realizadas por el sujeto 2

Letra \ Letra	a	b	d	e	l	r	s	t	u	v	w
a	5	0	0	0	0	0	0	0	0	0	0
b	0	5	0	0	0	0	0	0	0	0	0
d	0	0	5	0	0	0	0	0	0	0	0
e	0	0	0	5	0	0	0	0	0	0	0
l	0	0	0	0	5	0	0	0	0	0	0
r	0	0	0	0	0	3	0	0	2	0	0
s	0	0	0	0	0	0	5	0	0	0	0
t	0	0	0	0	0	0	3	2	0	0	0
u	0	0	0	0	0	0	0	0	5	0	0
v	0	0	0	0	0	0	0	0	1	4	0
w	0	0	0	0	0	0	0	0	0	0	5

Conforme a los resultados obtenidos en la Tabla 6.19 se puede observar que el sujeto2 no obtuvo buenos resultados cuando trató de hacer los gestos de las letras *t, v* y *w*. Donde el promedio de efectividad de traducción de la herramienta es de un 89 %.

Tabla 6.19 Matriz de confusión de los resultados de las pruebas para 11 letras del alfabeto con 19 características, realizadas por el sujeto 3

Letra \ Letra	a	b	d	e	l	r	s	t	u	v	w
a	5	0	0	0	0	0	0	0	0	0	0
b	0	5	0	0	0	0	0	0	0	0	0
d	0	0	5	0	0	0	0	0	0	0	0
e	0	0	0	5	0	0	0	0	0	0	0
l	0	0	0	0	5	0	0	0	0	0	0
r	0	0	0	0	0	5	0	0	0	0	0
s	0	0	0	0	0	0	5	0	0	0	0
t	0	0	0	0	0	0	2	3	0	0	0
u	0	0	0	0	0	0	0	0	5	0	0
v	0	0	0	0	0	0	0	0	2	3	0
w	0	0	0	0	0	0	0	0	4	0	1

De acuerdo a los resultados obtenidos en las Tablas 6.17, 6.18 y 6.19 se puede observar que sólo existen problemas para traducir las letras e, r, s, t, v y w. Donde sacando el promedio de las 3 pruebas se obtiene un 86 % de efectividad para traducir señas realizadas por personas que no están dentro del conjunto de datos con el que fue entrenada la red neuronal.

Ahora se muestra la Tabla 6.20 donde se muestran los resultados de las pruebas realizadas con la red neuronal que puede reconocer las señas que se realizan de costado con respecto a la posición del sensor. Dicha tabla contiene los resultados de los 3 sujetos con los cuales se probó la efectividad de reconocimiento de la herramienta de software.

Como se puede observar en la Tabla 6.20, sólo existen algunos conflictos al tratar de traducir las letras *f* y *o*, donde el promedio de efectividad de la herramienta de software en este caso es del 85 %.

Tabla 6.20 Matriz de confusión de los resultados de las pruebas para 6 letras de costado del alfabeto con 33 características

Letra \ Letra	c	f	i	o	p
c	15	0	0	0	0
f	0	9	0	0	6
i	0	0	15	0	0
o	5	0	0	10	0
p	0	0	0	0	15

En la Tabla 6.21 donde se muestran los resultados de las pruebas realizadas con la red neuronal que puede reconocer las señas que se realizan con la palma opuesta con respecto a la posición del sensor. Dicha tabla contiene los resultados de los 3 sujetos con los cuales se probó la efectividad de reconocimiento de la herramienta de software.

Tabla 6.21 Matriz de confusión de los resultados de las pruebas para 6 letras del alfabeto con 51 características

Letra \ Letra	g	h	m	n	y
g	15	0	0	0	0
h	0	15	0	0	0
m	0	0	15	0	0
n	0	0	6	9	0
y	0	0	0	0	15

Como se puede observar en la Tabla 6.21, sólo existen algunos conflictos al tratar de traducir la letra *n*, donde el promedio de efectividad de la herramienta de software en este caso es del 92 %.

Una vez obtenidos todos los porcentajes de efectividad de la herramienta de software reconociendo las 21 letras estáticas del lenguaje de señas mexicano, se puede obtener el promedio de efectividad de reconocimiento de dichas señas, el cual es de 87.6 %.

Capítulo 7

Conclusiones y trabajo futuro

En este trabajo se propone un conjunto de datos conformado sólo por ángulos que ayudan a detectar las posiciones de las falanges de los dedos de la mano derecha cuando una seña del alfabeto del lenguaje mexicano de señas es realizado, durante el desarrollo de las pruebas se puede observar como se fue refinando el proceso de selección de datos.

Primero se inició con un conjunto de datos que contemplaban 490 características, a las que se les aplicó una prueba estadística para determinar que características aportaban información relevante para reconocer señas, aquellas que no aportaban información relevante fueron eliminadas del conjunto de datos.

El conjunto de datos se redujo a 51 características las cuales fueron utilizadas para entrenar una red neuronal tipo perceptrón multicapa que pudiera reconocer las 21 señas estáticas del alfabeto del lenguaje mexicano de señas, en donde las pruebas de entrenamiento reflejaron que el conjunto de datos no aporta información específica para que dichas señas fueran clasificadas correctamente por la red neuronal.

Se procedió a hacer una partición en el número de señas que identifica la red neuronal, esto con el fin de aumentar la efectividad de reconocimiento por parte de la red neuronal. Se hizo una prueba tomando 6 señas del alfabeto, con dicha partición en la prueba de entrenamiento, la red neuronal alcanzó el 100 % de efectividad, después de procedió a probar la red neuronal en la

herramienta de software que realiza la traducción de señas a texto, los resultados no fueron los esperados ya que sólo se alcanzó el 66 % de efectividad en el reconocimiento de las 6 señas.

Se realizó un análisis de las características tomadas en cuenta para el reconocimiento de los gestos, el cual arrojó que los ángulos entre los dedos estaba generando algunos problemas para la identificación de las señas, por eso en lugar de tomar todas las falanges de los dedos para medir dicho ángulo, sólo se toma en consideración una falange, lo cual redujo el conjunto de características a 33.

Se realizaron las pruebas de entrenamiento con el nuevo conjunto de características, mismas que dieron como resultado el 100 % de efectividad con varios parámetros de entrenamiento, después se realizó la prueba de la herramienta de traducción, la cual arrojó un 87 % de efectividad.

Siguiendo con el análisis del conjunto de datos, se encontró que los ángulos entre falanges y los ángulos entre una falange y el vector normal de la palma, por separado, pueden dar suficiente información para realizar el reconocimiento de gestos, para probar dicha afirmación se realizaron las pruebas de entrenamiento de los 2 conjuntos de datos.

El conjunto de datos con sólo ángulos entre falanges, consta de 18 características, en los resultados de sus pruebas de entrenamiento, se observa que no son suficientes las características para reconocer gestos, por ello, este conjunto de datos fue descartado.

El conjunto de datos que contempla los ángulos entre las falanges y el vector normal de la palma obtuvo el 100 % de efectividad en las pruebas de entrenamiento, por ello se realizaron pruebas con la herramienta de traducción con 3 personas con diferentes tamaños de mano, dichas personas no participaron para generar los conjuntos de datos de entrenamiento y de prueba. Los resultados de las pruebas de la herramienta muestran un promedio de 87.6 % de efectividad para reconocer gestos.

Con estos resultados podemos concluir que se puede realizar el reconocimiento de gestos con un conjunto de características que sólo contempla ángulos, donde las pruebas de entrenamiento alcanzan el 100 % de efectividad y las pruebas de la herramienta de traducción llegan a un 87.6 % de efectividad.

La utilización de ángulos muestra que se usan características que son independientes del tamaño de la mano que genera los conjuntos de datos de entrenamiento y prueba, además como se vió en las pruebas de la herramienta de traducción, dicha herramienta es capaz de reconocer gestos realizadas con manos diferentes a la que generó los conjuntos de datos para entrenar la red neuronal.

Con estos resultados podemos decir que el conjunto de datos propuesto si es una buena generalización del problema tratado en este proyecto.

Como trabajo futuro se proponen algunas acciones:

- Obtener datos de señas con diferentes tamaños para que sean utilizados para entrenar la red neuronal.
- Aplicar técnicas de reconocimiento de trayectoria para realizar la traducción de gestos dinámicos.
- Realizar la operación inversa a este trabajo, es decir, traducir de texto a señas para obtener un traductor bidireccional.
- Realizar una aplicación educativa para enseñar a utilizar el lenguaje de señas mexicano.

Bibliografía

[1] K. Erdoğan, A. Durdu, and N. Yilmaz, "Intention recognition using leap motion controller and artificial neural networks," in *2016 International Conference on Control, Decision and Information Technologies (CoDIT)*, pp. 689–693, April 2016.

[2] J. S. Artal-Sevil and J. L. Montañés, "Development of a robotic arm and implementation of a control strategy for gesture recognition through leap motion device," in *2016 Technologies Applied to Electronics Teaching (TAEE)*, pp. 1–9, June 2016.

[3] G. Du, P. Zhang, and X. Liu, "Markerless human x2013;manipulator interface using leap motion with interval kalman filter and improved particle filter," *IEEE Transactions on Industrial Informatics*, vol. 12, pp. 694–704, April 2016.

[4] A. Škraba, A. Kološvari, D. Kofjač, and R. Stojanović, "Wheelchair maneuvering using leap motion controller and cloud based speech control: Prototype realization," in *2015 4th Mediterranean Conference on Embedded Computing (MECO)*, pp. 391–394, June 2015.

[5] P. Rittitum, W. Vatanawood, and A. Thongtak, "Digital scrum board using leap motion," in *2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS)*, pp. 1–4, June 2016.

[6] N. P. John, K. A. S. Pillai, and A. R. Anil, "Video interaction through finger tips," in *Circuit, Power and Computing Technologies (ICCPCT), 2015 International Conference on*, pp. 1–4, March 2015.

[7] G. Zhu, S. Cai, Y. Ma, and E. Liu, "A series of leap motion-based matching games for enhancing the fine motor skills of children with autism," in *2015 IEEE 15th International Conference on Advanced Learning Technologies*, pp. 430–431, July 2015.

[8] M. A. Rahman, M. Ahmed, A. Qamar, D. Hossain, and S. Basalamah, "Modeling therapy rehabilitation sessions using non-invasive serious games," in *Medical Measurements and Applications (MeMeA), 2014 IEEE International Symposium on*, pp. 1–4, June 2014.

[9] I. Perdana, "Teaching elementary school students new method of music performance with leap motion," in *Virtual Systems Multimedia (VSMM), 2014 International Conference on*, pp. 273–277, Dec 2014.

[10] A. Qamar, M. A. Rahman, and S. Basalamah, "Adding inverse kinematics for providing live feedback in a serious game-based rehabilitation system," in *2014 5th International Conference on Intelligent Systems, Modelling and Simulation*, pp. 215–220, Jan 2014.

[11] B. Demircioğlu, G. Bülbül, and H. Köse, "Turkish sign language recognition with leap motion," in *2016 24th Signal Processing and Communication Application Conference (SIU)*, pp. 589–592, May 2016.

[12] F. Soares, J. S. Esteves, V. Carvalho, C. Moreira, and P. Lourenço, "Sign language learning using the hangman videogame," in *Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), 2015 7th International Congress on*, pp. 231–234, Oct 2015.

[13] K. R. Moser and J. E. Swan, "Evaluation of hand and stylus based calibration for optical see-through head-mounted displays using leap motion," in *2016 IEEE Virtual Reality (VR)*, pp. 233–234, March 2016.

[14] S. M, S. Rakesh, S. Gupta, S. Biswas, and P. P. Das, "Real-time hands-free immersive image navigation system using microsoft kinect 2.0 and leap motion controller," in *2015 Fifth National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics (NCVPRIPG)*, pp. 1–4, Dec 2015.

[15] J. M. Suárez, "Interaction techniques comparison for leap motion in vrcollage app: Comparison between clutching, go-go and selection techniques," in *Computing Colombian Conference (10CCC), 2015 10th*, pp. 189–193, Sept 2015.

[16] J. Fillwalk, "Chromachord: A virtual musical instrument," in *3D User Interfaces (3DUI), 2015 IEEE Symposium on*, pp. 201–202, March 2015.

[17] A. S. Elons, M. Ahmed, H. Shedid, and M. F. Tolba, "Arabic sign language recognition using leap motion sensor," in *Computer Engineering Systems (ICCES), 2014 9th International Conference on*, pp. 368–373, Dec 2014.

[18] M. Mohandes, S. Aliyu, and M. Deriche, "Arabic sign language recognition using the leap motion controller," in *2014 IEEE 23rd International Symposium on Industrial Electronics (ISIE)*, pp. 960–965, June 2014.

[19] C. G. Ching-Hua Chuan, Erick Regina, "American sign language recognition using leap motion sensor," in *2014 13th International Conference on Machine Learning and Applications (ICMLA)*, pp. 541–544, 2014.

[20] G. Marin, F. Dominio, and P. Zanuttigh, "Hand gesture recognition with leap motion and kinect devices," in *2014 IEEE International Conference on Image Processing (ICIP)*, pp. 1565–1569, Oct 2014.

[21] Y. Chen, Z. Ding, Y. L. Chen, and X. Wu, "Rapid recognition of dynamic hand gestures using leap motion," in *Information and Automation, 2015 IEEE International Conference on*, pp. 1419–1424, Aug 2015.

[22] P. Chavan, T. Ghorpade, and P. Padiya, "Indian sign language to forecast text using leap motion sensor and rf classifier," in *2016 Symposium on Colossal Data Analysis and Networking (CDAN)*, pp. 1–5, March 2016.

[23] J. K. Sharma, R. Gupta, and V. K. Pathak, "Numeral gesture recognition using leap motion sensor," in *2015 International Conference on Computational Intelligence and Communication Networks (CICN)*, pp. 411–414, Dec 2015.

[24] V. Athitsos, Q. Yuan, J. Alon, and S. Sclaroff, "A unified framework for gesture recognition and spatiotemporal gesture segmentation," *IEEE Transactions on Pattern Analysis Machine Intelligence*, vol. 31, no. undefined, pp. 1685–1699, 2008.

[25] K. Y. Fok, N. Ganganath, C. T. Cheng, and C. K. Tse, "A real-time asl recognition system using leap motion sensors," in *Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2015 International Conference on*, pp. 411–414, Sept 2015.

[26] S. Aliyu, M. Mohandes, M. Deriche, and S. Badran, "Arabie sign language recognition using the microsoft kinect," in *2016 13th Internatio-*

nal Multi-Conference on Systems, Signals Devices (SSD), pp. 301–306, March 2016.

[27] P. G. Ahire, K. B. Tilekar, T. A. Jawake, and P. B. Warale, "Two way communicator between deaf and dumb people and normal people," in *Computing Communication Control and Automation (ICCUBEA), 2015 International Conference on*, pp. 641–644, Feb 2015.

[28] P. Gupta, G. Joshi, and M. Dutta, "Comparative analysis of movement and tracking techniques for indian sign language recognition," in *2015 Fifth International Conference on Advanced Computing Communication Technologies*, pp. 90–95, Feb 2015.

[29] N. Singh, N. Baranwal, and G. C. Nandi, "Implementation and evaluation of dwt and mfcc based isl gesture recognition," in *2014 9th International Conference on Industrial and Information Systems (ICIIS)*, pp. 1–7, Dec 2014.

[30] J. G. G. J. M. P. S. Tomažič and J. Sodnik, "An analysis of the precision and reliability of the leap motion sensor and its suitability for static and dynamic tracking," *Sensors*, vol. 14, no. 2, p. 3702, 2014.

[31] F. Weichert, D. Bachmann, B. Rudak, and D. Fisseler, "Analysis of the accuracy and robustness of the leap motion controller," *Sensors*, vol. 13, no. 5, p. 6380, 2013.

[32] R. Rojas, *Neural Networks : A Systematic Introduction*. Springer, 1996.

[33] M. A. Nielsen, *Neural Networks and Deep Learning*. Determination Press, 2015.

[34] J. L. G. Rosa, *Artificial Neural Networks - Architectures and Applications*. InTech, 2013.

[35] B. Kröse, B. Krose, P. van der Smagt, and P. Smagt, *An introduction to Neural Networks*. 1993.

[36] S. M. Shamsuddin, A. O. Ibrahim, and C. Ramadhena, *Weight Changes for Learning Mechanisms in Two-Term Back-Propagation Network*. INTECH Open Access Publisher, 2013.

[37] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[38] R. G. P. María Esther Serafín de Fleischmann, *Diccionario de Lengua de Señas Mexicana*. Consejo Nacional para Prevenir la Discriminación, 2011.

[39] D. Kriesel, *A Brief Introduction to Neural Networks*. 2007.

[40] G. van Rossum, "Python tutorial," Report CS-R9526, Apr. 1995.

[41] J. Bauer, "An introduction to Python," *j-LINUX-J*, vol. 21, jan 1996.