UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO
POSGRADO EN CIENCIA E INGENIERIA DE LA COMPUTACION

CONSTRUCCION DE UN SISTEMA DE INTELIGENCIA ARTIFICIAL PARA LA INTERPRETACION DEL LENGUAJE DE SEÑAS EN TIEMPO REAL

QUE PARA OPTAR POR EL GRADO DE MAESTRO EN CIENCIA E INGENIERIA

DE LA COMPUTACION

PRESENTA: ALEJANDRO SALINAS MEDINA

TUTOR:

M. EN C. GUSTAVO ARTURO MARQUEZ FLORES
POSGRADO EN CIENCIA E INGENIERIA DE LA COMPUTACION

[CIUDAD DE MEXICO, MAYO, 2021]

Agradecimientos

Para mis padres Alejandro Salinas Caso y María Guadalupe Medina Huerta;
quienes son los pilares de toda mi vida y estoy eternamente agradecido
por brindarme todas las herramientas y amor que he necesitado para
convertirme en la persona que soy.

Para CONACyT, por brindarme el apoyo económico durante mi
programa de Maestría; en el cual desarrolle este proyecto.

Índice general

Resumen

Este proyecto tiene la finalidad de resolver una problemática ocasionada por la discapacidad auditiva que puede llegar a presentarse en los humanos. El objetivo de este sistema es crear una herramienta que logre facilitar la comunicación para personas sordas mexicanas. Para esto, se crearán 4 distintos tipos de arquitectura de modelos de Inteligencia Artificial y se comparara cual de estos logra de una manera mas eficiente interpretar la lengua de señas mexicana.

Actualmente no existen muchos conjuntos de datos de la lengua de seña mexicana, por lo que otro objetivo de este proyecto es la creación y publicación de uno de los primeros conjuntos de datos de la lengua de seña mexicana suficientemente robusto para el entrenamiento de redes neuronales profundas.

Finalmente, se elegirá el mejor modelo y se desplegara a través de una aplicación amigable para el usuario que sea capaz de mejorar la comunicación entre personas sordas y personas que no estén familiarizadas con el lenguaje de señas mexicano.

Abstract

Today there are not many technologies that are capable of extracting data from images. Artificial Neural Networks have had a boom in recent times, which is why different types of them have been developed. The architecture of Neural Networks Convolutions is specialized to extract characteristics in images. This work presents a novel approach to use architectures based on convolutional neural networks and knowledge transfer for the interpretation of a newly generated "Mexican Sign Language" data set.

Capítulo 1. Introducción

"La inteligencia artificial es la nueva electricidad"

- Andrew Ng

Las aplicaciones de aprendizaje profundo están presentes en diferentes aspectos de nuestra vida diaria, como los motores de búsqueda web, las recomendaciones de redes sociales, el reconocimiento del lenguaje natural y las sugerencias de comercio electrónico [21]. El surgimiento de las redes neuronales convolucionales ha revolucionado la manera de resolver tareas específicas en el campo de la visión computacional como son:

- Clasificación de imágenes
- Reconocimiento y localización de objetos
- Clasificación y reconocimiento de acciones
- Segmentación de imágenes
- Generación de imágenes

Las redes neuronales convolucionales se han convertido en un parteaguas para la visión computacional. El estado del arte para la clasificación de imágenes incluye diversas arquitecturas de redes neuronales convolucionales que combinan distintos tipos de bloques de operaciones para lograr distintos resultados.

1.1 Motivación

En 2018 se estimó que 466 millones de personas en todo el mundo tenían una pérdida auditiva discapacitante[1]. En la actualidad, uno de los grandes retos que afrontan las personas sordomudas es lograr una comunicación eficiente con el resto del mundo, sin embargo pueden llegar a existir distintas complicaciones:

- Las personas no conocen el abecedario de lenguaje de señas
- Distintos lenguajes
- Mal uso del lenguaje de señas
- Problemas con la vista
- Discriminación

Para atacar directamente a todas estas complicaciones, se propone un sistema que cuente con la capacidad de traducir todas las señas del lenguaje de señas a valores alfanuméricos. El problema es que, al igual que con los idiomas hablados, todos los lenguajes de signos son diferentes. No existe una lengua de signos global compartida en todo el mundo [2]. Por lo tanto, una solución de traducción genérica no es suficiente para abordar este problema para todas las personas sordas del mundo.

En la actualidad no existe ninguna aplicación que ayude a la personas sordas en la interpretación el lenguaje de señas genérico en texto. Si hubiera solo uno, por ejemplo, una herramienta que interprete solo el lenguaje de señas americano no funcionaría en el lenguaje de señas mexicano. Debido a esto, es necesario la implentación de un sistema con la capacidad de interpretar el lenguaje de señas mexicano.

Existen diversos sistemas gratuitos que nos pueden ayudar a traducir del lenguaje alfanumérico a lenguaje de señas. Sin embargo para hacer la traducción inversa en tiempo real existe una enorme escasez en los sistemas gratuitos.

Gracias a los más recientes avances en el campo de la Inteligencia Artificial y del Desarrollo Web es posible pensar en diseñar y desarrollar un sistema de interpretación en línea del lenguaje de señas mexicano. Además, que ese sistema será de uso abierto y gratuito gracias a los nuevas herramientas tecnológicas que ofrece el Desarrollo Web.

1.2 Problema u Oportunidad

Cuando una persona de una familia se vuelve sorda o nace con problemas de audición, pueden surgir varios problemas[40]. En particular, las personas sordas que a menudo utilizan el lenguaje de señas para comunicarse, en muchos casos dependen de intérpretes cuando buscan comunicarse. Las personas que necesitan utilizar el lenguaje de señas a menudo no pueden comunicarse eficazmente con personas que no están familiarizadas con el lenguaje de señas [41]. En este contexto, una aplicación que traduzca

automáticamente el lenguaje de señas es beneficiosa ya que puede mejorar la calidad de vida de las personas sordas, especialmente en términos de mayor inclusión social y libertad individual. Aunque existen varios traductores de lenguaje alfanumérico a lenguaje de señas, muy pocos son los que traducen del lenguaje alfanumérico a lenguaje de señas y no vicecersa. También es importante mencionar que los sistemas no ofrecen una traducción en tiempo real y cuando se hablan de sistemas gratuitos, las opciones disminuyen. Para el caso contario, traductor de lenguaje de señas a lenguaje alfanumérico, las opciones se reducen y al tratarse de sistemas gratuitos de traducción de lenguaje de señas en tiempo real, las opciones son casi nulas.

1.3 Objetivo

1.3.1 Objetivo principal

- Llevar a cabo la construcción un algoritmo capaz de reconocer todas las señales del lenguaje de señas mexicano con alta precisión.

1.3.2 Objetivos específicos

- Creación y comparación de distintos modelos de Redes Neuronales.
- Desplegar y publicar este algoritmo en una aplicación de Internet (Web App).
- Adaptación de la interfaz de usuario a diseño responsivo.

1.4 Pregunta de Investigación o Hipótesis

Hoy en día los algoritmos de inteligencia artificial tienen una precisión mayor al 60% cuando se trata de generalizar predicciones con datos no incluidos en el conjunto de entrenamiento y validación de los respectivos modelos de inteligencia artificial. Las redes neuronales convolucionales cuentan con la robustez necesaria para poder crear un clasificador con el conjunto de datos generados el cual superara el 80% de precisión cuando se trate de generalizar con datos no incluidos en el conjunto de entrenamiento y validación de los respectivos modelos de inteligencia artificial.

1.5 Contribución y relevancia

Actualmente no hay abundantes bases de datos de lenguaje de señas mexicano, por lo que se ha trabajado en la construcción de una de las primeras bases de datos en lenguaje de señas mexicano que son lo suficientemente robustas para entrenar modelos de aprendizaje profundo. Se ha comparado distintas arquitecturas de redes neuronales convolucionales, finalmente se aplicó la técnica de aprendizaje por transferencia de una red previamente entrenada con la base de datos ImageNet: congelando sus primeras capas para mantener el aprendizaje que esta red previamente entrenada incluye y agregando un par de capas convolucionales al final para aprovechar la base de datos del lenguaje de señas mexicano. Este modelo de aprendizaje profundo se ha implementado con una interfaz de usuario fácil de usar a través de una aplicación web construida con un framework de Javascript (AngularJS) que permite a los usuarios comunicarse con el lenguaje escrito a través de lenguaje de señas con personas con discapacidad en tiempo real.

1.6 Trabajos Relacionados

- Mocialov [35] construyó un sistema que utiliza el transferencia por conocimiento para interpretar el lenguaje de señas británico.
- Koller [32] y Weissman [30] se han centrado en el reconocimiento de imágenes mediante el uso de Redes Neuronales Convolucionales.
- Mocialov [33] realizó un estudio que tiene como objetivo reconocer un flujo de señales continuas en un video.
- Halvardsson[42] creo un sistema que utiliza transferencia de aprendizaje para realizar una interpretación del lenguaje de señas sueco. Se realizó una comparación de distintos optimizadores en el entrenamiento de redes neuronales.

1.7 Metodología

Para llevar a cabo la construcción del sistema y asegurar el mejor desempeño de este se siguió la siguiente metodología:

1. Construcción y comparación de distintas arquitecturas de redes neuronales convolucionales para la interpretación de lenguaje de señas.
2. Construcción de una aplicación web.
3. Unificación de front end y back end a través de Restful APIs.
4. Adaptación de la aplicación web responsiva

1.8 Organización de la Tesis

El presente trabajo se estructura en seis capítulos.

En el presente capítulo, se da una introducción a los conceptos que forman parte del marco teórico, y que son necesarios para entender el contexto en el que se desenvuelve el trabajo.

En el capítulo Estado del Arte, se da un breve resumen sobre los textos que contienen información pertinente del estado del arte del tema.

El capítulo Propuesta y Diseño describe puntualmente la propuesta de solución, las tecnologías utilizadas y se da una breve introducción a todos los sistemas internos desarrollados.

En el siguiente capitulo, se describe puntualmente la implementación de distintas herramientas para crear uno de los primeros conjuntos de datos del sistema de señas mexicano.

Una vez presentado el diseño, en el capítulo Programación y pruebas se anexan algunas métricas de rendimiento que se le pueden aplicar al framework cuando en un futuro sea implementado.

Finalmente, en el capítulo Conclusiones y Trabajo Futuro se recapitulan los alcances del trabajo y se mencionan los puntos posibles a desarrollar para un trabajo futuro.

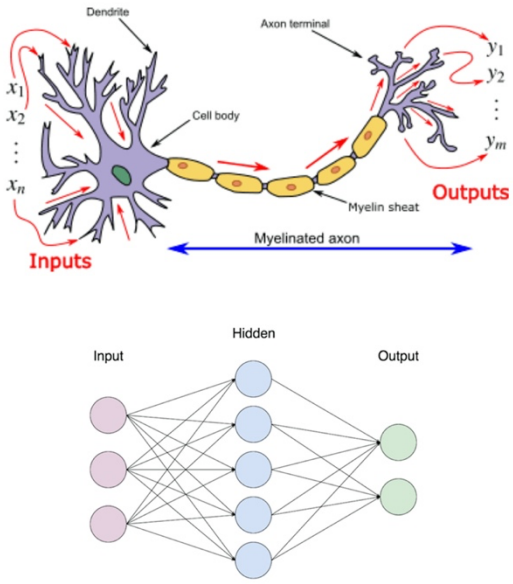


Figura 1. Comparación arquitectura red neuronal artificial y una neurona.

Capítulo 2. Estado del Arte

"Es difícil pensar en una industria importante que la IA no transformará. Esto incluye la salud, la educación, el transporte, el comercio, las comunicaciones y la agricultura. Hay caminos sorprendentemente claros para que la IA marque una gran diferencia en todas estas industrias".

- Andrew Ng

Se han realizado varios estudios destinados a interpretar lenguajes de señas. Algunos de ellos se han basado en guantes que pueden interpretar signos, mientras que otros se basan en la visión por computadora y comparaciones estadísticas.

2.1 Interpretación del lenguaje de señas mediante supervisión humana

Hernandez [26] utiliza un guante "AcceleGlove" para la interpretación del lenguaje de señas. En este trabajo, los diferentes ángulos por dedo se utilizan como entrada a un programa de computadora que analiza las posiciones. Se prueba en diferentes sujetos y es capaz de reconocer 30 signos con una mano con una precisión del 98%. Glenn [27] se basa en visión por computadora y comparaciones estadísticas. Cada seña se filma en un entorno controlado, se extrae el fondo, se recorta la imagen, se cambia el tamaño y se detectan los bordes, y finalmente se coloca en una base de datos estadística adaptativa. Para clasificar una seña como una palabra en particular, la imagen se procesa y luego se compara con todas las imágenes de la base de datos estadística. Akram [28] utilizó un sensor Kinect para reconocer las señas. Akram utiliza un sensor Kinect RGB-D colocado en la mano. Esto permite eliminar los fondos, ayuda con la resolución cuando se coloca la mano frente a la cara y simplifica el uso de señas 3D. La clasificación se realiza a través de una base de datos estadística. Los resultados son diferentes dependiendo de quién realice las señas. Un firmante recibió una precisión del 77% mientras que otro tuvo el 94%.

Figura 2. Guante Acceleglove

2.2 Interpretación de lenguaje de señas usando Redes Neuronales Convolucionales (CNN)

Los beneficios del uso de redes neuronales se basan en su capacidad para derivar el significado de patrones demasiado complejos para ser notados por humanos o algoritmos tradicionales [29]. Weissman [30] se ha centrado en el reconocimiento de gestos. Los experimentos se realizan con un CyberGlove, un guante con sensores de realidad virtual. El análisis se realiza mediante una Red Neuronal Artificial multicapa y la precisión es cercana al 100% para algunos gestos. Pugeault [31] presenta una interfaz gráfica de usuario interactiva, que muestra un buen rendimiento y solidez para múltiples usuarios. Koller [32] aprovecho la capacidad discriminativa de las Redes Neuronales Convolucionales con su aplicación para la clasificación de la forma de la mano en el ámbito del lenguaje de señas. Mocialov [33] realizó un estudio que tiene como objetivo reconocer un flujo de señales continuas en un video. La arquitectura de red neuronal utilizada es una red neuronal recurrente. La red tiene conexiones de retroalimentación que son adecuadas para el procesamiento de video. El artículo reporta una precisión del 80% en un flujo continuo de datos de video. Quirk [34] traduce los signos filmados con una cámara web. El estudio tiene como objetivo traducir el lenguaje de señas Auslan. El conjunto de datos para el alfabeto Auslan se genera extrayendo signos de videos de YouTube y dibujando cuadros sobre las manos. Usan redes neuronales convolucionales y la precisión final es del 86%.

2.3 Interpretación de lenguaje de señas mediante Transferencia de Conocimiento

Mocialov [35] construyó un sistema para interpretar el lenguaje de señas británico. Los conjuntos de datos del lenguaje de señas británico utilizaron una área para el inglés estándar con transcripciones para el lenguaje de señas y una área de preprocesado llamado Penn Treebank. Los videos de las señas están divididos por frases. Las arquitecturas de aprendizaje de máquina utilizadas se basan tanto en redes neuronales recurrentes como en redes neuronales convolucionales. Su trabajo muestra buenos resultados en palabras, pero las oraciones no se interpretan correctamente desde el punto de vista gramatical. Este estudio también puede utilizar el hecho de que existía un área del lenguaje británico con más

ejemplos por seña (algo que no está disponible para lenguaje de señas mexicano). Esto elimina la necesidad de crear nuevos datos para el proceso de aprendizaje por transferencia.

Capítulo 3. Propuesta y Diseño

"El aprendizaje automático nos permite crear soluciones de software que superan la comprensión humana y nos muestra cómo la IA puede inervar todas las industrias.
-Steve Jurvetson

CAPITULO 3. PROPUESTA Y DISEÑO

La propuesta inicial para el desarrollo de la arquitectura de este sistema se divide en dos principales ramas:

- Construcción y comparación de distintas arquitecturas de redes neuronales convolucionales para la interpretación de lenguaje de señas.
- Construcción de una aplicación web.

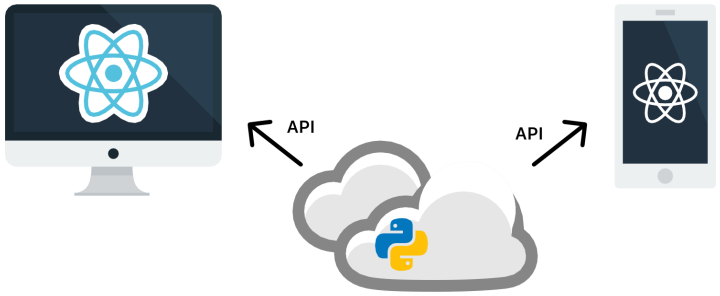


Figura 3. Diagrama de la arquitectura del proyecto. La nube representa la construcción de algoritmos de Inteligencia Artificial, mientras que cada flecha represente la API Rest que comunicara con el frontend en distintos dispositivos.

3.1 Tecnología Empleada

3.1.1 Protcolo HTTP

El Protocolo de transferencia de hipertexto (HTTP) es un nivel de aplicación de protocolo con la ligereza y rapidez necesarias para distribuir sistemas colaborativos de información hipermedia. Es un protocolo genérico orientado a objetos sin estado que se puede utilizar para muchas tareas, como creación de servidores web y sistemas de gestión de objetos distribuidos,mediante la extensión de sus métodos de solicitud. Una característica de HTTP es la tipificación de la representación de datos, lo que permite que los sistemas sean construidos independientemente de los datos que se transfieren[6]. Gracias a tipificación de la representación de datos, este protocolo

permitira crear un canal de comunicación entre el servidor y el cliente en el que se podra mandar las imágenes de los usuarios de la aplicación

3.1.2 Marco Model-View-Controller (MVC)

Actualmente, el desarrollo de sitios web ha avanzado mucho desde el comienzo de la World Wide Web. Una de las tecnologías para crear aplicaciones web es MVC. Este es un método para construir web mediante la separación de la capa del modelo, la capa del controlador y la capa de visualización para que sea más fácil y rápido la implementación de sitios y aplicaciones web. MVC ha demostrado sus beneficios para las aplicaciones web interactivas que permiten múltiples representaciones de la misma información, promueven la reutilización del código y ayudan a los desarrolladores a enfocarse en ciertas características de la aplicación. El marco MVC se ha convertido en general en un estándar en el desarrollo de software moderno[7].

3.1.3 Modularización en el diseño de software

El módulo es la unidad básica de desarrollo, mantenimiento y gestión de software. Una actividad básica del proceso de diseño de software es la división de la especificación del software en un cierto grupo de módulos de software que, en conjunto, satisfacen el planteamiento del problema original [8]. Los principales criterios teóricos para la modularización del software incluyen fuerza / cohesión y acoplamiento y ocultación de información [9].

Sin embargo, estos criterios son difíciles de cuantificar. Un observador independiente del proceso de desarrollo no puede determinar fácilmente los niveles de fuerza, acoplamiento y ocultación de información logrados en un módulo determinado. Por tanto, el uso de estos conceptos está limitado en un entorno en el que se hace hincapié en el aseguramiento de la calidad.

En consecuencia, se han adoptado medidas de tamaño (número de líneas de código fuente o declaraciones ejecutables) como un simple recurso [10]. Aunque se han afirmado muchos beneficios por las limitaciones del tamaño de los módulos, en la actualidad no existe una base teórica o evidencia empírica para usar el tamaño del módulo como criterio para la modularización del software [11].

3.1.4 Jupyter Notebook

Jupyter Notebooks es una herramienta de código abierto y basada en navegador que integra lenguajes interpretados, bibliotecas y herramientas de visualización [19]. Jupyter Notebooks puede funcionar de forma local o en la nube. Cada documento está compuesto por múltiples celdas, donde cada celda contiene lenguaje de escritura y la salida está incrustada en el documento. Las salidas típicas incluyen texto, tablas, cuadros y gráficos. El uso de esta tecnología facilita el intercambio y la reproducción de trabajos científicos, ya que los experimentos y los resultados se presentan de forma autónoma [20].

3.1.5 Google Colaboratory

Google Colaboratory (también conocido como Colab) es un servicio en la nube basado en Jupyter Notebooks para difundir la educación y la investigación del aprendizaje automático. Proporciona un tiempo de ejecución completamente configurado para un aprendizaje profundo y acceso gratuito a una GPU robusta.

Los recursos de hardware presentan riesgos [24]: subutilización y sobreutilización, depreciación del hardware y fallas. También hay costos relacionados con el mantenimiento, la energía y los recursos humanos. En la realidad de un grupo de investigación, puede ser difícil mantener una computadora robusta con varias GPU para las pruebas. En contraste, las unidades

de procesamiento de gráficos (GPU) son dispositivos masivamente paralelos candidatos para realizar una tarea en paralelo. Este tipo de acelerador es ubicuo, accesible y ofrece una alta tasa de GFlops / dólar [22]. Además, los principales marcos de aprendizaje profundo están programados para las GPU NVIDIA [23].

Colaboratory proporciona ambientes de ejecución de Python 2 y 3 preconfigurados con las bibliotecas esenciales de aprendizaje automático e inteligencia artificial, como TensorFlow, Matplotlib, Pytorch y Keras. Es posible transferir archivos desde el disco duro de la maquina virtual de Colab (VM) al equipo local del usuario. Finalmente, este servicio de Google proporciona un tiempo de ejecución acelerado por GPU. La infraestructura de Google Colaboratory está alojada en la plataforma Google Cloud [25].

3.1.6 Tensorboard

TensorBoard es el kit de herramientas de visualización de TensorFlow, permite realizar un seguimiento de métricas como la pérdida y la precisión, visualizar el gráfico del modelo, ver histogramas de pesos, sesgos u otros tensores a medida que cambian con el tiempo y mucho más. Es una herramienta de código abierto que forma parte del ecosistema de TensorFlow [38]. Debido a que la mayoría de los marcos de uso de aprendizaje profundo trabaja con tensores, Pytorch incluido, Tensorboard se vuelve una herramienta de gran utilidad para el trabajo relacionado con el aprendizaje profundo.

3.1.7 Aplicación de una sola pagina (SPA)

Los sitios web tradicionales que se introdujeron en los primeros días de la World Wide Web tenían el único propósito de ofrecer páginas estáticas (contenido) a su cliente, estos incluían contenido como imágenes, CSS, JavaScript, etc. Mientras la cantidad de data producida y contenida por la sociedad creció

exponencialmente, la gente comenzó a usar sitios web para publicar sus negocios. Con la introducción del comercio electrónico, la necesidad de proporcionar el estado actual de los negocios aumentó y ese requisito llevó a servir las páginas dinámicas que eran la imagen especular del estado de los negocios en vivo.

En la era moderna del desarrollo web, la mayoría de los sitios web utilizan SPA, esto es una aplicación web que se encuentra en una sola página como cualquier otra aplicación de escritorio. En SPA, todos los componentes como CSS, imágenes, scripts y cualquier otro recurso requerido se cargan al mismo tiempo en la carga inicial de la página y luego el contenido / componentes apropiados se cargan dinámicamente dependiendo de la interacción del usuario. La SPA se integra por componentes individuales que se pueden reemplazar / actualizar de forma independiente, sin actualizar / volver a cargar la página completa, de modo que no es necesario volver a cargar la página completa en cada acción del usuario. Toda la página está dividida en subcomponentes más pequeños que interactúan entre sí. La SPA es responsable de manejar todas las interacciones del usuario, como hacer clic en un botón, ingresar desde el teclado, etc., por lo tanto, conduce a una interfaz de usuario muy fluida. SPA permite una forma más flexible y eficiente de tratar los datos. Actualizar una parte particular o una sección de una página sin actualizar una página completa es el objetivo principal que ofrece SPA, pero toda esta flexibilidad requiere una interfaz más interactiva y esto conduce a una mejor experiencia del usuario [39].

3.1.8 OpenCV

Open CV es una biblioteca de visión por computadora multiplataforma con un código abierto que comenzó como un proyecto de investigación en Intel en 1998 y ha estado disponible desde el año 2000. Open CV tiene como objetivo proporcionar las herramientas necesarias para resolver problemas de visión por computadora, se puede preparar por lenguaje C y C ++, contiene una combinación de funciones de procesamiento de imágenes de bajo nivel y algoritmos de alto nivel [44-45].

3.2 Marcos de Trabajo

3.2.1 Flask

Contemporáneamente, los recursos disponibles en repositorios en la web han experimentado un crecimiento exponencial siendo la World Wide Web la principal herramienta para acceder a estos. La arquitectura web se compone principalmente de dos entidades: cliente y servidor [5]. El cliente web es una aplicación en la máquina host que solicita recursos, y el servidor web es una máquina en la web que es responsable de cumplir con la solicitud emitida por el cliente. El Protocolo de transferencia de hipertexto (HTTP) es el protocolo más popular utilizado por el cliente y el servidor para la comunicación web. En una web estática, el navegador emite una solicitud HTTP al servidor HTTP, que busca el recurso requerido en su base de datos y lo devuelve como una respuesta HTTP. Para evitar problemas de compatibilidad, cada solicitud emitida por el navegador tiene la forma de una URL (Localizador uniforme de recursos). El protocolo URL define las reglas para la comunicación entre el cliente y el servidor.

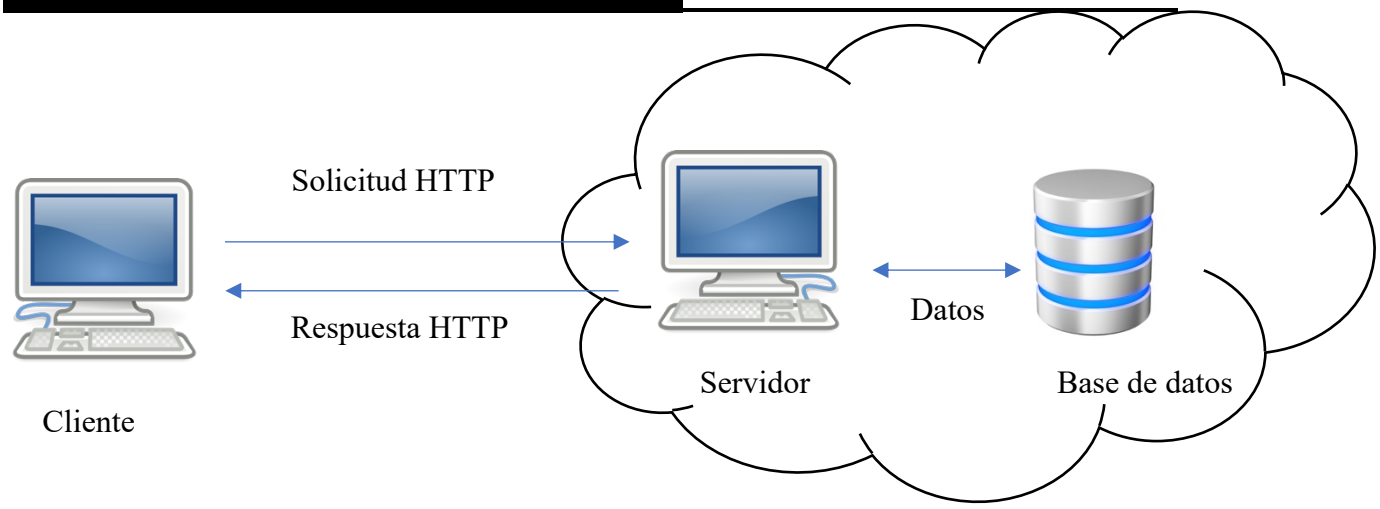


Figura 3.1. Arquitectura de aplicaciones web

El marco Flask es bien intencionado [3] por lo que su uso resultará en una arquitectura flexible. Flask se basa en Werkzeug y Jinja 2, no tiene dependencias aparte de la biblioteca estándar de Python. Además, Flask no incluye componentes que necesiten soporte de terceros. Los servicios ofrecidos por este marco incluyen servidor HTTP integrado, soporte para pruebas unitarias y servicio web RESTful [4]. Gracias a la combinación de la tipificación de la representación de datos y de la generalización ofrecida por el protocolo HTTP, en combinación con la falta de dependencias que ofrece Flask, una arquitectura sumamente flexible sera posible de ser construida para la comunicación entre el cliente, el cual sera encargado de tomar las fotos para interpretar, y el servidor, el cual sera el encargado de dar una interpretación a la foto.

El marco Model-View-Controller (MVC) se ha convertido en el estándar en el desarrollo de software moderno, con la capa del modelo, la capa de visualización y la capa del controlador haciéndolo más fácil y rápido. El marco de Flask no cuenta con una implementación para utilizar el método MVC, por lo que los archivos y códigos no son regulares. Se diseñó una arquitectura simple MVC dentro del marco Flask, en

la cual, la modularización de sus funciones es vital para lograr este tipo de arquitectura.

La modularización de las funciones escritas dentro de Flask, se dividieron de tal manera en la cual se puede dividir el proyecto siguiendo el marco MVC. Debido a los módulos creados, el proyecto se divide en las siguientes secciones:

- Controladores

  Dentro de esta rama del proyecto se crearon las funciones para transformar la imagen recibida a un tensor (transform.py) y hacer la predicción (prediction.py). En estos archivos se crearon funciones con el framework Pytorch, del cual se hablara en la siguiente función.

- Endpoints

  En esta parte del proyecto, se ha aprovechado los blueprints de Flask para establecer una ruta con un operador del protocolo HTTP para cada función modular de los controladores. Un Blueprint es una forma de organizar un grupo de vistas relacionadas y otro código.

- Modelos

  En esta sección se tiene un repositorio con los archivos .pth exportados del entrenamiento de la arquitectura final de la red neuronal.

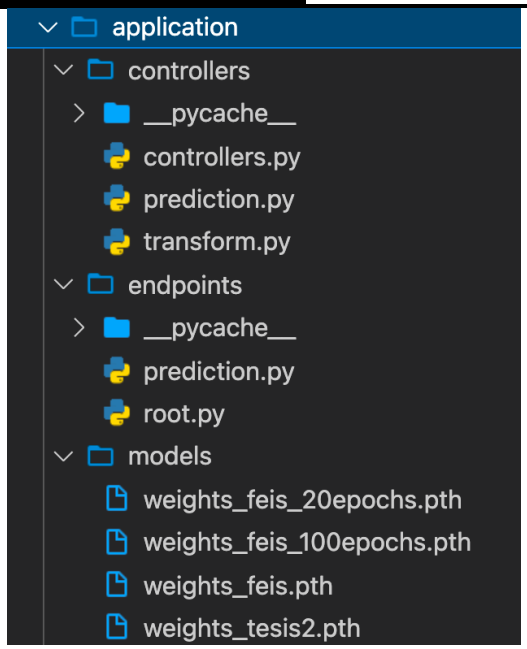


Figura 3.2. Arquitectura basada en MCV.

3.2.2 Pytorch

Debido al creciente interés en el aprendizaje profundo en los últimos años, ha existido una abundancia en la creación de herramientas de aprendizaje automático. Muchos marcos populares como Caffe [12], CNTK [13], TensorFlow [14] y Theano [15]. Estos construyen un gráfico de flujo de datos estático que representa el cálculo, que luego se puede aplicar repetidamente a lotes de datos. Este enfoque proporciona visibilidad de todo el cálculo por adelantado y, en teoría, se puede aprovechar para mejorar el rendimiento y la escalabilidad. Sin embargo, se produce a costa de la facilidad de uso, la facilidad de depuración y la flexibilidad de los tipos de cálculo que se pueden representar. El trabajo previo ha reconocido el valor de la ejecución dinámica y ávida para el aprendizaje profundo, y algunos marcos recientes implementan esta definición por ejecución enfoque, pero hecho a costa del rendimiento (Chainer [16]) o mediante el uso de un lenguaje menos expresivo y más rápido (Torch [17], DyNet [18]), lo que limita su aplicabilidad.

La ejecución dinámica y ávida se puede lograr en gran medida sin sacrificar el rendimiento. PyTorch es una biblioteca de Python que realiza la ejecución inmediata de cálculos de tensores dinámicos con diferenciación automática y aceleración de GPU. Lo hace mientras mantiene un rendimiento comparable al de las bibliotecas actuales más rápidas para el aprendizaje profundo.

Se utilizo la plataforma Google Colaboratory, en conjunto de las capacidades de realizar ejecuciones inmediatas de cálculos de tensores dinámicos con diferenciación automática y aceleración de GPU de Pytorch para realizar el entrenamiento de los modelos de aprendizaje profundo.

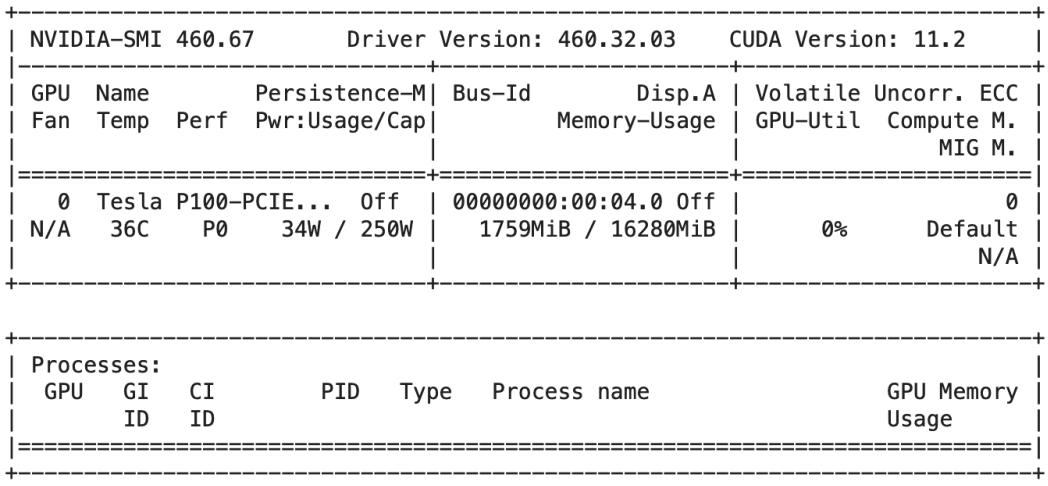


Figura 3.3. Datos GPU Colaboratory

Se utilizo un ambiente de ejecución de Python 3 dentro del Jupyter Notebook brindado por Google Colaboratory. Dentro de este, se definieron las siguientes funciones modulares:

- **Preparación**
  1. **Bibliotecas**

     En esta sección se definen todas las librerías requeridas para hacer entrenar los diferentes modelos.

2. **Utilerías**

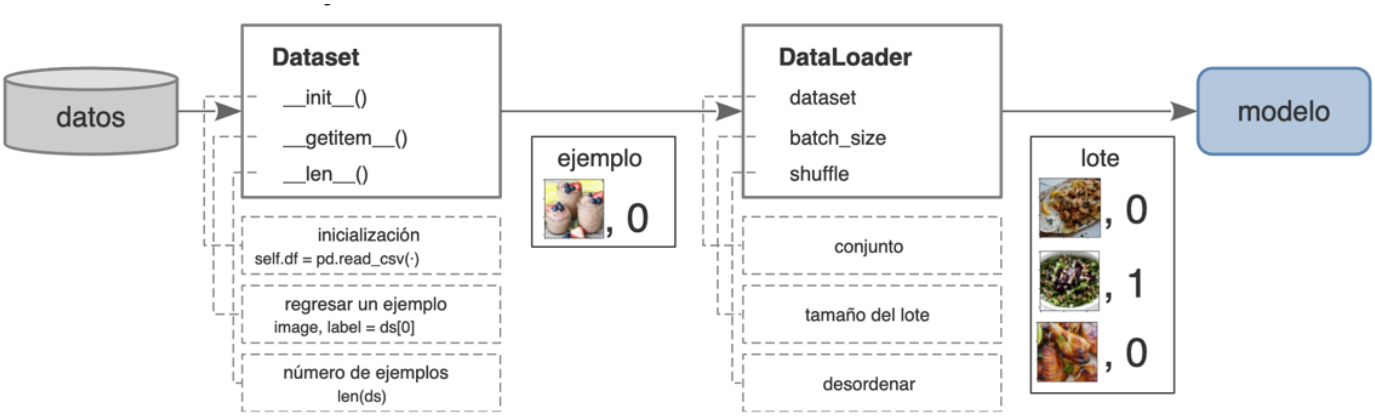
   Se generan tres funciones modulares para realizar las siguientes tareas: Desplegar ejemplos en una cuadrícula, desplegar un lote en una cuadrícula, regresar la marca de tiempo.

   Las primeras dos tareas son realizadas para mostrar resultados de una manera limpia mientras que la última se utiliza para la exportación de modelos.

- **Datos**

1. **Tuberías de datos con PyTorch**

   El código para procesar muestras de datos puede resultar complicado y difícil de mantener. Idealmente, los desarrolladores de Pytorch han logrado que su código de conjunto de datos se desacople de su código de entrenamiento para los modelos y lograr una mejor legibilidad y modularidad. PyTorch proporciona dos primitivas de datos: DataLoader y Dataset [36] que permiten utilizar conjuntos de datos precargados, así como datos generados



por el usuario.

Figura 3.4. Diagrama de flujo de las primitivas de datos: DataLoader y Dataset.

El conjunto de datos almacena las muestras y sus etiquetas correspondientes, y DataLoader envuelve un iterable alrededor del conjunto de datos para permitir un fácil acceso a las muestras.

2. Conjunto de datos

En esta parte se genera una función que utiliza la primitiva de datos Dataset para importar el conjunto de datos de lenguaje de señas mexicano al ambiente de desarrollo de Python 3.

3. Transformaciones

Torchvision tiene un conjunto de transformaciones [37] para ser ejecutadas de forma secuencial. El objetivo de ésta es transformar cada una de las imágenes a tensores y normalizar el conjunto de datos con la media y varianza del conjunto de datos IMAGENET (para el caso de la arquitectura en la que se hace aprendizaje por transferencia) y con sus propios valores (para el caso del resto de las arquitecturas). Dentro de esta función se define un numero de mini lote, en el cual el conjunto de datos de entrenamiento y validación quedaran divididos en pequeños grupos de tensores. Se realizaron pruebas con tamaños de 32 ,64 y 128. Se encontraron mejores resultados con grupos de 64.

Después de aplicar las transformaciones al conjunto de datos, las imágenes son transformadas a un tensor con la siguiente forma: (64, 3, 224, 224) para los modelos sin transferencia de aprendizaje y (64, 3, 299, 299) para los modelos con transferencia de aprendizaje. Las etiquetas a un vector con la siguiente forma: (64).

4. Cargadores de datos

Finalmente, se utiliza la primitiva de datos DataLoader para cargar los datos transformados y separados en 3 distintos grupos: datos de entrenamiento, datos de validación y datos de prueba. Los cargadores de datos pueden precargar el lote de entrenamiento en paralelo, esto depende del parámetro "*num_workers*".

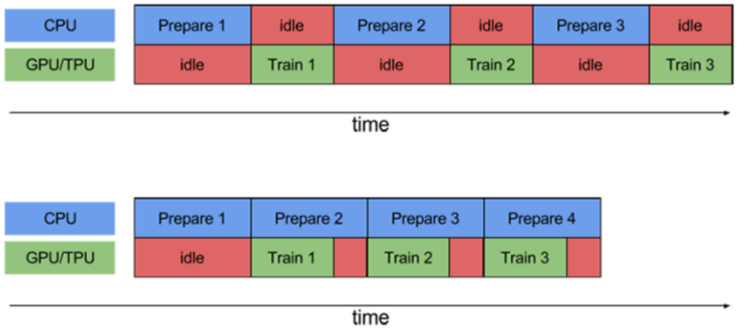


Figura 3,5. Diagrama precarga del lote de entrenamiento forma paralela

Figura 3.6. Cuadrícula con las imágenes del conjunto de datos de Entrenamiento.

- **Modelo**
    1. **Carga de pesos**

       Esta sección se desarrollo exclusivamente para los modelos que usan aprendizaje por transferencia. Se define una función para importar la arquitectura de la cual se vaya a aprovechar el conocimiento generado en ésta.
    2. **Transferencia de Conocimiento**

       Esta sección se desarrollo exclusivamente para los modelos que usan aprendizaje por transferencia. Se define una función para congelar los parámetros de las capas iniciales del modelo y agregar una nueva capa de clasificación con las especificaciones del conjunto de datos.

    3. **Definición Modelo**

       Este apartado fue creado para los modelos que no usan transferencia de conocimiento. En éste, se utilizan los métodos aportados por Pytorch para la generación de distintas arquitecturas de redes neuronales.

- **Entrenamiento**
    1. **Funciones de Entrenamiento**

       En esta sección se definen funciones para: Entrenar una época, Evaluar una época, Guardar un punto de control y la función "Entrenamiento", en la cual se invocan el resto de las funciones definidas.

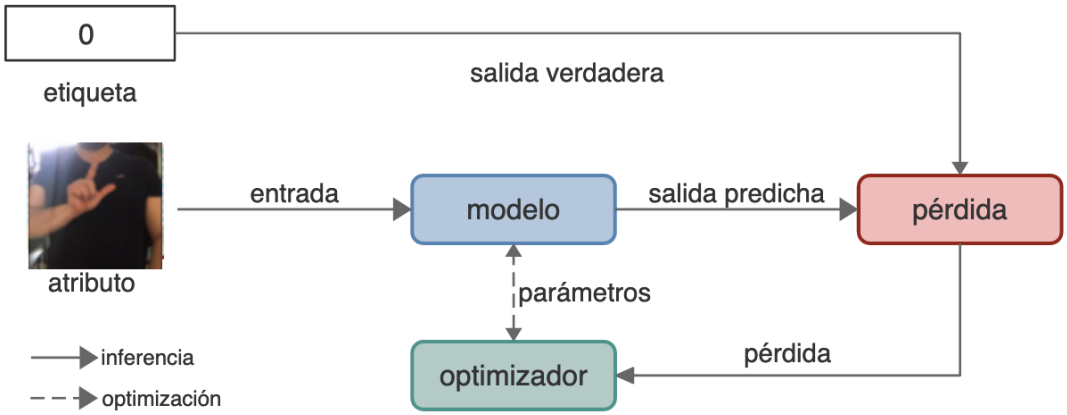


Figura 3.7. Diagrama Entrenamiento Modelo

2. **Monitoreo**

   Para realizar un monitoreo del comportamiento de los modelos bajo entrenamiento, se utiliza la librería Tensorboard, que permite graficar en tiempo real las métricas generadas por el entrenamiento de los modelos.

- **Evaluación**
  1. **Carga Modelo**

     En esta sección, se carga nuevamente la arquitectura de red neuronal que se este utilizando.

  2. **Conjunto y cargador de prueba**

     En este apartado, se utiliza la primitiva de datos para cargar, transformar, crear mini lotes y ordenar de una manera aleatoria el conjunto de datos para pruebas.

  3. **Evaluación Final**

     En esta sección se ha creado una función para imprimir las métricas finales de los distintos modelos.

4. **Inspección visual de resultados**

   En esta sección se ha creado una función para imprimir una cuadrícula con las imágenes del conjunto de datos de pruebas, mostrando la clase real y la predicha.

Figura 3.8. Cuadricula con imagen y valores predichos y verdaderos.

3.2.3 Angular JS

AngularJS es un marco de JavaScript de código abierto mantenido por Google y la comunidad que puede ayudar a los desarrolladores a crear aplicaciones de una sola página (SPA). Su propósito es ayudar a desarrollar aplicaciones web con capacidad modelo-vista-controlador (MVC). AngularJS ayuda a crear aplicaciones web basadas en HTML, CSS, JavaScript. AngularJS trae la capacidad MVC a la aplicación web y, por lo tanto, la hace más modular y fácil de desarrollar, mantener y probar. Los controladores AngularJS están escritos en Typescript (JavaScript), lo que agrega la lógica empresarial a las vistas que no son más que páginas HTML. Las SPA están completamente cargadas en la carga inicial y solo las regiones / secciones de la página se reemplazan o actualizan

con nuevos fragmentos de página cargados desde el servidor a pedido. AngularJS es una biblioteca totalmente orientada al lado del cliente. Esta se basa en un enlace de datos bidireccional completo, esta es una forma automática de actualizar una vista cuando cambia el modelo y actualizar un modelo cuando cambia la vista [39].

Aprovechando las características de carga de SPA, se han creado dos distintos componentes:

- Pantalla Inicial

  En esta se muestran los datos del autor y contiene una dedicatoria.

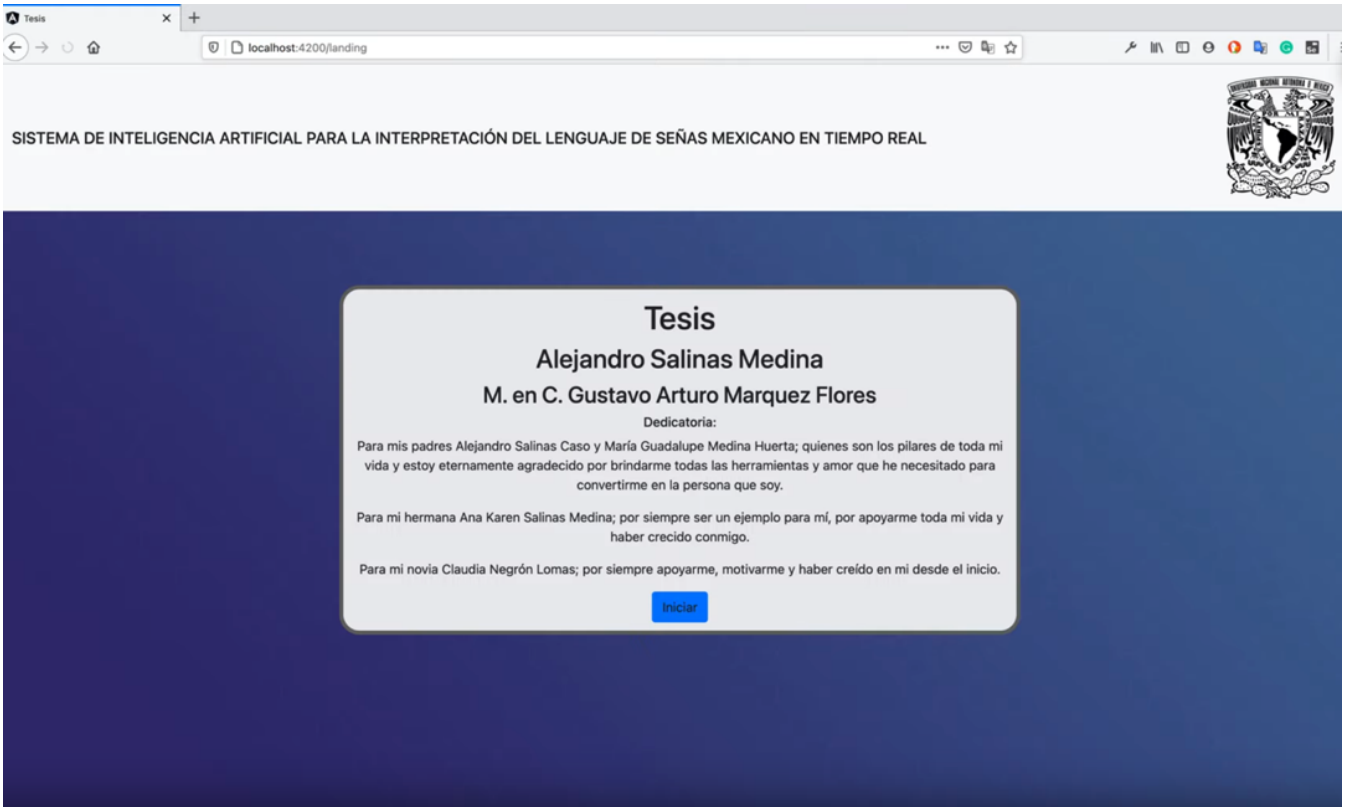


Figura 3.9. Pantalla Inicial

- Pantalla Principal

  En este componente se han creado distintas funciones para conectar el proyecto de frontend al de backend para mandar una foto capturada en vivo y se pueda hacer su interpretación en el proyecto de Flask [3].

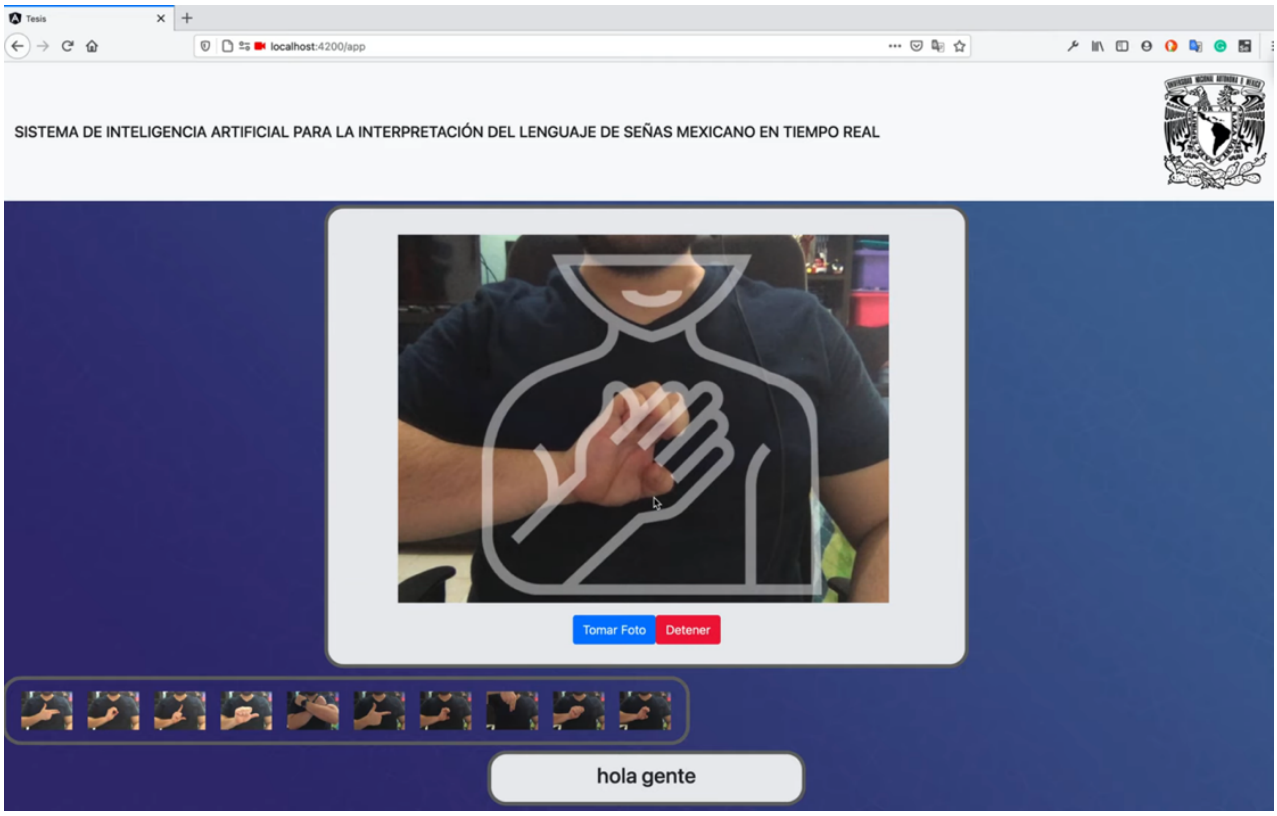


Figura 3.10. Pantalla Principal

Capítulo 4. Conjunto de Datos

"Tortura los datos y confesarán cualquier cosa".

- Ronald Coase

El problema es que, al igual que con los idiomas hablados, todos los idiomas de signos son diferentes. No existe una lengua de signos global compartida en todo el mundo [41]. Por lo tanto, una solución de traducción genérica no es suficiente para abordar este problema para todas las personas sordas del mundo. Hasta donde sabemos, en la actualidad no existe ninguna aplicación que los ayude a interpretar el lenguaje de señas genérico en texto. Si hubiera solo uno, por ejemplo, una herramienta que interprete solo el lenguaje de señas estadounidense no funcionaría en el lenguaje de señas mexicano [42] .

En esta sección se describirá detalladamente la recopilación y procesamiento de los datos del lenguaje de señas mexicano con un enfoque en la generación de un conjunto de datos genérico. La adquisición de imágenes y el procesamiento de imágenes se llevan a cabo en este paso del proceso. El análisis de características y la clasificación de imágenes se realizan en un paso posterior, cuando el modelo se vuelve a entrenar y se prueba en el nuevo conjunto de datos.

4.1 Adquisición de Data

El primer paso, la adquisición de imágenes, se realiza filmando todas las letras del alfabeto de señas mexicano durante 20 segundos, para un total de 10 sujetos. Se graban 21 letras se graban diez veces, para el caso de las letras J,K,Ñ,Q,X,Z, no se hacen grabación debido a que estas no son se realizan con una posición fija de la mano, sino, dinámica, es por esto que estas letras no se incluyen en el conjunto de datos y no están contempladas para la interpretación de las señas de este proyecto. Solo se incluye una parte del cuerpo y la parte de la cara, la persona está centrada y su mano está centrada frente al pecho. Finalmente se incluye un video del sujeto con los brazos cruzados para representar los espacios en blanco.

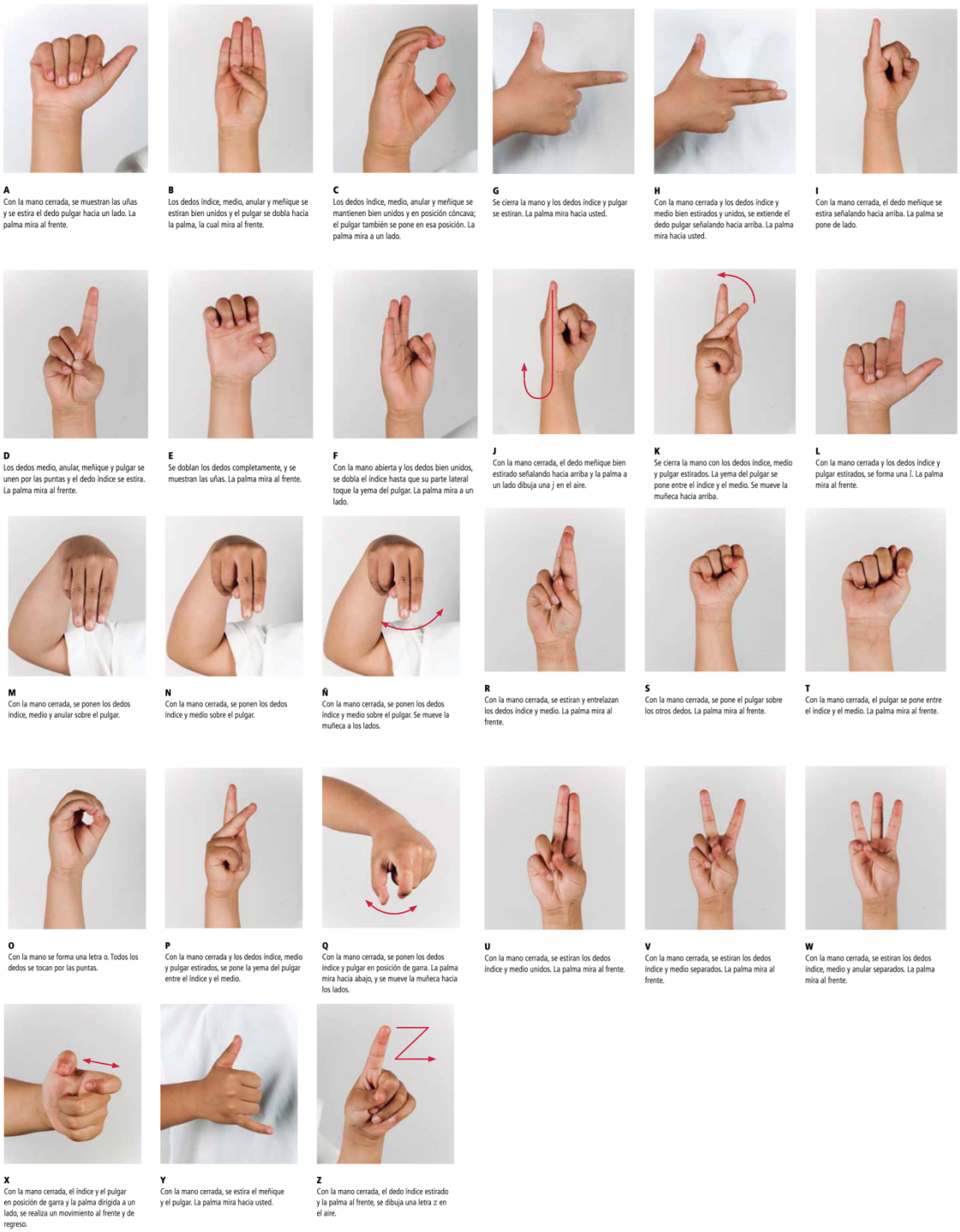


Figura 4.0. Abecedario Señas Mexicano definido por CONAPRED

Las grabaciones de los 10 sujetos son la base de los conjuntos de datos de entrenamiento, validación y prueba. Las etiquetas y posicionamiento de las manos está basado en el documento “Manos con Voz. Diccionario de Lengua de Señas Mexicanas” [43] proporcionado por CONAPRED. Las condiciones entre las grabaciones varían según el fondo y la ropa. Al filmar, las manos se giran ligeramente para permitir una mayor variación. Cada signo de una letra se clasifica como perteneciente a esa letra específica. Al recopilar datos diversos, se podría mejorar la capacidad de los modelos de aprendizaje automático para generalizar y diferentes tipos de firmantes y manos. Finalmente se crearon 10500 imágenes de dimensiones de 1080X720.

Id.	Edad	Genero	# Imágenes	Tiempo Grabación
1	25	Femenino	10500	420 segundos
2	25	Masculino	10500	420 segundos
3	62	Masculino	10500	420 segundos
4	55	Femenino	10500	420 segundos
5	28	Femenino	10500	420 segundos
6	31	Masculino	10500	420 segundos
7	25	Masculino	10500	420 segundos
8	60	Femenino	10500	420 segundos
9	75	Masculino	10500	420 segundos
10	58	Masculino	10500	420 segundos

Figura 4.1. Datos para la elaboración del Conjunto de datos

4.2 Procesado de Data

4.2.1 Procesamiento inicial

El procesamiento inicial se realiza en cada video. Utilizando un ambiente de ejecución de Python 3 y mediante la aplicación de métodos aportados por la biblioteca Open CV, se crea una función para leer todos los vídeos por letra y crear un fotograma por 0.04 segundos para que cada grabación se convierta en 500 imágenes. De esta manera se generan 500 imágenes por cada carácter del lenguaje de señas mexicano seleccionado, teniendo como resultado final 10,500 imágenes. Se generaron dos conjuntos de datos con estas 10,500 imágenes: uno con un pos-procesamiento y uno sin este último.

Figura 4.2. Imágenes recopiladas

4.2.2 Pos-procesamiento

Las 10,500 imágenes generadas conforman el primer conjunto de datos de lenguaje de señas mexicano. Las imágenes generadas varían sus tamaños entre 1080X720 y 480X848. Esto se debe a que las imágenes fueron recolectadas en con el uso de dos dispositivos diferentes. Con en el objetivo de normalizar las medidas de todas las imágenes en el conjunto de datos, se crea un pos-procesamiento de este conjunto de datos, se cambia el tamaño a 300X400 y se corta la imagen en el área próxima a la mano. Se ha escogido esta medida ya que una de las

arquitecturas finales utilizadas de transferencia de aprendizaje fue entrenada con imágenes con un tamaño de 299X299.

Para evitar el desbordamiento de la memoria, las imágenes del conjunto de datos de entrenamiento se dividen en tamaños de lote de 32,64 y 128 que se envían al modelo en lotes. Estos lotes son creados mediante las primitivas de datos ofrecidas por Pytorch.

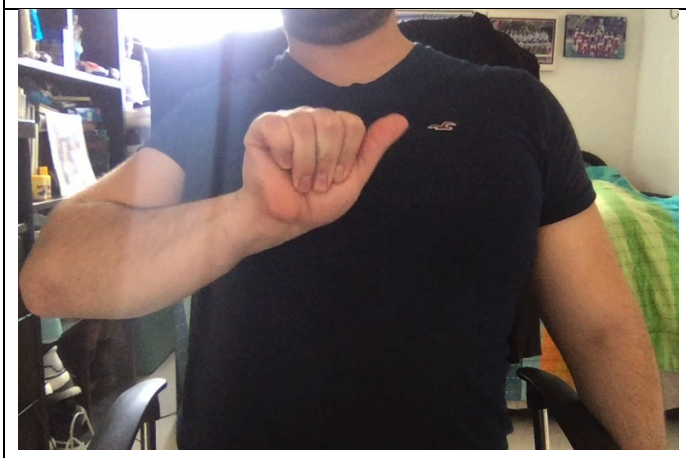
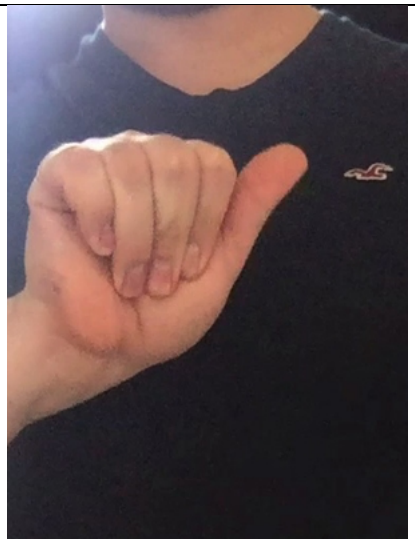
Imagen sin Pos-procesamiento	Imagen con Pos-procesamiento
	

Figura 4.3. Comparación Imagen con pos-procesamiento y sin este.

4.2.3 Generación de etiquetas

Los archivos con formato de valores separados por comas (CSV) se han utilizado para intercambiar y convertir datos entre varios programas de hojas de cálculo durante bastante tiempo [46]. Al mismo tiempo, varios programas y sistemas operativos han comenzado a utilizar diferentes tipos de MIME para este formato. Existe un RFC que ha documentado el formato de los archivos CSV y registra formalmente el tipo MIME "texto / csv" para CSV de acuerdo con RFC 2048 [47].

Contemporáneamente se ha demostrado que el formato CSV es el formato predominante en el panorama de datos abiertos [48]. La razón principal es la simplicidad e independencia de este formato: almacena datos tabulares en texto plano donde cada línea del archivo es un registro de datos. Cada registro consta de uno o más campos que están separados por un delimitador, generalmente una coma.

Históricamente, el formato CSV se desarrolló a partir de la necesidad de un formato de intercambio sin una formalización inicial; por lo tanto, existen muchas variaciones en el uso y no existe un formato "CSV" único y completamente especificado. En 2005, la IEFT publicó un primer intento de estandarizar CSV proponiendo un dialecto estricto y su implementación:

- Cada registro se ubica en una línea, delimitada por un salto de línea.
- El uso de una línea de encabezado (que aparece como la primera línea del archivo) es opcional, sin embargo, “la presencia o ausencia de la línea de encabezado debe indicarse mediante el parámetro de encabezado opcional de este tipo MIME” [49].
- Los campos de un registro están separados por una coma y cada línea contiene el mismo número de campos.

Sin embargo, se pueden observar muchas variantes de esto. especificación, y hoy en día CSV significa más "valores separados por caracteres" [50].

Considerando que se esta buscando publicar y poner a acceso publico todo el conjunto de datos de lenguaje de señas mexicana generado, se utilizo un archivo *csv* debido a que es el formato predominante en el panorama de datos abiertos [48].

A cada imagen se le asigna un nombre que se integra con los siguientes componentes:

- El primer componente corresponde a un valor numérico vinculado respectivamente a cada letra del conjunto de datos.
- El segundo componente corresponde al valor alfanumérico de la letra.
- El siguiente componente indica el genero de la persona en la imagen.
- El cuarto componente corresponde a un valor numérico de identificación que fue asignado a cada individuo que fue grabado para la creación del conjunto de datos.
- El último componente corresponde al valor correspondiente dentro de la secuencia [0-10500] para cada carácter.

0_a_0_06_2000.jpg

Figura 4.4. Imagen del conjunto de datos del lenguaje de señas mexicano generado.

Capítulo 5. Programación y Pruebas

5.1 Redes Neuronales Artificiales

5.1.1 Redes Neuronales Artificiales

En el mundo de la modelación de sistemas existen funciones que no pueden ser representadas por modelos lineales; por ejemplo, la regresión lineal no puede representar las funciones cuadráticas y los clasificadores lineales no pueden representar la función lógica XOR [55]. Existen diversas formas particulares de solucionar este problema: definiendo características o funciones básicas.

Es posible representar funciones complejas no lineales conectando juntas muchas unidades de procesamiento simples en una red neuronal artificial, cada una de las cuales calcula una función lineal, posiblemente seguida de una no linealidad. Además, estas unidades pueden calcular algunas funciones sorprendentemente complejas. Estas redes se denominan perceptrones multicapa.

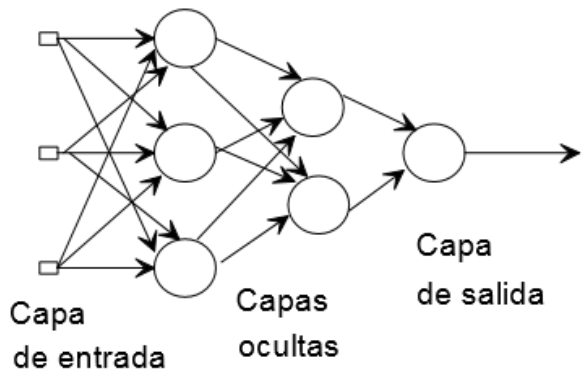


Figura 5.0. Estructura de un Perceptrón multicapa.

En los últimos años, las técnicas de aprendizaje automático [52] y, en particular, de aprendizaje profundo [53, 54] han demostrado ser muy eficaces para automatizar tareas complejas. Las redes neuronales artificiales se inspiran en los primeros modelos de procesamiento sensorial del cerebro. Se puede crear una red neuronal artificial simulando una red de neuronas en una computadora. Al aplicar algoritmos que imitan los procesos de las neuronas reales, se puede hacer que la red "aprenda" a resolver numerosos tipos de problemas. Una neurona modelo se denomina unidad de umbral y su función se ilustra en la figura 5.0. :

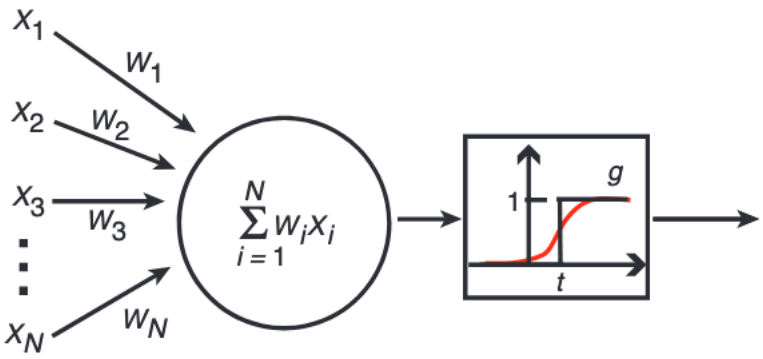


Figura 5.0. Diagrama modelo Redes Neuronales Artificiales

Estos modelos reciben información de varias unidades o fuentes externas, pesa cada entrada y las suma. Si la entrada total está por encima de un umbral/sesgo, la salida de la unidad es uno; de lo contrario, es cero. Por lo tanto, la salida cambia de 0 a 1 cuando la suma ponderada total de entradas es igual al umbral. Los puntos en el espacio de entrada que satisfacen esta condición definen un hiperplano. En dos dimensiones, un hiperplano es una línea, mientras que, en tres dimensiones, es un plano normal. Los puntos en un lado del hiperplano se clasifican como 0 y los del otro lado como 1. Significa que un problema de clasificación puede resolverse mediante una unidad de umbral si las dos clases pueden separarse mediante un hiperplano. Se dice que tales problemas son linealmente separables [51].

5.1.2 Antecesores

- ADALINE (Adaptive Linear Elements)

ADALINE [57] se desarrolló por primera vez para reconocer patrones binarios, de modo que, si se trataba de leer bits de transmisión desde una línea telefónica, podía predecir el siguiente bit [56].

El funcionamiento de la red ADALINE se basa en tomar la suma de los pesos de las entradas y producir una salida con 0 o 1 dependiendo si pasa o no un umbral. Este comportamiento hace analogía al funcionamiento de una neurona que se activa si la actividad total procedente de las conexiones con las otras neuronas sobrepasa un nivel. Esta red consiste en un ALC (Combinador lineal adaptativo) y un cuantizador (función bipolar de salida).

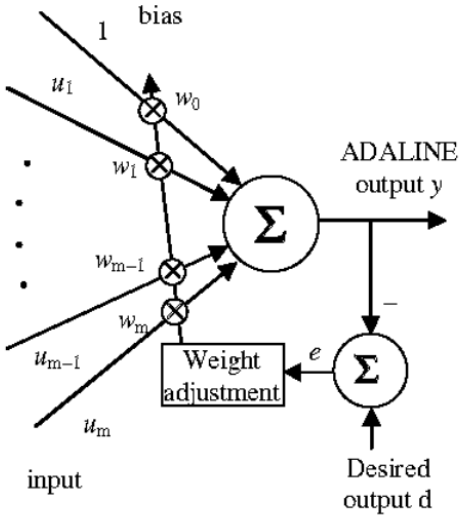


Figura 5.1. Arquitectura unidad ADALINE

- MADALINE (Multiple Adaptive Linear Elements)

MADALINE fue la primera red neuronal que se aplicó a un problema del mundo real, utilizando un filtro adaptativo que elimina las opciones en las líneas telefónicas [56]. La arquitectura de la red MADALINE consiste en unidades ADALINE conectadas que pueden ser organizadas en capas [58]. MADALINE logra producir funciones más complicadas abarcando soluciones más complejas que las de problemas linealmente separables. Esta arquitectura consta de una capa de ADALINES y una función de mayoría cuya respuesta binaria depende de las respuestas de las ADALINES.

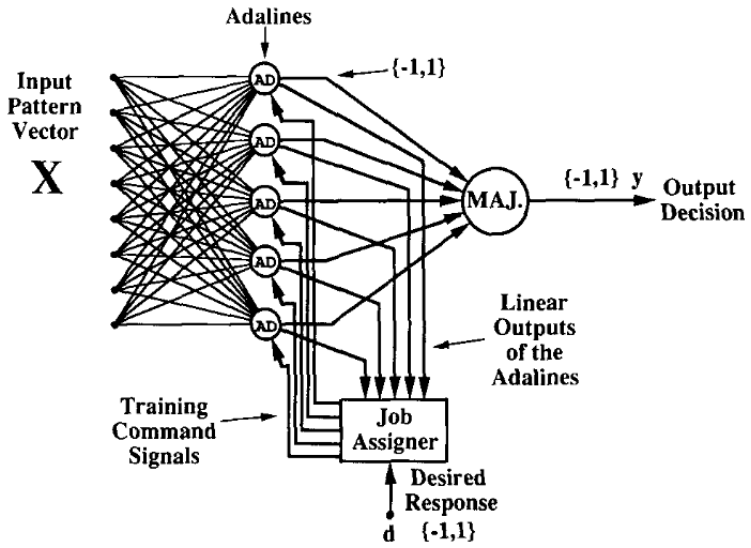


Figura 5.2. Un ejemplo de cinco Adaline de la arquitectura Madeleine

5.1.3 Red Perceptrón (Unidad)

Este es un modelo cuyo principal objetivo es la clasificación de datos linealmente separables. Este modelo no es capaz de clasificar datos que no sean linealmente no separables.

El perceptrón consta de 4 partes.

- Valores de entrada (x) o una capa de entrada: en este, el perceptrón recibe los datos y un valor de umbral/sesgo (b) con los que trabajara.
- Pesos (w): los pesos son valores vectoriales inicializados aleatoriamente. Todos los valores de entrada x se multiplican por estos pesos w.
- Suma neta: Se suman todos los valores multiplicados: Suma ponderada.
- Función de activación: Se aplica esa suma ponderada a la función de activación (Φ) correcta.

Este comportamiento se describe en la ecuación 1:

$$a = \phi\left(\sum_j w_j x_j + b\right) \quad (1)$$

Dónde:

- *xj* son las entradas al perceptrón.
- *wj* son los pesos.
- *b* es el umbral.
- Φ es la función de activación no lineal.
- *a* es la activación de la unidad.

5.1.3.1 Algoritmo de aprendizaje: perceptrón

1. Se inicializan pesos y sesgo con zeros o un número aleatorio pequeño
2. Para cada ejemplo en el conjunto de entrenamiento :
   2.1 Se calcula la salida:

$$\hat{y}^{(i)} = \varphi(w(t)^{\top} x^{(i)} + b) \qquad (2)$$

2.2 Se actualiza cada peso w_j,j = 1,...,d y el sesgo b

$$w_j[t+1] = w_j[t] + (y^{(i)} - \hat{y}^{(i)}) \cdot x^{(i)} j \qquad (3)$$

$$b[t+1] = b[t] + (y^{(i)} - \hat{y}^{(i)}) \qquad (4)$$

3. Es necesario realizar este proceso hasta que este converja o hayan pasado un número de épocas

5.1.4 Red Perceptrón Multicapa

Las Redes Perceptrón Multicapa (MLP) pueden entenderse básicamente como una red de múltiples neuronas artificiales en múltiples capas. Aquí, la función de activación no es lineal (como en Adaline), pero se utiliza una función de activación no lineal como la sigmoidea logística o la tangente hiperbólica.

Estas deben ser capaces de establecer en una serie de reglas y aplicarlas en un orden jerárquico. Los problemas no retroalimentados no pueden establecer un orden jerárquico es por esto por lo que los sistemas expertos no funcionan en esta modelación. Una red semántica no puede ser identificada por sistemas expertos.

En esta red los valores de los pesos son inicializados aleatoriamente; los MLP modelan las funciones a través de un algoritmo de retropropagación (definido en la sección 6.3.4.2); estos deben incluir gradientes (un gradiente debe ser derivable por eso se usan funciones sigmoides).

La arquitectura del MLP consiste en:

- Capa de Entrada: En esta es donde la red recibe los datos con los que trabajara. El número de neuronas en la capa de entrada es igual al número de variables / instancias de datos que tiene más un valor umbral.
- Capas Ocultas: Las capas ocultas permiten que la red neuronal aprenda clasificaciones que no son linealmente separables. Por ejemplo: una red neuronal que tiene solo 2 nodos de entrada conectados directamente a un nodo de salida puede aprender a funcionar como una compuerta AND o una compuerta OR. Una red neuronal de 2 entradas y 1 salida con al menos dos nodos ocultos también puede aprender a funcionar como una compuerta XOR.
- Capas de Salida: La última capa es la capa de salida, y tiene una unidad para cada valor de salida de la red (es decir, una soló unidad en el caso de regresión o clasificación binaria, o unidades K en el caso de la clasificación de clase K).

5.1.5 Funciones de Activación

5.1.5.1 Sigmoidea

La función sigmoidea solo producirá números positivos entre 0 y 1. Esta función de activación es más útil para entrenar datos que están entre 0 y 1. Es una de las funciones de activación más utilizadas [59].

La función de activación sigmoidea esta definida por la ecuación 5:

$$\sigma\,(z) = \frac{1}{1+e^{-z}} \qquad (5)$$

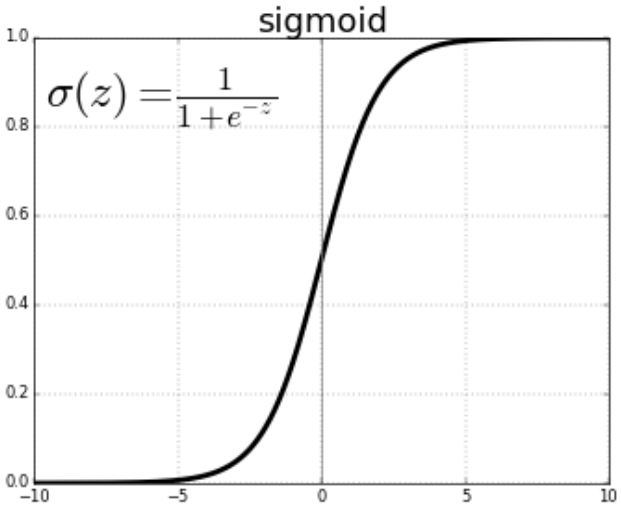


Figura 5.3. Función de activación Sigmoidea

5.1.5.2 Tangente Hiperbólica

Las redes pueden presentar valores de salida no centrados en cero, esto degrada el rendimiento de la red. La función de activación Tangente Hiperbólica denominada función de activación multiestado (MSAF) [60], es capaz de representar tres o más estados para mejorar el rendimiento de clasificación de la función de activación saturada.

Distintos autores [61] propusieron la función de activación de la tangente hiperbólica que penaliza el gradiente de su parte negativa para evitar el problema de saturación. La función de activación tangencial hiperbólica esta definida por la ecuación 6:

$$\sigma\,(z) = \frac{e^{z}-e^{-z}}{e^{z}-e^{-z}} \qquad (6)$$

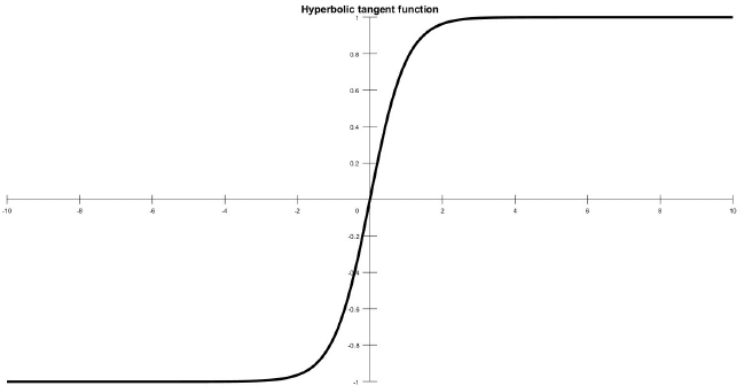


Figura 5.4. Función de activación tanh

5.1.5.3 ReLU

La función ReLU es una función de activación insaturada [62]. Esta supera el problema de saturación y el problema del gradiente de desaparición porque los términos exponenciales se eliminan en su retropropagación. Cuando los datos caen en la región negativa de ReLU es incapaz de reactivar durante el proceso de entrenamiento [63]. Existen distintas variantes de esta función de activación que han intentado resolver este problema, por ejemplo: Leaky ReLU (LReLU) resolvió este problema asignando una variable constante en la región negativa de la función de activación de ReLU [63], la función de activación adaptativa como ReLU paramétrica (PReLU) [64] asignó un coeficiente entrenable en la parte negativa de LReLU para reemplazar el coeficiente constante en la función de activación de LReLU. La función de activación rectificadora esta definida por la ecuación 7:

$$\sigma\,(z) = \max(0, z) \qquad (7)$$

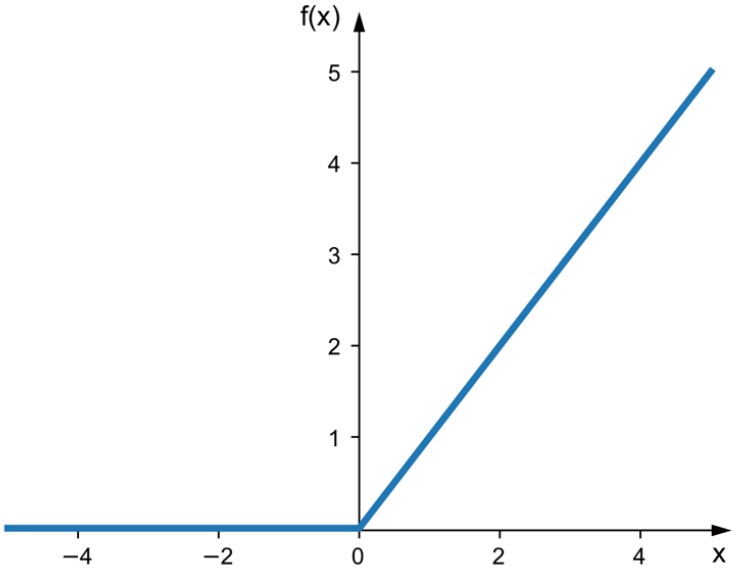


Figura 5.5. Función de activación ReLU.

5.1.6 Función de Pérdida

En el caso de contar con una red que este trabajando con funciones de activación sigmoidea o logística es recomendable utilizar la función de pérdida:

- Entropía Cruzada Binaria (ECB) ; definida en las ecuaciones 8,9 y 10:

$$ECB(y,\hat{y}) = -\sum_{i=1}^{N}\left[y^{(i)}\log\left(\hat{y}^{(i)}\right) + \left(1-y^{(i)}\right)\log\left(1-\hat{y}^{(i)}\right)\right] \quad (8)$$

$$\frac{\partial ECB}{\partial W_j} = \frac{1}{N}\sum_{i=1}^{N}\left[\left(\hat{y}^{(i)} - y^{(i)}\right)\cdot x_j^{(i)}\right] \quad (9)$$

$$\frac{\partial ECB}{\partial b} = \frac{1}{N}\sum_{i=1}^{N}\left[\hat{y}^{(i)} - y^{(i)}\right] \quad (10)$$

Dónde:

- *xj* son las entradas a la red
- *wj* son los pesos
- *y son los valores verdaderos*
- $\hat{y}$ son los valores resultados de la red
- b son los valores de umbral de la red
- N corresponde al número de iteraciones.

A diferencia del proceso de los problemas de clasificación binaria, la clasificación multiclase no tiene que elegir un umbral de puntuación para realizar predicciones. En el caso de que la naturaleza del problema requiera que se clasifique más de una clase, se debe implementar una ECB de cada categoría como función de perdida

- La función ECB de cada categoría esta definida en la ecuación 11 :

$$ECB(y_k, \hat{y}_k) = -\sum_{i=1}^{N} \left[y_k^{(i)} \log(\hat{y}_k^{(i)}) + \left(1 - y_k^{(i)}\right) \log\left(1 - \hat{y}_k^{(i)}\right) \right] \qquad (11)$$

Dónde:

- *xj* son las entradas a la red.
- *wj* son los pesos.
- *y son los valores verdaderos.*
- $\hat{y}$ son los valores resultados de la red.
- K es el número de clases.
- N corresponde al número de iteraciones.

Para la clasificación multiclase las neuronas de la capa de salida tienen una función de activación softmax . La función softmax es una función que convierte un vector de K valores reales en un vector de K valores reales que suman 1, generando una función de distribución de probabilidad para K elementos. La función softmax transforma los valores de entrada en valores entre 0 y 1, para que puedan interpretarse como probabilidades. Si una de las entradas es pequeña o negativa, la función softmax la convierte en una probabilidad pequeña, y si una entrada es grande, la convierte en una probabilidad grande, pero siempre permanecerá entre 0 y 1 [65].

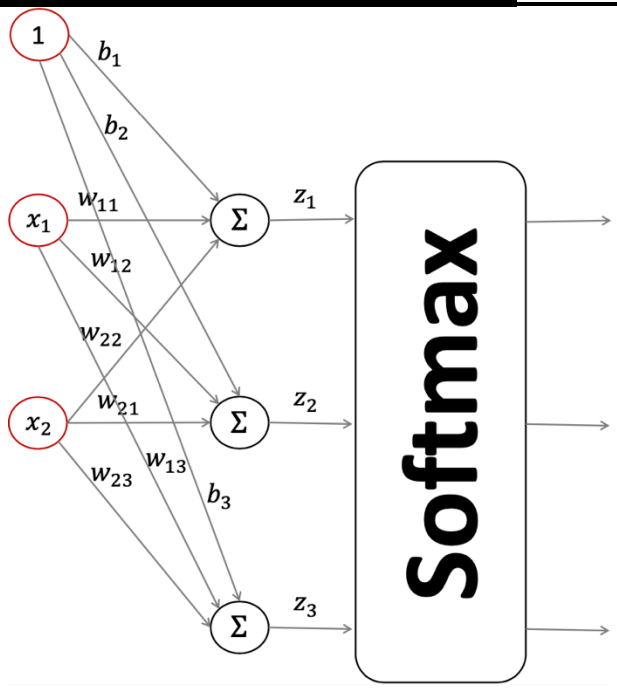


Figura 5.6. Clasificación Softmax

Todos los elementos del vector de entrada a la función softmax pueden tomar cualquier valor real, positivo, cero o negativo. Se aplica una función exponencial estándar a cada elemento del vector de entrada. Esto da un valor positivo por encima de 0, que será muy pequeño si la entrada fue negativa y muy grande si la entrada fue grande. Sin embargo, todavía no está fijo en el rango (0, 1) que es lo que se requiere de una probabilidad. Se requiere una normalización. Esta asegura que todos los valores de salida de la función sumen 1 y cada uno esté en el rango (0, 1), constituyendo así una distribución de probabilidad válida.

Los valores en la ultima capa se mandan en cada iteración, estos son los que integran el vector de K valores de entradas a la función softmax. Esta los convierte a una función de probabilidad cuya sumatoria es igual a 1. La clase con la mayor probabilidad es la salida de la red clasificatoria.

- La función softmax esta definida como se muestra en la ecuación 12:

$$softmax(\hat{z})i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \qquad (12)$$

Dónde:

- $\hat{z}$ El vector de entrada a la función softmax, compuesto por (z0, ... zK).
- z_i Todos los valores de zi son los elementos del vector de entrada a la función softmax.
- e^{z_i} La función exponencial estándar aplicada a cada elemento del vector de entrada.
- $\sum_{j=1}^{K} e^{z_j}$ Término de normalización.
- K El número de clases en el clasificador de clases múltiples.

Finalmente, las ecuaciones que describen el comportamiento de la función entropía cruzada categórica (ECC) están definidas en las ecuaciones 13,14 y 15:

$$ECC(Y, \hat{Y}) = -\sum_{i=1}^{N}\sum_{k=1}^{K}[\, y_k^{(i)} \cdot \log\left(\frac{e^{z_k^{(i)}}}{\sum_{j=1}^{K} e^{z_j^{(i)}}}\right)]x_j^{(i)} \qquad (13)$$

$$\frac{\partial ECC}{\partial W_j} = \frac{1}{N}\sum_{i=1}^{N}\left[\, (\hat{y}^{(i)} - y^{(i)}) \otimes x_j^{(i)}\right] \qquad (14)$$

$$\frac{\partial ECC}{\partial b} = \frac{1}{N}\sum_{i=1}^{N}\left[\hat{y}^{(i)} - y^{(i)}\right] \qquad (15)$$

Dónde:

- *xj* son las entradas a la red
- *wj* son los pesos
- *y son los valores verdaderos*
- $\hat{y}$ son los valores resultados de la red
- b son los valores de umbral de la red
- N corresponde al número de iteraciones.

- K es el número de clases.

6.3.4 Aprendizaje

Si el problema de clasificación es separable, todavía necesitamos una forma de establecer los pesos y el umbral, de modo que la unidad de umbral resuelva correctamente el problema de clasificación. Esto se puede hacer de manera iterativa presentando ejemplos con clasificaciones conocidas, uno tras otro. Este proceso se llama aprendizaje o formación, ya que es parecido al proceso que atravesamos cuando aprendemos algo. La simulación del aprendizaje por computadora implica realizar pequeños cambios en los pesos y el sesgo cada vez que se presenta un nuevo ejemplo de tal manera que se mejora la clasificación. El entrenamiento se puede implementar mediante algoritmos diferentes.

Durante el entrenamiento, el hiperplano se mueve hasta encontrar su posición correcta en el espacio, después de lo cual no cambiará tanto [51].

6.3.4.1 Propagación hacia Adelante

La propagación hacia adelante toma como entrada los valores de la capa anterior (los datos de entrada para la primera capa), realizan la suma ponderada y generan un vector numérico. Finalmente, este vector pasa por la función de activación de la capa y se manda hacia la siguiente capa.

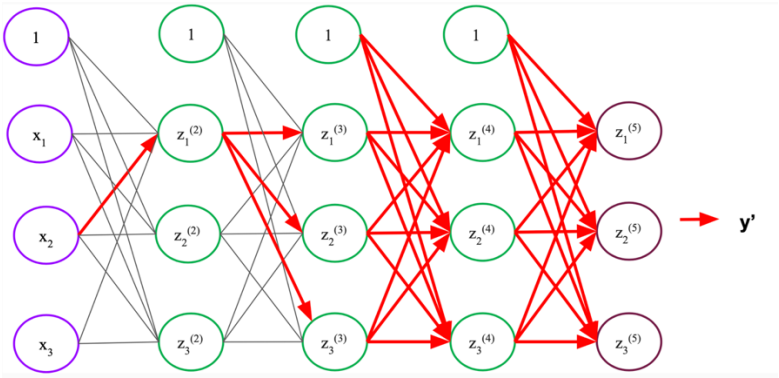


Figura 5.7. Propagación hacia adelante en redes densas

Considerando una red densa con 1 capa de entrada, 1 capa oculta con o neuronas con activación sigmoide y 1 neurona de salida con activación lineal. La propagación hacia adelante estaría dada de la manera en que se describe en las ecuaciones 16 a la 21:

$$a\{1\} = x(i) \quad (16)$$

$$z\{2\} = W\{1\} \cdot a\{1\} \quad (17)$$

$$a\{2\} = \varphi(z\{2\}) \quad (18)$$

$$z\{3\} = W\{2\} \cdot a\{2\} \quad (19)$$

$$a\{3\} = \varphi(z\{3\}) \quad (20)$$

$$\hat{y} = a\{3\} \quad (21)$$

Dónde:

- a hace referencia a los vectores de datos después de pasar por una capa
- x son los datos en la entrada de la red
- W son los pesos de la red
- z hace referencia a los datos de una capa anterior producto con los respectivos pesos
- φ hace referencia a la funciones de activación
- $\hat{y}$ es la salida de la red

6.3.4.2 Propagación hacia atrás (Retropropagación)

Los algoritmos de retropropagación tienen como objetivo minimizar, maximizar, etc. (según sea el criterio de optimización) la función de costo ajustando los pesos y sesgo de la red. El nivel de ajuste está determinado por los gradientes de la función de costo con respecto a esos parámetros. La función de costo es la sumatoria de los valores generados por la función de pérdida en cada iteración. El gradiente de la función de activación mide la sensibilidad al cambio del valor de la función (valor de salida) con respecto a un cambio en su argumento x (valor de entrada). En otras palabras, el

gradiente nos dice la dirección en la que va la función, este muestra cuánto debe cambiar el parámetro x (en dirección positiva o negativa) para optimizar.

El algoritmo de aprendizaje de retropropagación funciona para redes de retroalimentación con salida continua. El entrenamiento comienza estableciendo todos los pesos de la red en pequeños números aleatorios. Ahora, para cada ejemplo de entrada, la red da una salida, que comienza de forma aleatoria. Se mide la diferencia al cuadrado entre esta salida y la salida deseada: la clase o valor correcto. La suma de todos estos números en todos los ejemplos de entrenamiento se denomina error total de la red. Si este número fuera cero, la red sería perfecta, y cuanto menor sea el error, mejor será la red. Al elegir los pesos que minimizan el error total, se puede obtener la red neuronal que mejor resuelve el problema en cuestión. En la retro propagación, los pesos y los sesgos se cambian cada vez que se presenta un ejemplo, de modo que el error se reduce gradualmente. Esto se repite, a menudo cientos de veces, hasta que el error ya no cambia [51]. En la propagación hacia atrás, una técnica de optimización numérica llamada descenso de gradiente es utilizada.

6.3.4.2.1 Algoritmo de retropropagación

1. Se propagamos cada entrada $x^{(i)}$ hacia adelante para generar la correspondiente salida $\hat{y}^{(i)}$
2. Se calculan derivadas parciales de la pérdida respecto a cada peso y umbral capa por capa, empezando con la de salida y propagándolas hacia atrás para calcular las de la capa anterior

El calculo del gradiente de la función de pérdida con respecto a $W^{\{2\}}$ se define con las ecuaciones 22 a la ecuación 26 (Suponiendo una tarea de regresión y la función de pérdida ECM):

$$\frac{\partial ECM}{\partial W^{\{2\}}} = \frac{\partial \sum_{i}^{\frac{1}{2}} (y^{(i)} - \hat{y}^{(i)})^2}{\partial W^{\{2\}}} \qquad (22)$$

$$= \frac{\sum_i \partial \frac{1}{2}(y^{(i)} - \hat{y}^{(i)})^2}{\partial W^{\{2\}}} \quad (23)$$

$$\frac{\partial \frac{1}{2}(y^{(i)} - \hat{y}^{(i)})^2}{\partial W^{\{2\}}} = (y - \hat{y}) \cdot \left(-\frac{\partial \hat{y}}{\partial W^{\{2\}}}\right) \quad (24)$$

$$= (y - \hat{y}) \cdot \left(-\frac{\partial \hat{y}}{\partial z^{\{3\}}} \cdot \frac{\partial z^{\{3\}}}{\partial W^{\{2\}}}\right) \quad (25)$$

$$= (y - \hat{y}) \cdot \left(-\frac{\partial \hat{y}}{\partial z^{\{3\}}} \cdot a^{\{2\}}\right) \quad (26)$$

3. Se calcula el gradiente de la función de pérdida respecto a $W^{\{n\}}$
4. Se actualizan los pesos y umbrales con base al calculo del gradiente
5. Se continua este proceso hasta llegar a uno de los valores de optimización buscados.

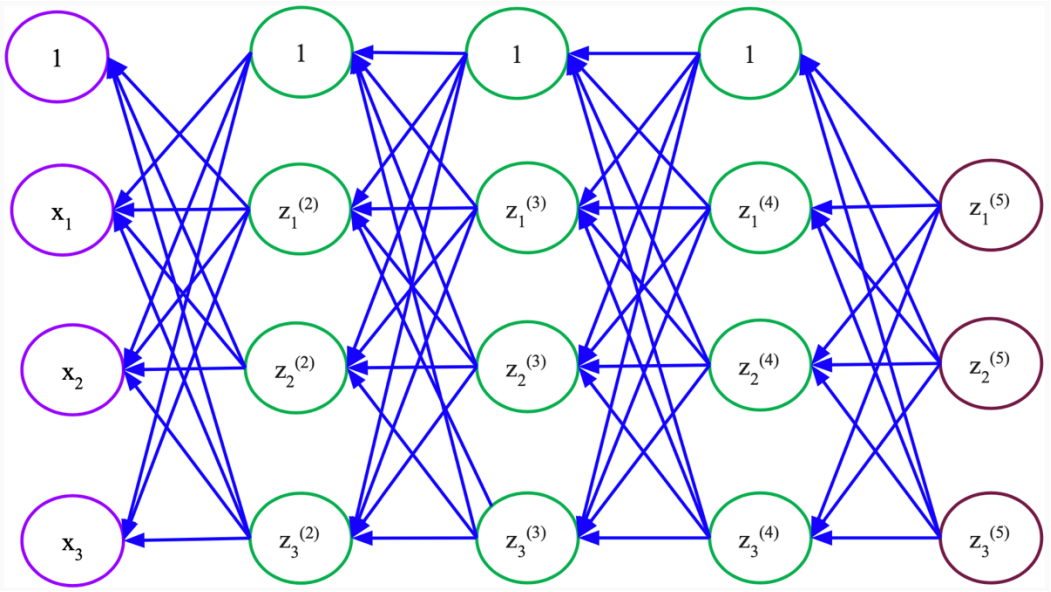


Figura 5.8. Cálculo del gradiente por retropropagación

6.3.4.2.2 Descenso por Gradiente (GD)

El descenso de gradientes es uno de los algoritmos más populares para realizar la optimización y la forma más común de optimizar las redes neuronales. Al mismo tiempo, cada biblioteca principal de Aprendizaje Profundo de última generación contiene implementaciones de varios algoritmos para optimizar el descenso de gradientes. Existen distintas variantes de descenso de gradiente, que difieren en la cantidad de datos que usamos para calcular el gradiente de la función objetivo. Dependiendo de la cantidad de datos, hacemos un compromiso entre la precisión de la actualización de los parámetros y el tiempo que lleva realizar una actualización. Este es un algoritmo iterativo de primer orden que va moviendo los pesos w y umbrales b hacia donde la pérdida descienda más rápido en el vecindario, esto se define como lo marca la ecuación 27:

$$\theta[t+1] = \theta[t] - \alpha \nabla L(\theta[t]) \qquad (27)$$

Donde:

- $\theta = \{w,b\}$
- $\nabla L(\theta[t]) = \left[\frac{\partial L}{\partial \theta_0[t]}, \dots, \frac{\partial L}{\partial \theta_d[t]}\right]$
- α corresponde a la tasa de aprendizaje

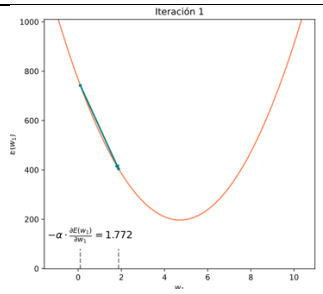
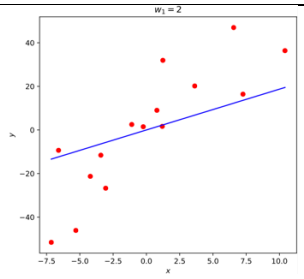
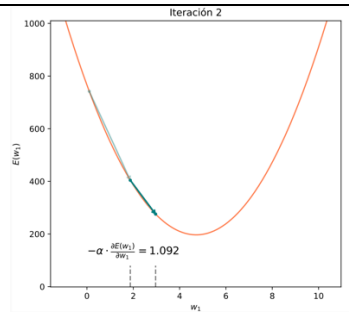
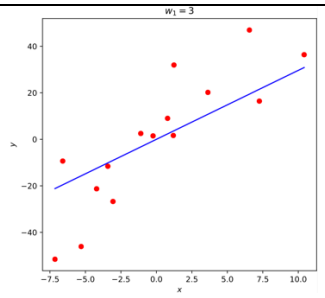
6.3.4.2.3 Descenso de gradiente estocástico (SGD)

El descenso de gradiente estocástico (SGD) realiza una actualización de parámetros para cada entrenamiento para cada ejemplo x(i) y etiqueta y(i). Este es una aproximación estocástica de descenso por gradiente:

- Estima $\nabla L(\theta[t])$ y actualiza pesos y sesgos con un minilote de b ejemplos de entrenamiento
- b es un hiperparámetro

- Es común dividir y ordenar aleatoriamente el conjunto de N ejemplos totales de entrenamiento en k mini lotes (b × k ≈ n); una época o iteración ocurre cada vez que se han considerado los k mini lotes.

El entrenamiento de redes neuronales multicapa usualmente se lleva a cabo través del descenso por gradiente estocástico (SGD) o variantes. En la práctica se ha observado que en el entrenamiento de redes neuronales con SGD encuentra mejores soluciones, especialmente con minilotes pequeños [67,68,69]. Sin embargo, existe un problema, que es el calculo eficiente de las derivadas parciales respecto a los pesos y sesgos de las capas ocultas. Las implementaciones de este algoritmo en las librerías de aprendizaje profundo hacen que el uso de este sea sencillo.

Iteración	Gradiente	Resultados
1		
2		

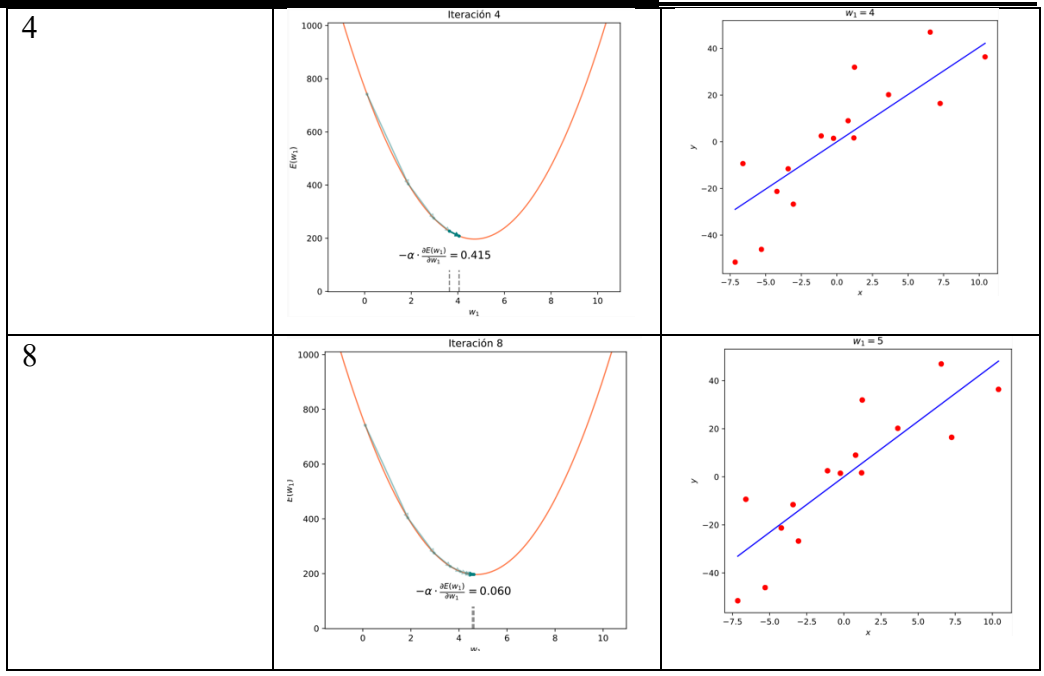


Figura 5.9. Ejemplo del algoritmo de descenso por gradiente.

6.3.4.3 Sobreajuste

El sobreajuste ocurre cuando la red tiene demasiados parámetros para aprender de la cantidad de ejemplos disponibles. Para cualquier método de clasificación o regresión, las redes neuronales parecen especialmente propensas a la parametrización excesiva. Por ejemplo, una red con 10 unidades ocultas para resolver nuestro problema de ejemplo tendría 221 parámetros: 20 pesos y un umbral para las 10 unidades ocultas y 10 pesos y un umbral para la unidad de salida. Son demasiados parámetros para aprender de 100 ejemplos. Es poco probable que una red que sobreajuste los datos de entrenamiento se generalice bien a las entradas que no están en los datos de entrenamiento [51]. Hay muchas formas de limitar el sobreajuste, pero las más comunes incluyen promediar en varias redes, regularización y usar métodos de estadísticas bayesianas.

6.3.4.4 Generalización

Para estimar el rendimiento de generalización de la red neuronal, se necesita para probarlo en datos independientes, que no se han utilizado para entrenar la red. Esto generalmente se hace mediante validación cruzada [51], donde el conjunto de datos se divide en, por ejemplo, diez conjuntos de igual tamaño. Luego, la red se entrena en nueve conjuntos y se prueba en el décimo, y esto se repite diez veces, por lo que todos los conjuntos se utilizan para la prueba. Esto da una estimación de la capacidad de generalización de la red; es decir, su capacidad para clasificar insumos en los que no fue entrenado. Para obtener una estimación no sesgada, es muy importante que los conjuntos individuales no contengan ejemplos que sean muy similares.

6.3.5 Inicializadores de pesos y sesgos

Una Red Neuronal Artificial contiene una secuencia de vectores numéricos denominados pesos y umbrales. Estos son los que serán actualizados mediante el algoritmo de aprendizaje, en este caso retropropagación. Estos vectores deben tener algún valor inicial antes de comenzar a ser actualizados por el algoritmo. Existen distintos tipos de técnicas de inicialización para pesos y umbrales, en este proyecto se utilizaron dos: Inicialización Xavier e Inicialización con Ceros.

6.3.5.1 Inicialización Xavier

El método de inicialización de Xavier se introdujo en el artículo de Xavier en 2010 [90]. En este, se propone que los pesos se elijan de una distribución uniforme aleatoria acotada entre:

$$\left(-\frac{\sqrt{6}}{\sqrt{n_i+\sqrt{n_j}}}, \frac{\sqrt{6}}{\sqrt{n_i+\sqrt{n_j}}}\right)$$

Dónde:

- n_i es el número de conexiones de red entrantes.
- n_j es el número de conexiones de red salientes de esa capa.

El objetivo inicial de Xavier fue explorar por qué el descenso de gradiente estándar a partir de la inicialización aleatoria funciona mal en redes neuronales profundas. Se concluyó que la activación logística sigmoidea no es adecuada para redes neuronales profundas con inicialización aleatoria [91]. Esto conduce a la saturación de las capas iniciales en una etapa muy temprana del entrenamiento. También se encontró que las unidades saturadas pueden salir de la saturación por sí mismas con la inicialización adecuada [90]. Para encontrar un mejor rango para valores de peso iniciales aleatorios, equipararon la varianza de cada capa. La varianza de la entrada y la salida de una capa se calculó de modo que no se pierda mucha varianza entre la entrada y la salida de una sola capa. Con esta idea, la solución encontrada es un rango de valores de peso iniciales. El rango es:

$$\mathrm{U}\left[\left(-\frac{\sqrt{6}}{\sqrt{n_i+\sqrt{n_j}}},\frac{\sqrt{6}}{\sqrt{n_i+\sqrt{n_j}}}\right)\right]$$

Dónde:

- n_i es el número de conexiones de red entrantes.
- n_j es el número de conexiones de red salientes de esa capa.
- U es una distribución uniforme aleatoria.

6.3.6 Optimizadores

A diferencia de la creciente complejidad de las arquitecturas de redes neuronales [92,93,94], los métodos de entrenamiento siguen siendo relativamente simples [99]. La mayoría de los métodos de optimización prácticos para redes neuronales profundas se basan en el algoritmo de descenso de gradiente estocástico. Sin

embargo, la tasa de aprendizaje de SGD, como un hiperparámetro, es a menudo difícil de ajustar, ya que las magnitudes de los diferentes parámetros varían ampliamente y se requiere un ajuste a lo largo del proceso de entrenamiento [99]. Para abordar este problema, se desarrollaron varias variantes adaptativas de SGD, incluyendo Adagrad [95], Adadelta [96], RM-Sprop [97], Adam [98]. La funcionalidad de estos algoritmos tiene como objetivo adaptar la tasa de aprendizaje a diferentes parámetros de forma automática, en función de las estadísticas de gradiente.

6.3.6.1 Optimizador Adam

Adam es uno de los distintos enfoques propuestos para ajustar automáticamente las tasas de aprendizaje para hacer frente a la dificultad de encontrar tasas de aprendizaje adecuadas. Adam, propuesto por Kingma y Ba [100] es también un método para ajustar automáticamente la tasa de aprendizaje en SGD.

En el método Adam, la actualización de los parámetros se realiza de la siguiente manera [101]:

$$\theta^{nueva} = \theta^{anterior} - K \cdot \frac{\widehat{m}}{\sqrt{\widehat{v}}+\varepsilon} \qquad (28)$$

Dónde:

- ε es un hiperparámetro del método de Adam
- $\widehat{m} = \frac{m}{(1-\lambda_1^t)}$
- $\widehat{v} = \frac{v}{(1-\lambda_2^t)}$

Dónde:

- λ_1 y λ_2 son hiperparámetros; estos son utilizados para calcular las medias exponenciales correspondientes.
- *m, v* son la media móvil exponencial de $\nabla\theta$ y $\nabla\theta^2$, respectivamente.

6.4 Redes Neuronales Convolucionales (CNN)

Las Redes Neuronales Convolucionales son un tipo de Red Neuronal Artificial. El término Aprendizaje Profundo hace referencia a las redes neuronales artificiales con múltiples capas. Durante las últimas décadas, se ha considerado una de las herramientas más poderosas y se ha vuelto muy popular en la literatura, ya que es capaz de manejar una gran cantidad de datos. Una de las redes neuronales profundas más populares es la red neuronal convolucional. Toma este nombre de una operación matemática lineal entre matrices llamada convolución. CNN tiene múltiples capas; incluida la capa convolucional, la capa de no linealidad, la capa de agrupación y la capa completamente conectada. La CNN tiene un excelente desempeño especialmente en aplicaciones que tratan con datos de imágenes, en general de la rama de la visión por computadora [66].

5.2.1 Elementos de una Red Neuronal Convolucional

A. Capa de Convolución

La entrada de una red neuronal puede ser una imagen o un video que tiene valores de sensor de ancho y alto (L × L) y las profundidades están asociadas con diferentes marcos de tiempo [70,71,72].

Una red puede recibir píxeles sin procesar como entrada. Por lo tanto, para conectar la capa de entrada a una sola neurona (por ejemplo, en la capa oculta en el perceptrón multicapa), como ejemplo, debería haber 32 × 32 × 3 conexiones de peso para el conjunto de datos con imágenes de tamaño 32X32 pixeles con una escala RGB [66].

La idea principal de una capa de convolución es realizar una extracción de características de las imágenes a través de filtros. Se define una matriz numérica de tamaño (n x m) y se utiliza como filtro. Este filtro pasa (convoluciona) por toda la imagen realizando operaciones de convolución y genera un nuevo vector de datos

con las características de la imagen analizada. Esta operación es equivariante y las representaciones obtenidas son más eficientes.

Existen distintos tipos de convolución:

- Convolución en 1 dimensión, definida en dos casos:

  Continua: $s(t) = (x * k)(t) = \int_m x(m)k(t - m)dm$ (29)

  Discreta: $s(i) = (x * k)[i] = \sum_{m=-\infty}^{\infty} x[m]k[i - m]$ (30)

- Convolución en 2 dimensiones, definida en dos casos:

  Continua: $S[i,j] = (I * K)[i,j] \sum_m \sum_n I[m,n]K[i-m, j-n]$ (31)

  Discreta: $S[i,j] = (K * I)[i,j] \sum_m \sum_n I[i-m, j-n]K[m,n]$ (32)

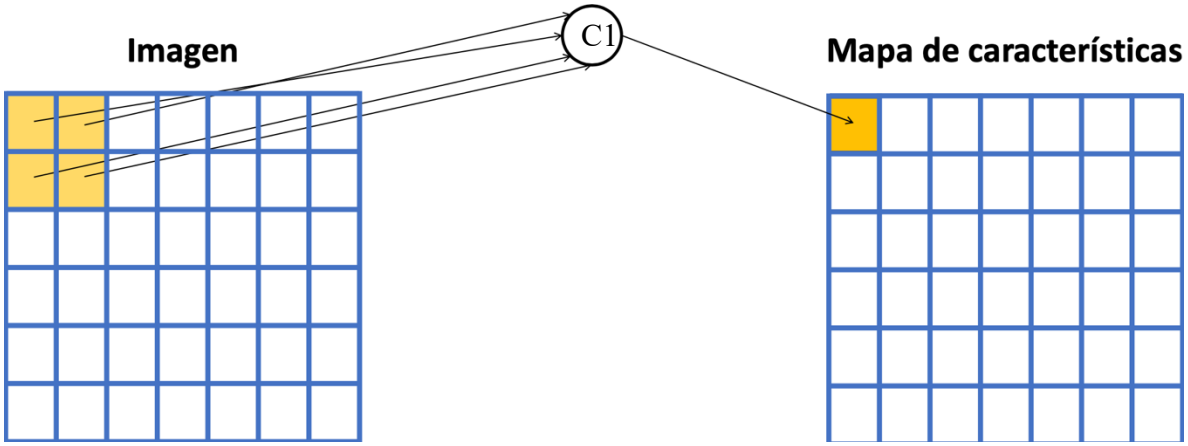


Figura 5.10. Representación matricial de una operación de convolución. En esta figura la convolución se realizar en el círculo "C1" tomando entradas de "Imagen" y sacando resultados a "Mapa de características"

La capa convolucional esta compuesta de distintos filtros, estos dependen principalmente del tipo de características deseadas a extraer con el filtro, por ejemplo, existen filtros para detectar patrones horizontales, verticales, esquinas, etc.; la salida (mapas de activaciones) es un volumen.

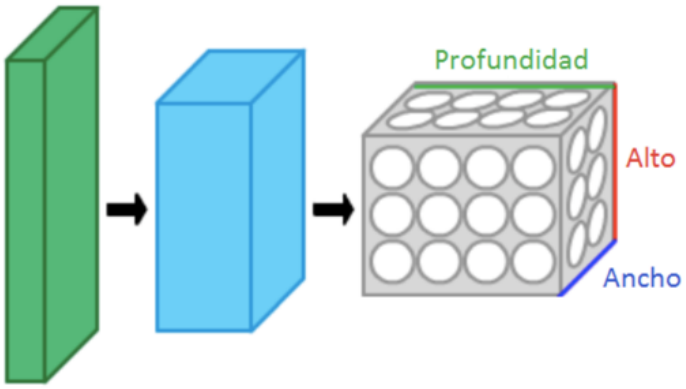


Figura 5.11. Representación Mapas de activaciones.

Existen distintos parámetros en esta capa:

- Número de filtros $N_{filtros}$: Cada filtro esta definido por un tamaño (F_H × F_W). Para el comportamiento de los filtros se deben definir los hiperparámetros: desplazamiento (stride) y relleno (padding). El desplazamiento es el número de pixeles que se mueven los filtros y el padding crea una capa externa de pixeles para no perder información en las esquinas de las imágenes.

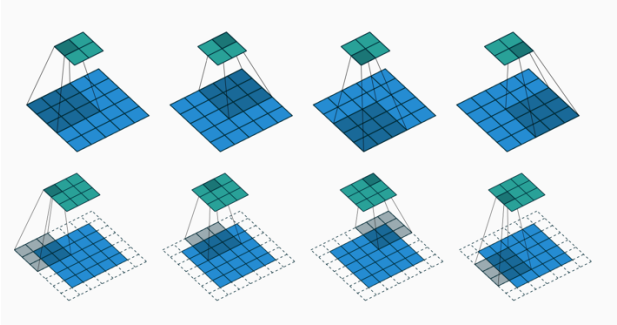


Figura 5.12. Representación grafica de filtros (verde), stride (avance) y padding (líneas punteadas) [82].

El alto H{l} y ancho W{l} de la salida de una capa de convolución están definidos por las ecuaciones 33 y 34:

$$H^{\{l\}} = \frac{H^{\{l-1\}} + 2P - F_H}{S} + 1 \qquad (33)$$

$$W^{\{l\}} = \frac{W^{\{l-1\}} + 2P - F_W}{S} + 1 \qquad (34)$$

Dónde:

- $H^{\{l\}}$ es la altura de la salida de la capa de convolución
- $H^{\{l-1\}}$ es la altura de la salida de la capa anterior
- P corresponde al hiperparámetro padding (relleno)
- F_H corresponde a la altura del filtro
- F_W corresponde al ancho del filtro
- S corresponde al hiperparámetro stride (paso)
- $W^{\{l\}}$ es el ancho de la salida de la capa de convolución
- $W^{\{l-1\}}$ es el ancho de la salida de la capa anterior

La siguiente capa después de la convolución es la no linealidad. La no linealidad se puede utilizar para ajustar o cortar la salida generada. Esta capa se aplica para saturar la salida o limitar la salida generada. Se pueden usar distintas funciones de activación como: Sigmoidea, ReLU, etc.

B. Relleno (Padding)

Uno de los inconvenientes del paso de convolución es la pérdida de información que podría existir en el borde de la imagen [66]. Debido a que solo se capturan cuando el filtro se desliza, nunca tienen la oportunidad de ser vistos. Un método muy simple pero eficiente para resolver el problema es usar relleno de ceros. El otro beneficio del relleno de ceros es administrar el tamaño de salida.

El relleno es un término que se refiere a la cantidad de píxeles agregados a una imagen cuando está siendo procesada por el filtro de una CNN. La idea del relleno ayuda a evitar que el tamaño de salida de la red se reduzca con la profundidad. Por lo tanto, es posible tener cualquier número de redes convolucionales profundas. [73].

C. Desplazamiento (Stride)

Una CNN tiene más opciones que brindan muchas oportunidades incluso para disminuir cada vez más los parámetros y, al mismo tiempo, reducir algunos de los efectos secundarios. Una de estas opciones es el desplazamiento. Podemos manipular el filtro que convoluciona sobra una imagen controlando el desplazamiento. Si movemos el filtro un nodo cada vez, la extracción de características ira revisando la imagen píxel por píxel, sin embargo, si utilizamos un paso mayor, el filtro se ira recorriendo de una forma (n), siendo n el tamaño del paso. Esto generaría un resultado con menor información, es decir, se reducirá el tamaño de la salida [71,73].

D.Decimación (Pooling)

Esta es una capa para reducir dimensiones tomando el valor máximo (max-pooling) o el promedio (average pooling) de un grupo de activaciones usualmente contiguas. La idea principal de la decimación es el muestreo descendente para reducir la complejidad de las capas adicionales. La decimación no afecta la cantidad de filtros. La decimación máxima (max pooling) es uno de los tipos más comunes de métodos de agrupación. Este divide la imagen en rectángulos de subregión y solo devuelve el valor máximo del interior de esa subregión. Debe tenerse en cuenta que el muestreo descendente no preserva la posición de la información. Además, la agrupación se puede utilizar con filtros y pasos no iguales para mejorar la eficiencia. Por ejemplo, una agrupación máxima de 3x3 con paso 2 mantiene algunas superposiciones entre las áreas. [72].

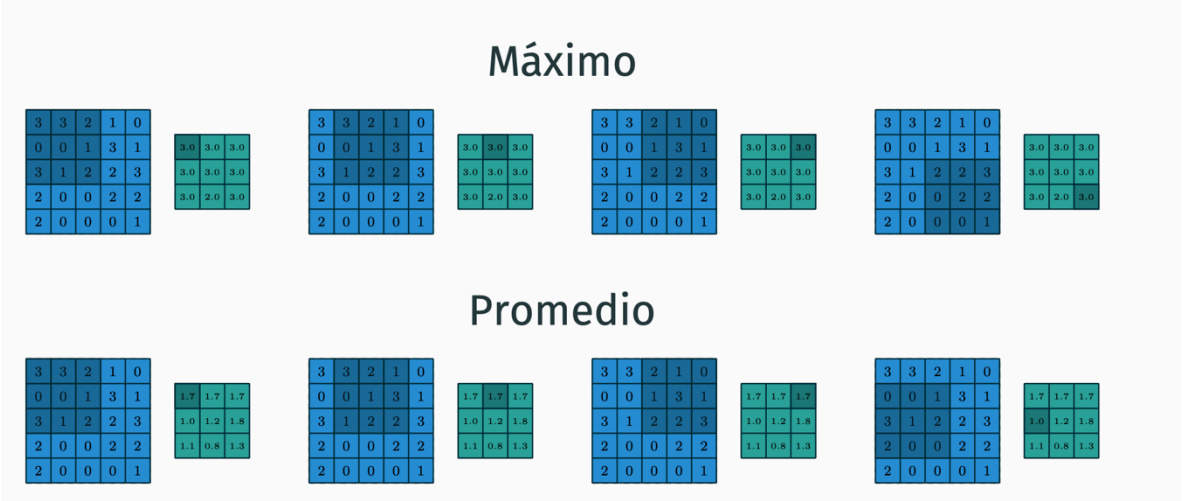


Figura 5.13. Representación grafica de pooling[82]. Se puede observar como toma distintos valores para la operación de máximo y promedio con los mismos filtros (sección obscurecida en las imágenes que se muestran con un color azul).

El alto H{l} y ancho W{l} de la salida de una capa de submuestreo está dado por las ecuaciones 35 y 36:

$$H^{\{l\}} = \frac{H^{\{l-1\}} + 2P - F_H}{S} + 1 \qquad (35)$$

$$W^{\{l\}} = \frac{W^{\{l-1\}} + 2P - F_W}{S} + 1 \qquad (36)$$

Dónde:

- $H^{\{l\}}$ es la altura de la salida de la capa de convolución
- $H^{\{l-1\}}$ es la altura de la salida de la capa anterior
- P corresponde al hiperparámetro padding (relleno)
- F_H corresponde a la altura del filtro
- F_W corresponde al ancho del filtro
- S corresponde al hiperparámetro stride (paso)
- $W^{\{l\}}$ es el ancho de la salida de la capa de convolución
- $W^{\{l-1\}}$ es el ancho de la salida de la capa anterior

E. Capa Completamente Conectada

La capa completamente conectada es similar a la forma en que las neuronas están organizadas en una red neuronal tradicional [66]. Por lo tanto, cada nodo en una capa completamente conectada está directamente conectado a cada nodo tanto en la capa anterior como en la siguiente.

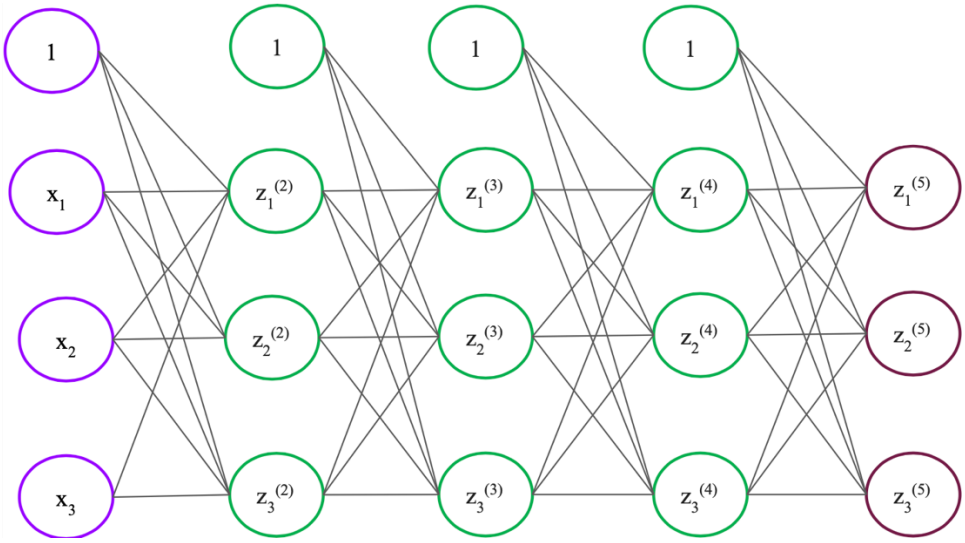


Figura 5.14. Capa Completamente Conectada

Por lo general la salida de la ultima capa de decimación esta conectada a las capas completamente conectadas. Esto genera los parámetros más utilizados con la CNN dentro de estas capas, y su formación lleva mucho tiempo [74,75]. El mayor inconveniente de una capa completamente conectada es que incluye muchos parámetros que necesitan cálculos complejos en los ejemplos de entrenamiento. Los nodos y la conexión eliminados pueden satisfacerse mediante la técnica de dropout (detallada a continuación). Por ejemplo, LeNet y AlexNet diseñaron una red amplia y profunda manteniendo constante el complejo computacional [76,77,78].

F. Dropout

Dropout es un algoritmo para entrenar redes neuronales que se describió en NIPS 2012 [80]. En su forma más simple, durante el entrenamiento, en cada presentación de ejemplo, los detectores de características se eliminan con probabilidad q = 1; p = 0.5 y los pesos restantes se entrenan por retropropagación [81]. Todos los pesos se comparten en todas las presentaciones de ejemplo. Durante la predicción, los pesos se dividen por dos. La principal motivación detrás del algoritmo es evitar la co-adaptación de los detectores de características, o el sobreajuste, al obligar a las neuronas a ser robustas y depender del comportamiento de la población, en lugar de la actividad de otras unidades específicas. La co-adaptación en la detección de características puede suceder cuando dos neuronas detectan patrones iguales en diferentes imágenes, es por esto que al apagar una, se elimina la co-adaptación generada.En el Dropout se activan y se desactivan neuronas en las capas ocultas, a través de una malla de máscaras, aquellas neuronas inactivas no contribuyen en ninguna operación para la salida. Es un mecanismo para evitar el sobreajuste [79]. Dropout tiene un hiperparámetro definido por el usuario el cual es la cantidad de nodos al ser apagados de manera aleatoria y se expresa en una probabilidad.

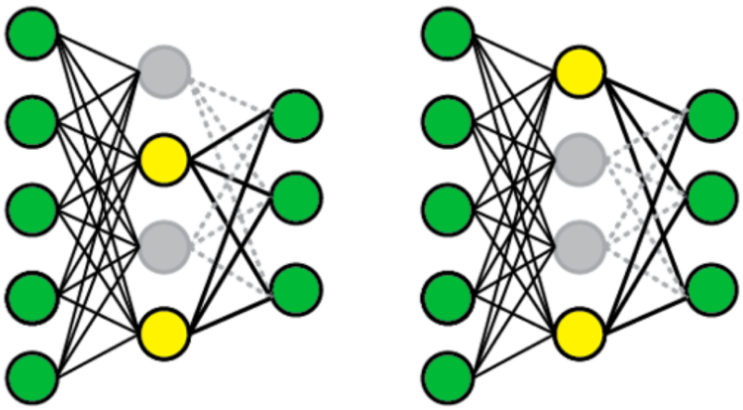


Figura 5.15. Esquema representativo del funcionamiento del Dropout, los nodos en gris son las neuronas inactivas y los nodos en amarrillo las neuronas activas.

5.2.2 Retropropagación en capas convolucionales y de submuestreo.

Para el caso de propagación hacia adelante:

- Se aplican las operaciones de convolución con correspondiente activación y decimación.

Para el caso de propagación hacia atrás:

- En la convolución cada neurona actualiza los gradientes por separado y al final se suman para actualizar los pesos compartidos.
- En la decimación se actualiza sólo la neurona ganadora (max-pooling).

5.3 Desarrollo de Modelos de Inteligencia Artificial

Se construyeron cuatro modelos de Inteligencia Artificial y se han comparado los resultados de todos los modelos. Las arquitecturas de redes neuronales construidas fueron las siguientes:

- Red Neuronal Perceptron Multicapa
- Red Neuronal Convolucional con 3 capas
- Red Neuronal Residual ResNet50
- Red Inception V3 Transferencia de Aprendizaje

5.3.1 Plataforma de desarrollo

Debido al gran tamaño del conjunto de datos “Lenguaje de señas mexicana” generado, los 4 modelos de inteligencia artificial construidos fueron desarrollados en la plataforma Google Colaboratory. Google Colaboratory (también conocido como Colab) es un servicio en la nube basado en Jupyter Notebooks [19] para difundir la educación y la investigación del aprendizaje automático. Proporciona un tiempo de ejecución completamente configurado para un modelo de aprendizaje profundo gracias a una GPU robusta. Google Colaboratory es descrito mas explícitamente en la sección 3.1.5.

Se aprovecho las características ofrecidas por el ambiente de ejecución de Colab para reducir drásticamente los tiempos de entrenamiento ya que se realizó todo en una GPU virtual ofrecida por Colab (TESLA P100 PCIe GPU ACCELERATOR [83]).

5.3.2 Lenguajes y tecnologías empleadas

5.3.2.1 Python

Python es un lenguaje de programación de alto nivel, interpretado, multiparadigma, dinámico y fuertemente tipado. Aunque Python fue conceptualizado a fines de la década de 1980 y después de su implementación en 1989, ha surgido como una nueva plataforma de lenguaje de múltiples paradigmas con la llegada de la Big Data. Debido al auge de la cantidad de datos públicos disponibles, en esta llamada era de la Big Data, considerando que el análisis de datos se ocupa principalmente del desarrollo de métodos computacionales para obtener información sobre los datos y obtener información, especialmente a través de herramientas de visualización [85] y el multiparadigma ofrecido por Python, este se ha convertido en la principal herramienta en la construcción de algoritmos de Inteligencia Artificial. Python incluye varias estructuras de datos, bibliotecas estándar con la implementación de análisis de sentimientos (proceso de identificar y categorizar computacionalmente las opiniones expresadas en un fragmento de texto) y ciencia de datos [84].

Python se ha vuelto popular por varias razones, incluyendo su simple sintaxis que es como un pseudocódigo; su modularidad; su diseño orientado a objetos; sus capacidades de creación de perfiles, portabilidad, pruebas y auto-documentación; y la presencia de una biblioteca numérica que permite el almacenamiento y manejo efectivo de enormes cantidades de información numérica [86].

5.3.2.2 Pytorch

Para el desarrollo de este sistema se aprovecharon las cualidades de PyTorch [87], una biblioteca de aprendizaje automático bien establecida para Python. Este es un paquete informático científico basado en Python que utiliza el poder de las unidades de procesamiento de gráficos. También es una de las plataformas de investigación de aprendizaje profundo preferidas, construida para proporcionar la máxima flexibilidad y velocidad. Es conocido por proporcionar dos de las funciones de más alto nivel; a saber, cálculos de tensores con un fuerte soporte de aceleración de GPU y construcción de redes neuronales profundas.

PyTorch rastrea todas las operaciones realizadas durante la simulación, lo que permite técnicas de optimización del aprendizaje automático bien establecidas basadas en la propagación hacia atrás [88].

PyTorch, a diferencia de los ndarrays de Numpy más utilizados [89], rastrean los gradientes de cada operación realizada en ellos. Esto crea un gráfico de cálculo dinámico, que permite el uso de retropropagación.

Se puede encontrar más información del uso de Pytorch en la sección 3.2.2.

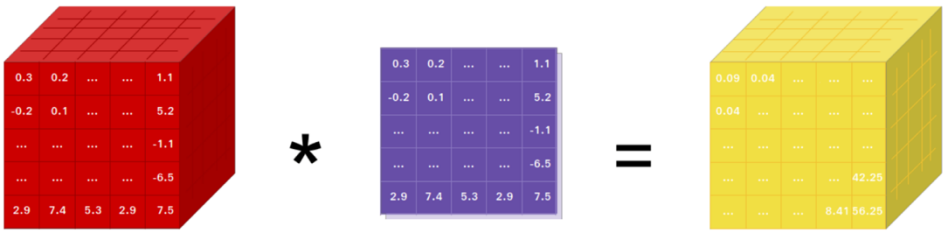


Figura 5.16. PyTorch proporciona tensores (amarillo) que pueden vivir en la CPU o en la GPU y acelera enormemente el cálculo.

5.3.3 Arquitecturas de Redes Neuronales Implementadas

5.3.3.1 Red Neuronal Perceptron Multicapa

El primer modelo construido esta constituido por las siguientes capas:

- Capa Totalmente Conectada con una función activación ReLU.
- Capa Totalmente Conectada con una función activación ReLU.
- Capa Totalmente Conectada con una función activación sigmoidea.
- Capa Softmax

Gracias a la capa Softmax, podemos generalizar para nuestro conjunto de datos del lenguaje de señas mexicano ya que una capa final con una función de activación sigmoidea solo generalizaría para dos clases (0,1).

Se ha implementación una función para inicializar los parámetros utilizando una inicialización Xavier para los pesos y una inicialización de ceros para los umbrales. Finalmente se ha utilizado un optimizador ADAM para la actualización de los pesos y de los umbrales a través de la retropropagación con una tasa de aprendizaje de $1e^{-3}$. El modelo se entreno durante 300 épocas.

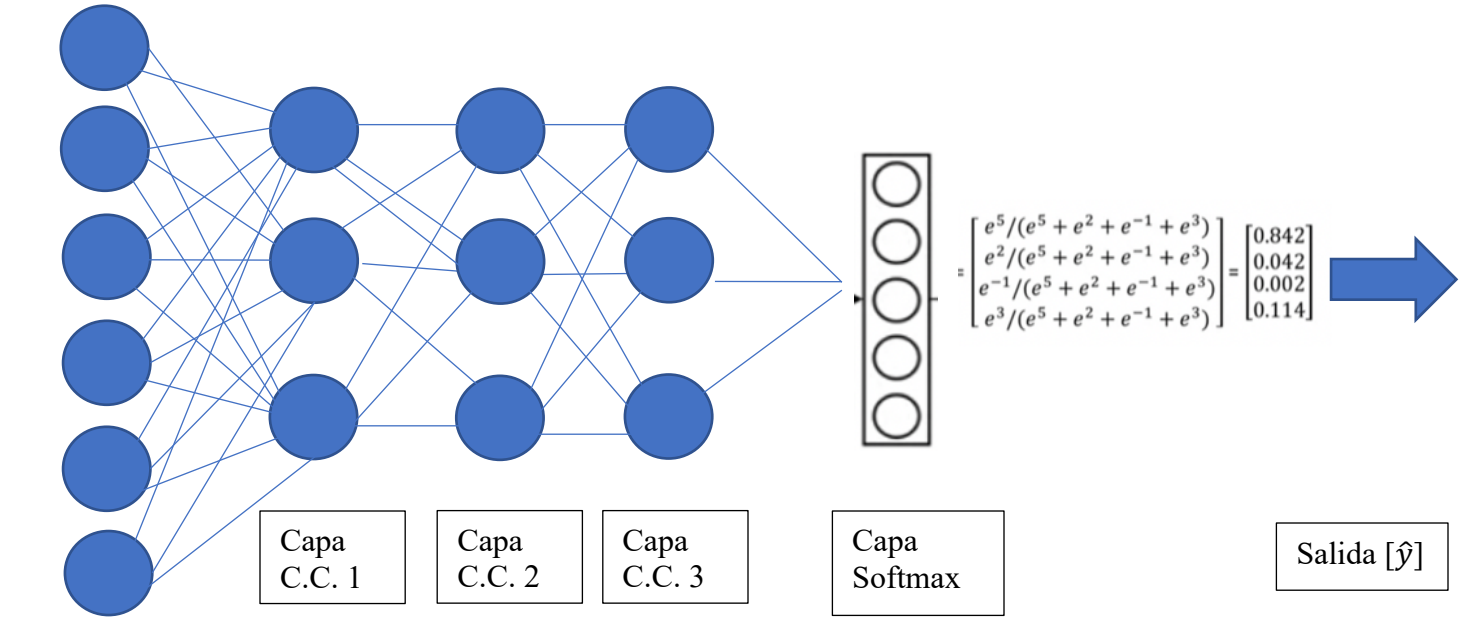


Figura 5.17. Esquema modelo Red Neuronal Perceptrón Multicapa

5.3.3.2 Red Neuronal Convolucional con 3 capas

El segundo modelo construido esta constituido por las siguientes capas:

- Capa Convolucional 2D con una función activación ReLU y un muestreo max pooling.
- Capa Convolucional 2D con una función activación ReLU y un muestreo max pooling.
- Capa aplanadora
- Capa Totalmente Conectada con una función activación sigmoidea.
- Capa Softmax

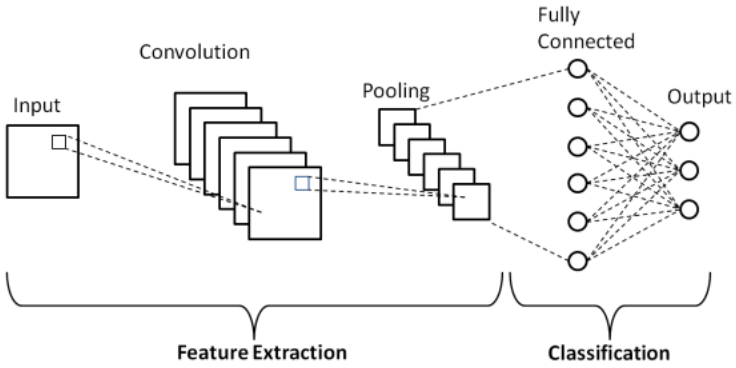


Figura 5.18. Esquema Red Neuronal Convolucional con una capa convolucional y una capa totalmente conectada

Se ha implementación una función para inicializar los parámetros utilizando una inicialización Xavier para los pesos y una inicialización de ceros para los umbrales.
Se ha utilizado una función de costo función entropía cruzada categórica con logits. Estos son el resultado de multiplicar los pesos y agregar los umbrales. Los logits se pasan a través de una función de activación (como un ReLU), y el resultado se llama "activación".

Pytorch ofrece una implementación de la función entropía cruzada categórica con una capa softmax llamada “softmax_cross_entropy_with_logits”. Esta toma los logits como entrada y se utiliza el modelo para predecir usando softmax, haciendo una comparación de las predicciones con las etiquetas verdaderas usando la función de entropía cruzada. Estos se realizan con una sola función para optimizar los cálculos. Finalmente se ha utilizado un optimizador ADAM para la actualización de los pesos y de los umbrales a través de la retropropagación con una tasa de aprendizaje de $1e^{-3}$. El modelo se entrenó durante 100 épocas.

5.3.3.3 Red Neuronal Residual ResNet50

Las redes neuronales más profundas son más difíciles de entrenar. A lo largo de los años ha existido una tendencia a profundizar más, a resolver tareas más complejas y también a aumentar / mejorar la precisión de clasificación / reconocimiento. Pero, a medida que se profundiza; el entrenamiento de la red neuronal se vuelve difícil y también la precisión comienza a saturarse y luego también se degrada. El aprendizaje residual intenta resolver ambos problemas. ResNet-50 es una red neuronal convolucional que se entrena en más de un millón de imágenes de la base de datos ImageNet. La red tiene 50 capas de profundidad y puede clasificar imágenes en 1000 categorías de objetos, como teclado, ratón, lápiz y muchos animales. Como resultado, la red ha aprendido representaciones de características ricas para una amplia gama de imágenes. La red tiene un tamaño de entrada de imagen de 224 por 224 [102].

Las redes neuronales convolucionales profundas, apilan varias capas y se entrenan para la tarea en cuestión. La red aprende varias características de nivel bajo / medio / alto al final de sus capas. Sin embargo, cuando se trata de aprendizaje residual, en lugar de intentar aprender algunas características, se trata de aprender algo residual.

El aprendizaje residual se puede entender como la diferencia de la característica aprendida de la entrada de esa capa. ResNet hace esto usando conexiones de acceso directo (conectando directamente la entrada de la enésima capa a alguna (n + x) capa). Se ha demostrado que entrenar esta forma de redes es más fácil que entrenar redes neuronales convolucionales profundas simples, pero se resuelve el problema de la degradación de la precisión. ResNet reformula los mapeos que se desean aprender a residuales lo que hace que sean más fácil de optimizar las características residuales que el mapeo original (se ha mostrado que son aproximadores universales) [103]. La ResNet logra implementar aprendizaje residual mediante la implementación del bloque residual:

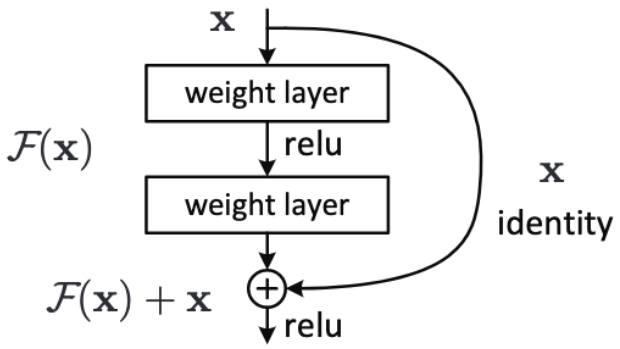


Figura 5.19. Esquema del bloque residual para el aprendizaje residual[102].

En el bloque residual básico se pasan las entradas del bloque (x) por dos capas convolucionales F (x). Posteriormente se suma de nuevo la entrada F (x) + x al final del bloque, la formulación de F (x) + x se puede realizar mediante redes neuronales de retroalimentación con "conexiones de acceso directo". Las conexiones de acceso directo [104] son las que se saltan una o más capas. Finalmente se pasa la salida de la suma por la activación ReLU [102].Si x es la entrada y F (x) es la salida de la capa, entonces la salida del bloque residual como se define en la ecuación 37:

$$Y = F(x) + x \qquad (37)$$

Ésta es la definición más básica de un bloque residual. Sin embargo, puede haber algunos escenarios en los que la salida de la capa y la entrada de identidad tengan diferentes dimensiones. Por ejemplo, si se considera una CNN donde se sabe que después de la operación de convolución, el tamaño de la entrada se reduce (dimensionalmente), entonces agregarle una entrada es un problema. Entonces, lo que aquí se puede hacer es que, en la conexión de acceso directo, se le agregue alguna operación o función de modo que la entrada se cambie o se configure a las dimensiones requeridas. De esta manera, la definición se puede actualizar aquí como se define en la ecuación 38:

$$F(x, \{Wi\}) + Ws * x \qquad (38)$$

Actualmente hay dos arquitecturas de bloques residuales, una básica y una de con filtros más profundos (bottleneck)

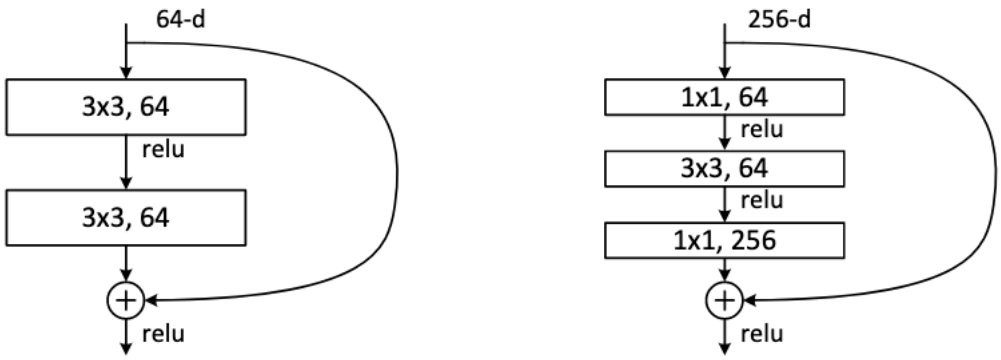


Figura 5.20: Arquitecturas de bloques residuales [102].

Las arquitecturas de cuello de botella más profundas establecen que para cada función residual F, usamos una pila de 3 capas en lugar de 2. Las tres capas son convoluciones 1 × 1, 3 × 3 y 1 × 1, donde las capas 1 × 1 son responsables de reducir y luego aumentar (restaurar) las dimensiones, dejando la capa 3 × 3 como un cuello de botella con dimensiones de entrada / salidas más pequeñas [102]. La figura 5.20 muestra un ejemplo, donde ambos diseños tienen una complejidad de tiempo similar.

El tercer modelo construido esta constituido por las siguientes capas definidas en la Figura 5.21 para la ResNet de 50 capas:

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figura 5.21. Arquitectura de distintas ResNet con diferentes números de capas

Es importante mencionar que para el tercer modelo construido no se aplico aprendizaje por transferencia, solo se tomo la arquitectura de red neuronal artificial ResNet 50 sin ningún tipo de aprendizaje en sus pesos. Se han utilizado las mismas inicializaciones y función de costo que para el segundo modelo construido. Finalmente se ha utilizado un optimizador ADAM para la actualización de los pesos y de los umbrales a través de la retropropagación con una tasa de aprendizaje de $1e^{-3}$ para la actualización de los pesos y de los umbrales a través de la retropropagación. El modelo se entreno durante 40 épocas.

5.3.3.4 Red Inception V3 Transferencia de Aprendizaje

La idea para la arquitectura Inception nace del de la [105] participación ganadora de Krizhevsky [104] en la competencia ImageNet en 2012. Esto estimuló una nueva línea de investigación que se centró en encontrar redes neuronales convolucionales de mayor rendimiento. La arquitectura Inception de GoogLeNet [106] es diseñada para funcionar bien incluso bajo estrictas restricciones de memoria y presupuesto computacional. Esta solo empleó sólo 5 millones de parámetros, lo que representó una reducción de 12 × con respecto a sus predecesores, AlexNet, que utilizó 60 millones de parámetros [107]. El costo computacional de Inception también es mucho menor que VGGNet o sus sucesores de mayor rendimiento [108]. Esto ha hecho factible utilizar redes Inception en escenarios de datos grandes [109], donde se necesita procesar una gran cantidad de datos a un costo razonable o escenarios donde la memoria o la capacidad computacional es inherentemente limitada, por ejemplo, en configuraciones de visión móvil [107]. Las principales ventajas que la arquitectura Inception ofrece son:

- Evita los cuellos de botella representativos, especialmente al principio de la red.
- Las representaciones de mayor dimensión logran ser más fáciles de procesar localmente dentro de una red.
- La agregación espacial se puede hacer sobre incrustaciones de dimensiones inferiores sin mucha o ninguna pérdida en el poder de representación.
- Se equilibra el ancho y la profundidad de la red. Se puede alcanzar un rendimiento óptimo de la red equilibrando el número de filtros por etapa y la profundidad de la red.

Gran parte de los beneficios originales de la red GoogLeNet [106] surgen de un uso muy generoso de la reducción de dimensiones. Esto puede verse como un caso especial de factorizar las convoluciones de una manera computacionalmente eficiente. Dado que las redes de Inception son completamente convolucionales, cada peso corresponde a una multiplicación por activación. Por lo tanto, cualquier reducción en el costo computacional da como resultado un número reducido de parámetros. Esto significa que, con una factorización adecuada, es posible terminar con parámetros más desenredados y, por lo tanto, con un entrenamiento más rápido.

Las convoluciones con filtros espaciales más grandes (por ejemplo, 5 × 5 o 7 × 7) tienden a ser desproporcionadamente caras en términos de cálculo. [107]. Estos pueden ser reemplazados por una red multicapa con menos parámetros con el mismo tamaño de entrada y profundidad de salida.

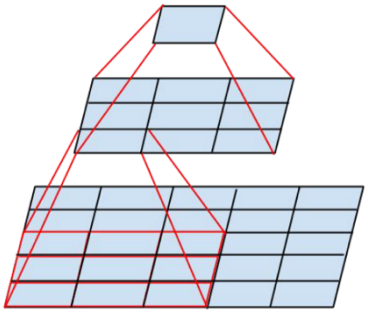


Figura 5.22. Mini-red que reemplaza las convoluciones grandes (5 × 5; 7X7) [106].

La arquitectura Inception introduce los bloques inception, estos constan de convoluciones de 1x1 seguidas de convoluciones de 3x3, esta capa cuenta con convoluciones de 1x1 seguidas de convoluciones de 5x5 seguido de t una capa de max pooling de 3x3 seguida de convoluciones de 1x1 con una convolución única de 1x1. La idea es que, si usamos todas las convoluciones y el pooling en un bloque, algunas de ellas serán lo

suficientemente eficientes como para extraer información significativa de las imágenes. Para asegurarse de que se mantengan las dimensiones de la imagen, las convoluciones de 3x3 tienen un relleno de 1 y la capa de 5x5 tiene un relleno de 2 para que las imágenes de entrada y salida tengan el mismo tamaño. Y finalmente, todos están apilados juntos. Al deslizar esta pequeña red sobre la cuadrícula de activación de entrada se reduce a reemplazar la convolución 5 × 5 con dos capas de convolución 3 × 3. Esta configuración reduce claramente el número de parámetros al compartir los pesos entre zonas adyacentes.

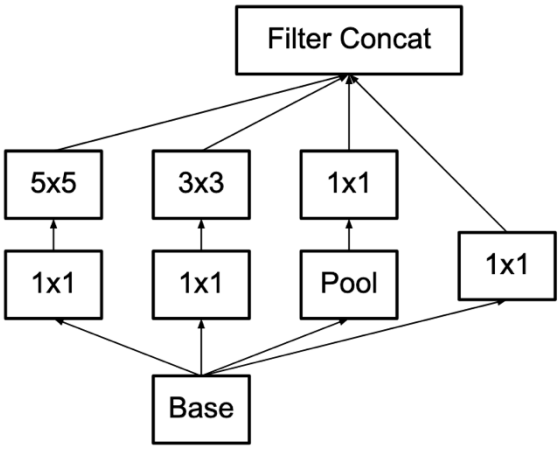


Figura 5.23. Bloque Inception[106].

El número de filtros en cada capa en los bloques Inception está diseñado de tal manera que se obtenga el número deseado de canales como salida para el siguiente bloque. La arquitectura de la red Inception esta dividida en 5 etapas, estas se definen de la siguiente manera:

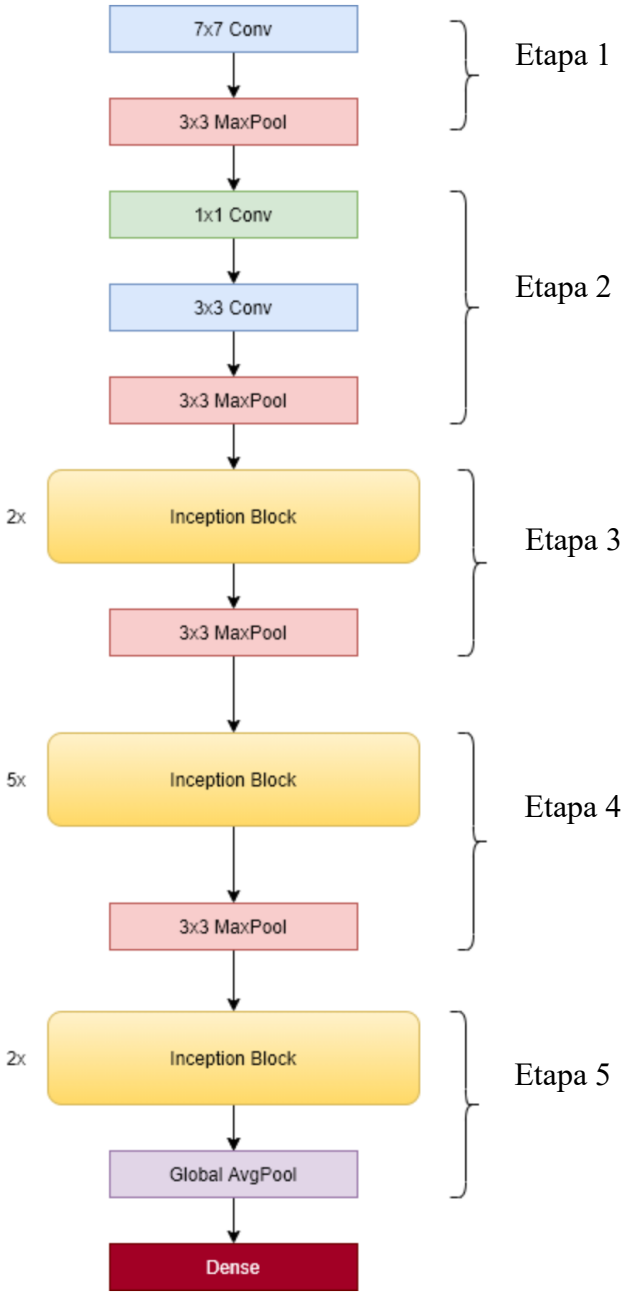


Figura 5.24. Arquitectura Red Inception

- Etapa 1 y 2

  La red comienza con un tamaño de imagen de 224x224x3. Luego pasa por 1x1 Conv, 3x3 MaxPool, 1x1 Conv, 3x3 Conv y 3x3 MaxPool, y da como resultado una imagen de tamaño 192x28x28.

- Etapa 3

  La etapa 3 tiene dos bloques Inception y al final una capa Max Pool. Pero los bloques de inicio no tienen la misma asignación de canales, como se ve en la figura. El bloque 1 tiene 256 canales, mientras que el bloque 2 tiene 480 canales. Entonces, la imagen de entrada de 192x28x28 ahora se convierte en 480x14x14, después de los dos bloques de inicio y la capa MaxPooling.

- Etapa 4 y 5

  La etapa 4 y 5 son muy similares a la etapa 3. La etapa 4 tiene 5 bloques de inicio seguidos de un MaxPool, y la etapa 5 tiene 2 bloques de inicio seguidos de un GlobalAveragePool. Finalmente, la Etapa 5 se conecta a una capa totalmente conectada.

Así que esta es la arquitectura Inception o GoogleLeNet que se publicó originalmente. Más tarde, la arquitectura se ha mejorado aún más en varias versiones diferentes v2, v3 y v4. Para Inception V3,que es la versión que se utiliza en este sistema, se hicieron los siguientes cambios con el fin de mejorar la extracción de características:

- Se ha reemplazado conv. 5x5 con múltiples convoluciones 3x3.
- Se ha reemplazado conv. 5x5 con convoluciones 1x7 y 7x1.
- Se ha reemplazado conv. 3x3 con convoluciones 1x3 y 3x1.

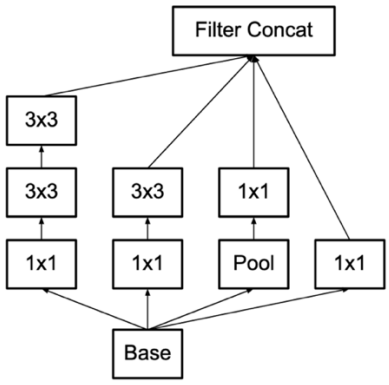


Figura 5.25. Bloque Inception V3[106].

Es importante mencionar que para el cuarto modelo construido se aplico aprendizaje por transferencia. Se ha utilizado los pesos iniciales de la arquitectura Inception V3 entrenada con ImageNet, se utilizo la misma función de costo que para los modelos anteriores. Finalmente se ha utilizado un optimizador ADAM para la actualización de los pesos en las ultimas capas y de los umbrales a través de la retropropagación con una tasa de aprendizaje de $1e^{-3}$ para la actualización de los pesos y de los umbrales a través de la retropropagación. El modelo se entreno durante 20 épocas.

5.3.3.4.1 Aprendizaje por Transferencia

La arquitectura Inception utilizada en este proyecto es una arquitectura de vanguardia que se utiliza por su capacidad para generalizar el conocimiento aprendido a conjuntos de datos distintos de aquellos en los que se entrenaron originalmente a través del aprendizaje por transferencia. El objetivo principal del aprendizaje por transferencia es utilizar las redes neuronales artificiales en los problemas de clasificación de imágenes, aunque es posible que no se disponga de grandes conjuntos de entrenamiento para estas imágenes [112]. Este es el principal motivo de utilizar esta técnica, el tamaño del conjunto de datos del lenguaje de señas mexicano generado.

Distintos investigadores han demostrado que utilizar el aprendizaje por transferencia, incluso en los casos en los que los conjuntos de datos base y objetivo son diferentes, es mejor que inicializar aleatoriamente las ponderaciones de la red neuronal [111]. La capacidad de transferir el conocimiento de una red neuronal que se entrenó en los datos de ImageNet se ha demostrado en diferentes campos, por ejemplo, en imágenes médicas y en imágenes de seguridad por rayos X [110].

5.3.4 Entrenamiento de la red.

Para el entrenamiento de los modelos se implemento una función titulada *train*; del cual se presenta el siguiente pseudocódigo:

1. Migración de todos los cálculos a la GPU ofrecida por Colab.
2. Definición de función: *train_epoch.* Usada para el entrenamiento de una época:
   2.1 Calculó los logits(propagación hacia adelante).
   2.2 Calculó de la predicción $\hat{y}$.
   2.3 Calculó del valor de la función perdida.
   2.4 Vaciado de gradiente.
   2.5 Retropropagación.
   2.6 Actualización de los parámetros de aprendizaje.
3. Definición de función: *save_checkpoint.* Implementada para guardar un punto de control (se guarda el archivo .pth; el cual contiene los valores vectoriales actuales de los pesos y umbrales).
4. Definición de función: *eval_*epoch. Usada para evaluar una época:
   4.1 Inferencia para obtener los logits.
   4.2 Computo de probabilidades (capa softmax).
   4.3 Obtención de la clase predicha (se toma la probabilidad mayor de la capa softmax).
   4.4 Cómputo de los valores de pérdida y de exactitud.
5. Inicialización del optimizador.
6. Para transferencia de conocimiento: congelación de los valores de los pesos y de las estadísticas σ y μ.
7. Definición de un ciclo con el número de épocas:
   7.1 Entrenamiento de una época: *train_epoch*
   7.2 Evaluación de la época en entrenamiento: *eval_epoch*
   7.3 Registro de valores de perdida y exactitud para el mini-lote correspondiente al conjunto de entrenamiento.
   7.4 Evaluación de la época en validación.

7.5 Registro de valores de perdida y exactitud para el mini-lote correspondiente al conjunto de validación.

8. Si existe una mejoría en las métricas del modelo a comparación con la época anterior se procede a guardar punto de control.

5.3.4.1 Reentrenamiento de Modelo

Para el modelo que se diseñó para utilizar transferencia de conocimiento; cuando se importa el modelo previamente entrenado y se genera el nuevo conjunto de datos, comienza el entrenamiento del nuevo modelo. Esto se hace agregando capas adicionales. Las primeras capas del modelo pre-entrenado se congelan (los pesos del modelo se volvieron inmutables) para no volver a entrenarlos pero mantener su conocimiento. Se agrego una capa completamente conectada con una salida de 21, ya que estas son el número total de etiquetas. Se elige a Adam porque en general converge rápidamente y es adecuado para problemas con una gran cantidad de parámetros y tuvo resultados superiores al descenso por gradiente estocástico con mini lotes.

El modelo final, al tratarse de una CNN, consta de 21 828 597 parámetros.Se entrena por 20 epocas.

Debido a los problemas generados por las variaciones intraclase e interclase, se decide realizar todas las pruebas de precisión finales en un conjunto de datos de prueba que no haya sido utilizado en el entrenamiento del modelo. Se corre con el mismo modelo y tomando las 10500 imágenes como datos de entrenamiento.

5.3.5 Métricas Evaluación de Modelos

Considerando la naturaleza del problema que ataca este sistema, se ha utilizado la métrica precisión como la principal métrica para medir el desempeño de los modelos. La precisión es una métrica para evaluar modelos de clasificación. De manera informal, la precisión es la fracción de predicciones que nuestro modelo acertó:

$$precisión = \frac{\#\ predicciones\ correctas}{\#\ total\ de\ predicciones} \qquad (39)$$

Se ha utilizado esta métrica ya que, al ser un clasificador multiclase, es importante comparar al número totales de predicciones con el número de predicción correcta de todas las clases. Además, al ser un clasificador, se esta considerando como predicción incorrecta a cualquier predicción que no concuerde con el valor verdadero (etiqueta) de cada imagen.

5.3.6 Resultados

5.3.6.1 Red Neuronal Perceptron Multicapa

El primer modelo presento los siguientes resultados:

Precisión Entrenamiento	99.90%
Precisión Validación	97.34%
Precisión Prueba	71.66%

Figura 5.26. Comparación resultados para los conjuntos de datos de entrenamiento, validación y prueba

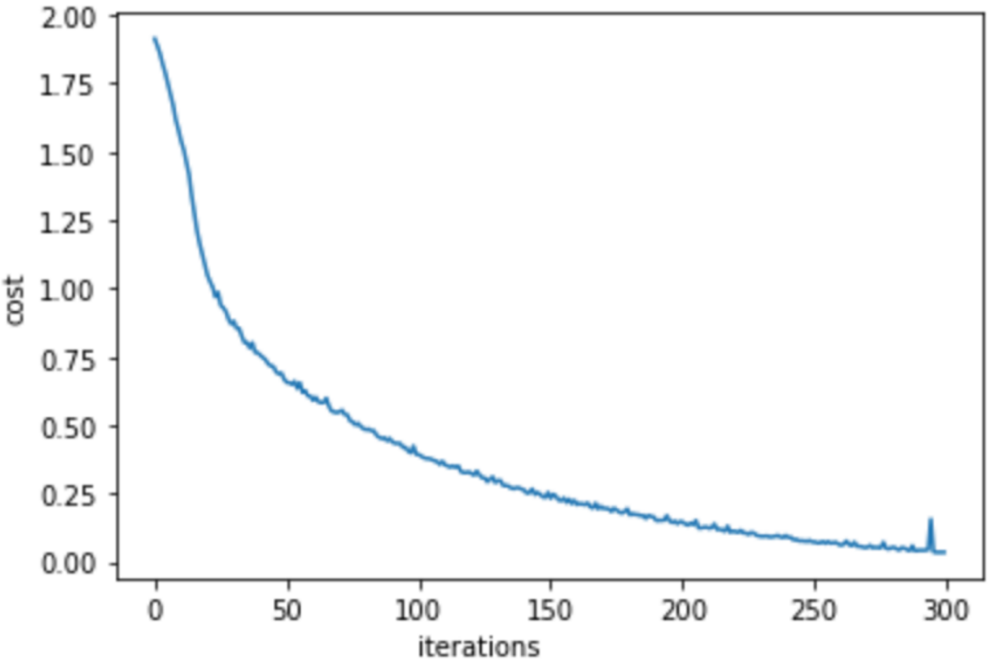


Figura 5.27. Grafica con valores de la función de costo para el primer modelo

5.3.6.2 Red Neuronal Convolucional con 3 capas

El primer modelo convolucional presentó los siguientes resultados:

Precisión Entrenamiento	94.07%
Precisión Validación	92.52%
Precisión Prueba	78.33%

Figura 5.28. Comparación resultados para los conjuntos de datos de entrenamiento, validación y prueba

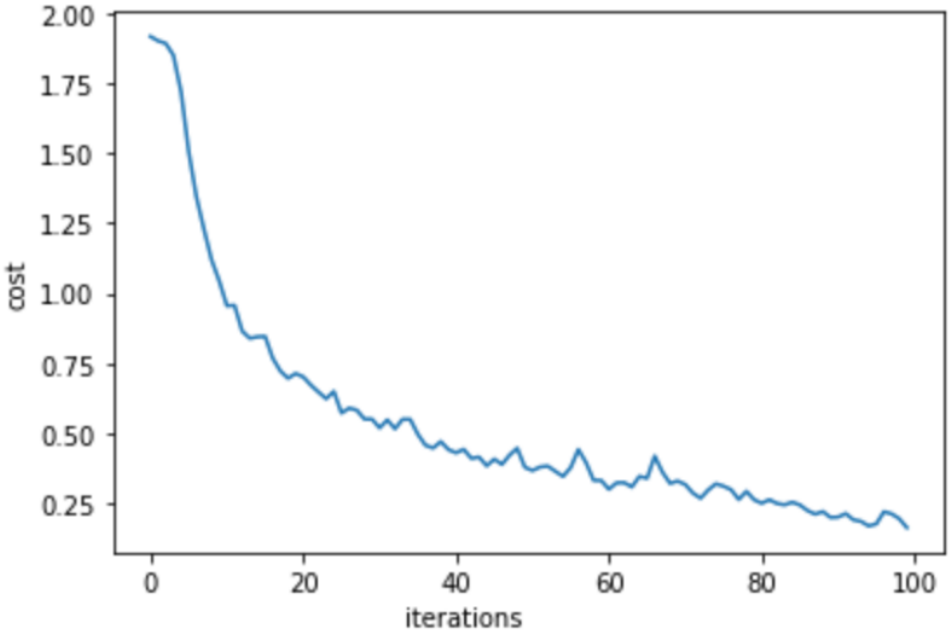


Figura 5.29 . Grafica con valores de la función de costo para el primer modelo convolucional

5.3.6.3 Red Neuronal Residual ResNet50

La red neuronal residual con 50 capas presento los siguientes resultados:

Precisión Entrenamiento	97.17%
Precisión Validación	93.92%
Precisión Prueba	86.66%

Figura 5.30. Comparación resultados para los conjuntos de datos de entrenamiento, validación y prueba

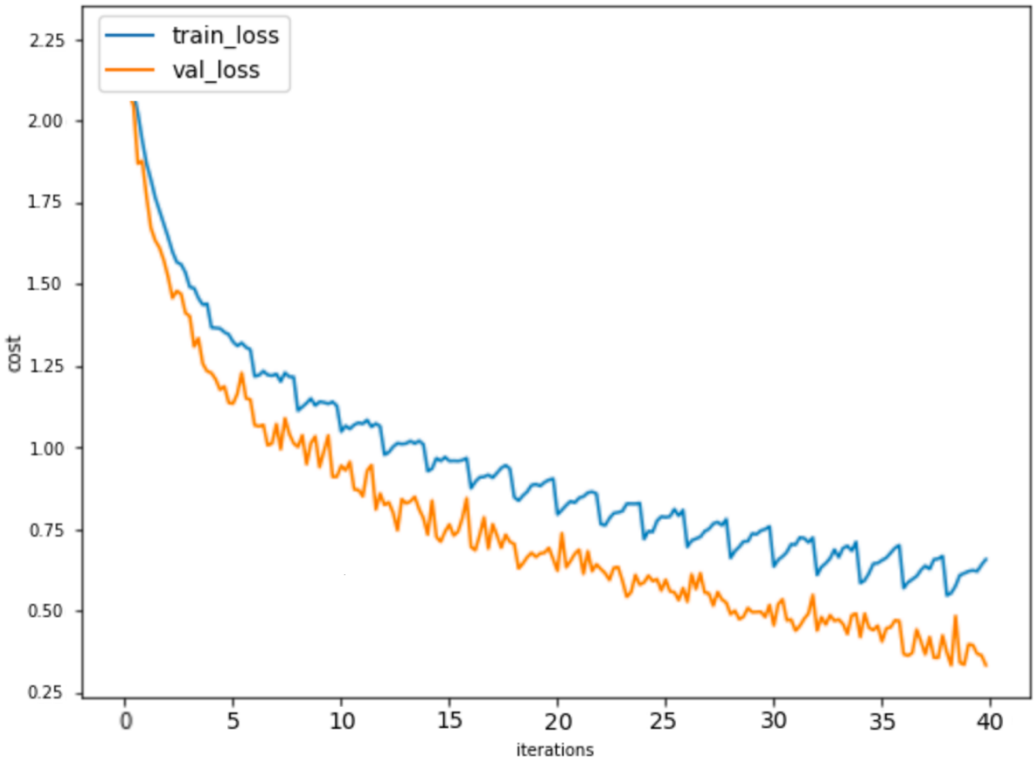


Figura 5.31. Grafica con valores de la función de costo para los conjuntos de entrenamiento y validación.

5.3.6.4 Red Inception V3

La red neuronal diseñada por Google[106] en combinación del aprendizaje por transferencia logró tener el más alto índice de precisión para el conjunto de datos de prueba:

Precisión Entrenamiento	99.99%
Precisión Validación	99.56%
Precisión Prueba	91.56%

Figura 5.32. Comparación resultados para los conjuntos de datos de entrenamiento, validación y prueba.

Es por esto que este modelo fue escogido para la exportación de los valores vectoriales de los pesos y los umbrales para la API de la aplicación.

Figura 5.33. Grafica con valores de la función de costo para los conjuntos de entrenamiento y validación visualizados en Tensorboard

En este modelo también se hizo la comparación de utilizar el conjunto de datos con y sin pos-procesamiento, estos fueron los resultados:

	Datos Sin Pos procesamiento	Datos con pos-procesamiento
Validación Cruzada de Datos	99%	99%
Datos de prueba	50%	91%

Figura 5.34. Resultados de los Modelos con Datos con y sin Pos procesamiento.

5.4 API

Se ha utilizado el framework Flask de Python debido a que es bien intencionado [3] y se adecua de una manera eficiente al marco de trabajo MVC. Se puede encontrar más información sobre Flask en la Sección 3.2.1 de este documento.

5.4.1 Arquitectura del sistema

La arquitectura del sistema esta estructurada de la siguiente forma:

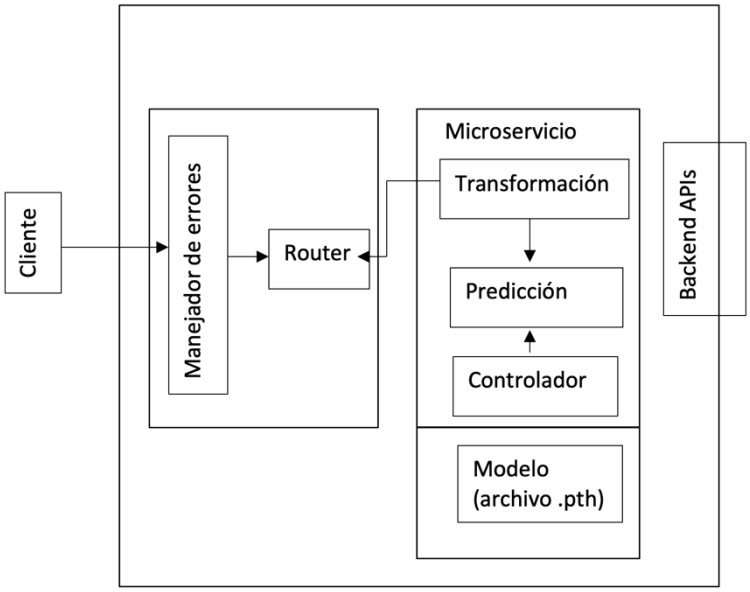


Figura 5.35. Arquitectura del sistema

- Modelo

  En esta sección del proyecto se han guardo los archivos con formato .pth provenientes de Colab. Estos archivos contienen el "aprendizaje" generado por las redes neuronales artificiales.

- Controlador

  Esta sección se divide en:

  1. Controlador (*controllers.py*)

  Se ha generado el archivo *index_to_name.json;* este contiene todo el mapeo para traducir el índice numérico del carácter al valor real del carácter predicho.

Dentro del Controlador se genera la función *render_prediction* esta es la responsable de tomar la salida del modelo de inteligencia artificial, hacer el mapeo con el archivo *index_to_name.json* y regresar la etiqueta de la clase.

2. Predicción (*prediction.py*)

   En este archivo de Python se carga el modelo exportado por Colab, importar el modelo InceptionV3, cargar los pesos exportados en el nuevo modelo y realizar el remplazo de la última capa. Finalmente se genera la función *get_prediction*; la cual se encarga de realizar propagación hacia adelante para generar los logits y generar la predicción.

3. Transformación (*tranform.py*)

   En este archivo se define la función *transform_image,* la cual se encarga de recibir la imagen del frontend y recortar un área cuadrada alrededor de la seña, esto con el objetivo de lograr una similitud entre las imágenes recibidas de la interfaz grafica al conjunto de datos generados pos procesado. Finalmente convierte esta imagen a un tensor.

- Endpoints
  1. Root (*root.py*)

     En esta archivo se define la ruta principal de la API("/"); la cual , atraves de una operación HTTP GET, regresa el siguiente json:

     {'msg': 'Try POSTing to the /predict endpoint with an RGB image attachment'}

2. Predicción(*prediction.py*)

   En esta sección del proyecto se define la ruta (“/predict”), la cual, a través de una función HTTP POST, manda llamar la función *predict;* la cual funciona con el siguiente pseudocodigo:

   2.1 Se importa todas las funciones descritas en la sección de controlador.

   2.2 Se crea una estructura condicional en la cual, si no detecta que la solicitud POST contiene un archivo adjunto, detiene la ejecución y en caso de si tener un archivo continua con la ejecución.

   2.3 Con al función *transform_image* genera el tensor de entrada a la red.

   2.4 Con la función get*_prediction* genera la inferencia y hace la predicción del tensor recibido.

   2.5 Con la función render*_prediction* genera la etiqueta de la predicción.

   2.6 Regresa un json con los valores de id y de etiqueta predicha:

   {'class_id': class_id, 'class_name': class_name}

5.5 Interfaz Gráfica

Para la implementación de la Interfaz Grafica se utilizar el framework de JavaScript AngularJS. Se utiliza este framework ya que el objetivo es que la interfaz grafica sea una SPA. En SPA, todos los componentes como CSS, imágenes, scripts y cualquier otro recurso requerido se cargan al mismo tiempo en la carga inicial de la página y luego el contenido / componentes apropiados se cargan dinámicamente dependiendo de la interacción del usuario. Se puede encontrar más información de las SPA en la sección 3.1.7.

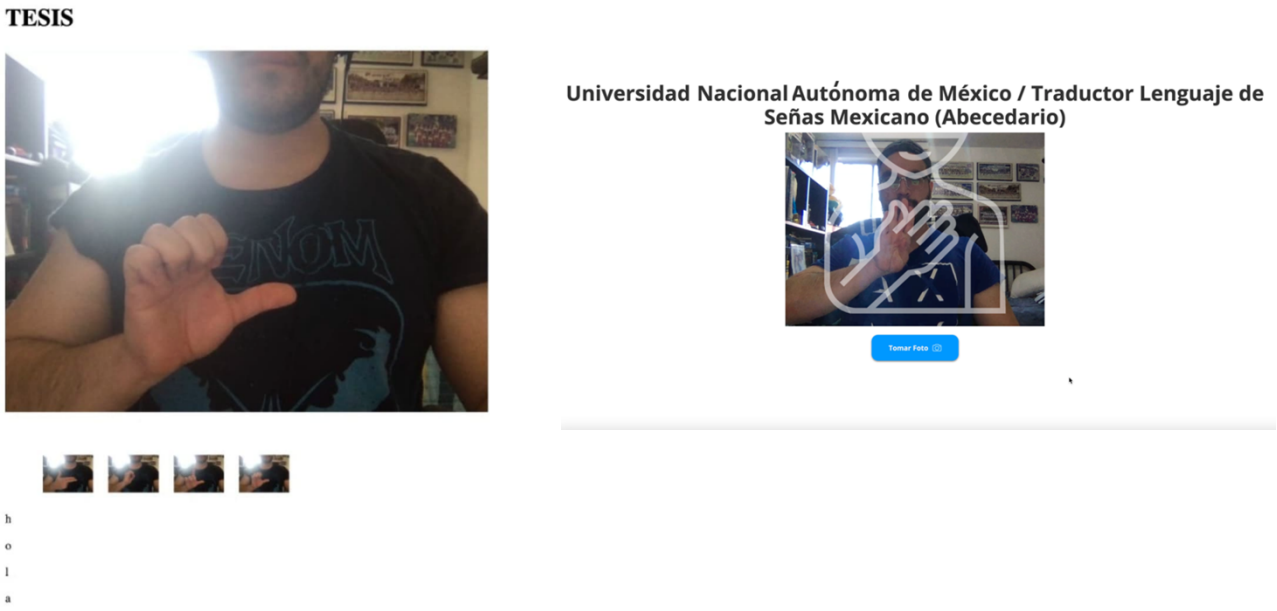


Figura 5.36. Diseños Iniciales Interfaz Grafica

5.5.1 Arquitectura del sistema

Al ser una SPA, la arquitectura del sistema esta basada en un simple archivo base del proyecto HTML llamado *index.html*. Angular JS lograr una modularidad en las SPA, de tal forma en que solo es necesario generar componentes que serán renderizados en este archivo, todos los componentes estas integrados por los siguientes elementos:

- Componente.componentes.css: encargado de importar todos los estilos de css para las distintas clases de los elementos generados por el autor
- Componente.componentes.html: este archivo carga con todos los - elementos html que se incluyen el archivo base del proyecto
- Componente.module.ts: encargado de importar todos los paquetes y librerías requeridas.

- Componente.component.ts: es en este archivo donde se incluye todo el código JavaScript, mediante typescript (superconjunto de JavaScript, que esencialmente añade tipos estáticos y objetos basados en clases.).
- Componente.component.spec.ts: en este archivo se especifican todas las especificaciones de los objetos creados.

El proyecto cuenta con 2 principales componentes que se renderizan en el archivo base del proyecto gracias a la programación orientada a eventos y al sistema de *routing:*

- Componente: Photo-capturer

  En este se ha implementado un elemento HTML *video* que es el encargado de obtener la imagen del usuario. Este componente funge como el componente principal del proyecto. Se ha agregado una imagen sobre el componente de video para indicarle al usuario en que posición poner su mano y así lograr que el recorte de la imagen hecho por el backend siempre contenga una área cuadrada rodeando la mano.

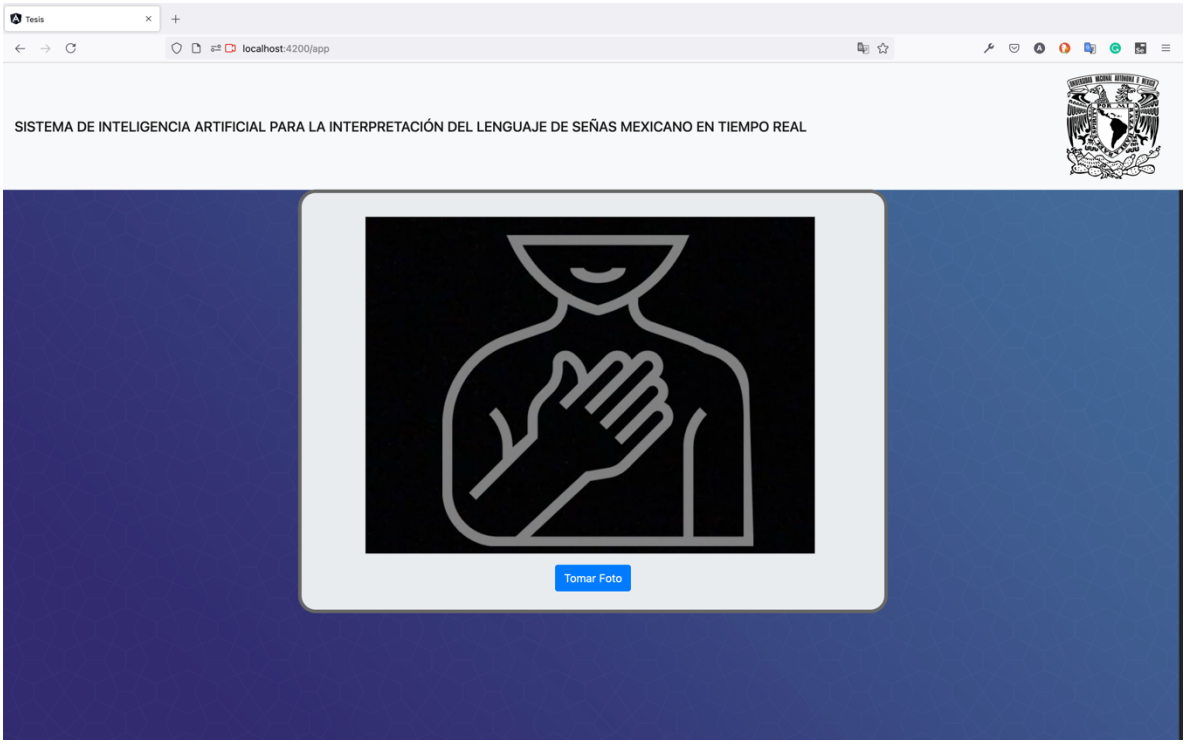


Figura 5.37. Componente Photo-capturer

- Componente: Landing

Este es el encargado de inicializar la aplicación, esta incluida en la ruta principal del proyecto. En este solo se incluyen datos del autor y una dedicatoria.

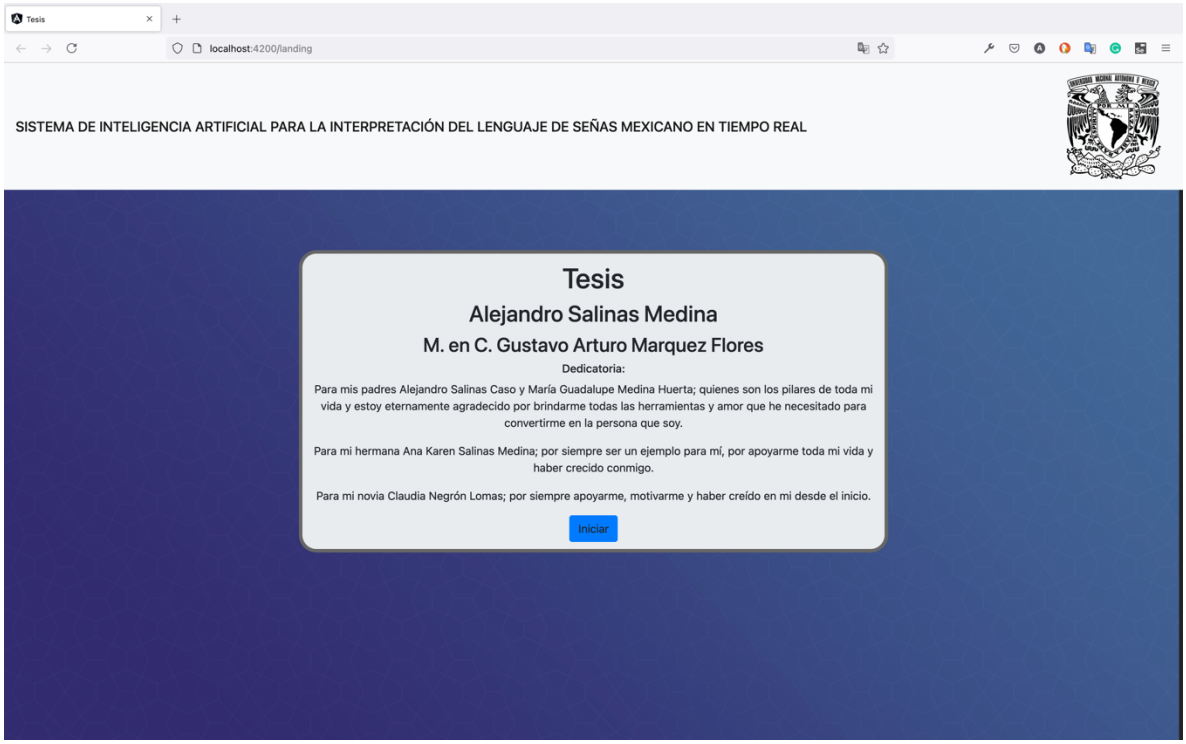


Figura 5.38. Componente Landing

5.5.2 Operación del sistema

- Componente: Landing

  En este, solo se ha incluido un botón vinculado a un evento clic, el cual usa el sistema de routing de AngularJS para cargar el componente principal de la aplicación.

- Componente: Photo-capturer

  Se ha implementado una función inicial al momento de carga de este componente que le pide al usuario escoger entre los dos modos de uso: manual y automático.

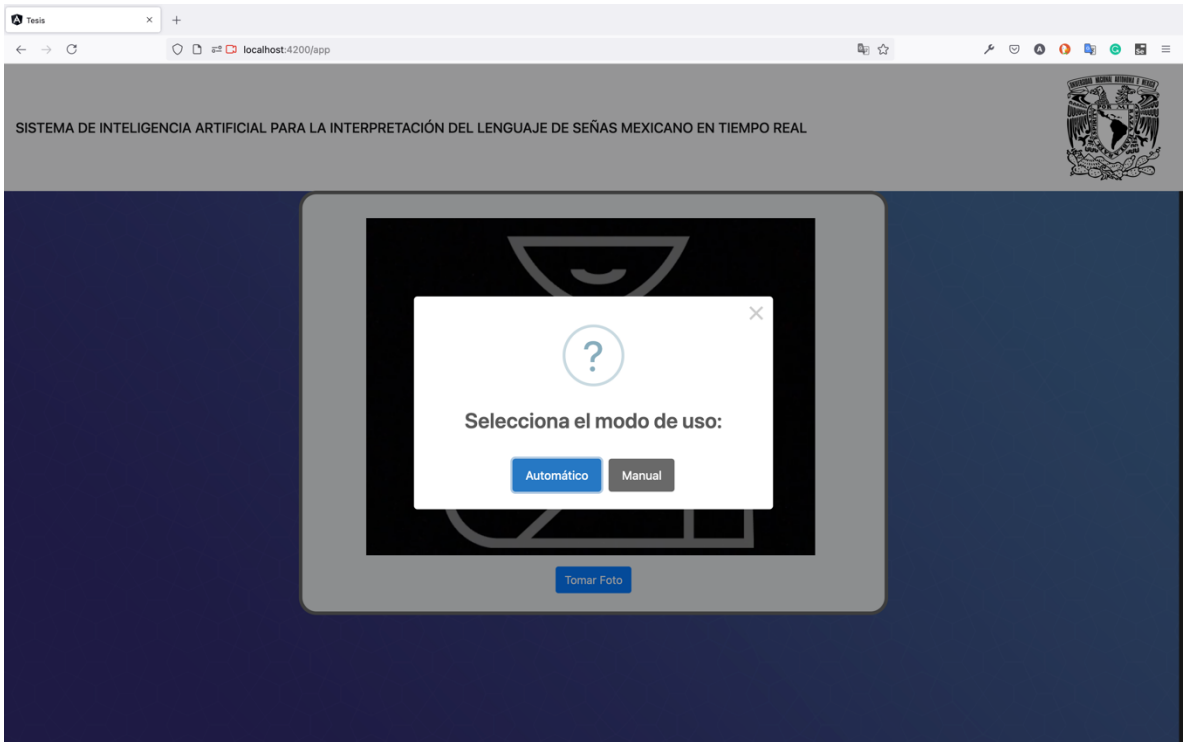


Figura 5.39. Componente Photo-capturer: selección de modo de uso

5.5.2.1 Modo Manual

Al seleccionar el modo manual se renderiza en la interfaz un simple botón para realizar la captura de la imagen.

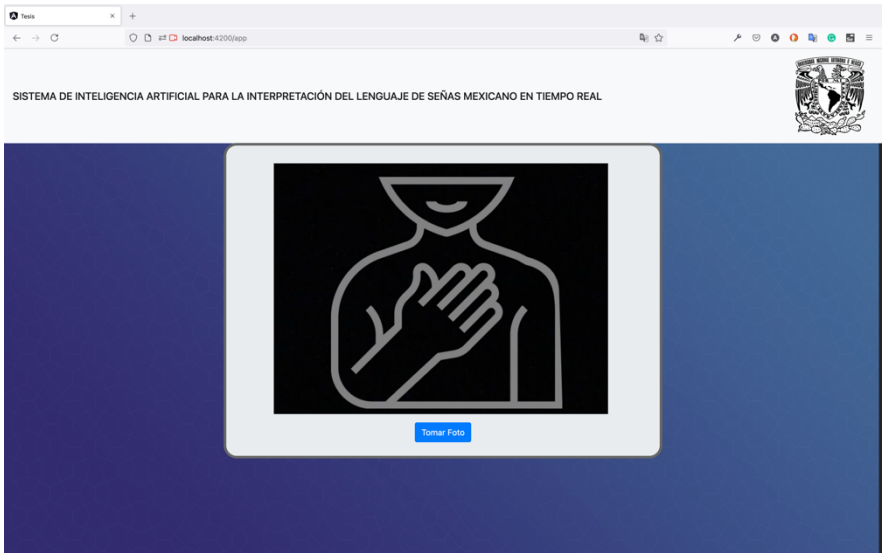


Figura 5.40. Componente Photo-capturer; Modo Manual.

Este botón esta vinculado a través de una función orientada en eventos, con el evento clic, a la función *capture()*. La función *capture()*se define con el siguiente pseudocodigo:

1. Convierte la imagen capturada un formato de URL gracias al método *toDataURL* de JavaScript.
2. Convierte la imagen en formato URL a un objeto BLOB mediante el método *convertDataUrlToBlob* de JavaScript. Un objeto Blob representa un objeto tipo fichero de datos planos inmutables.
3. El objeto BLOB es convertido una interfaz File. La interfaz File provee información acerca de los archivos y permite que el código JavaScript en una página web tenga acceso a su contenido, también facilita su inclusión una interfaz FormData.
4. Realiza la conversión de la interfaz File a la interfaz FormData. La interfaz FormData proporciona una manera sencilla de construir un conjunto de parejas clave/valor que representan los campos de un formulario y sus valores, que pueden ser enviados fácilmente con operaciones HTTP.
5. Se realiza una operación HTTP POST a la ruta del servidor del backend donde se realiza la clasificación de la imagen: "*/predict*" con la interfaz FormData adjunta en la solicitud.
6. Se renderiza la interpretación realizada por el modelo de Inteligencia Artificial en la Interfaz Grafica.

5.5.2.2 Modo Automático

El método automático se describe con el siguiente pseudocodigo:

1. Despliega un aviso al usuario para posicionar la cámara de tal manera que su cuerpo coincida con la posición de la imagen guía.
2. Genera una demora de 10 segundos.
3. Genera un intervalo en el que se hace clic al botón para “Tomar Foto”, esto genera llamadas a la función *capture()* en intervalos de 5 segundos
4. El intervalo es interrumpido en el momento de activar el evento vinculado al botón “Detener”.

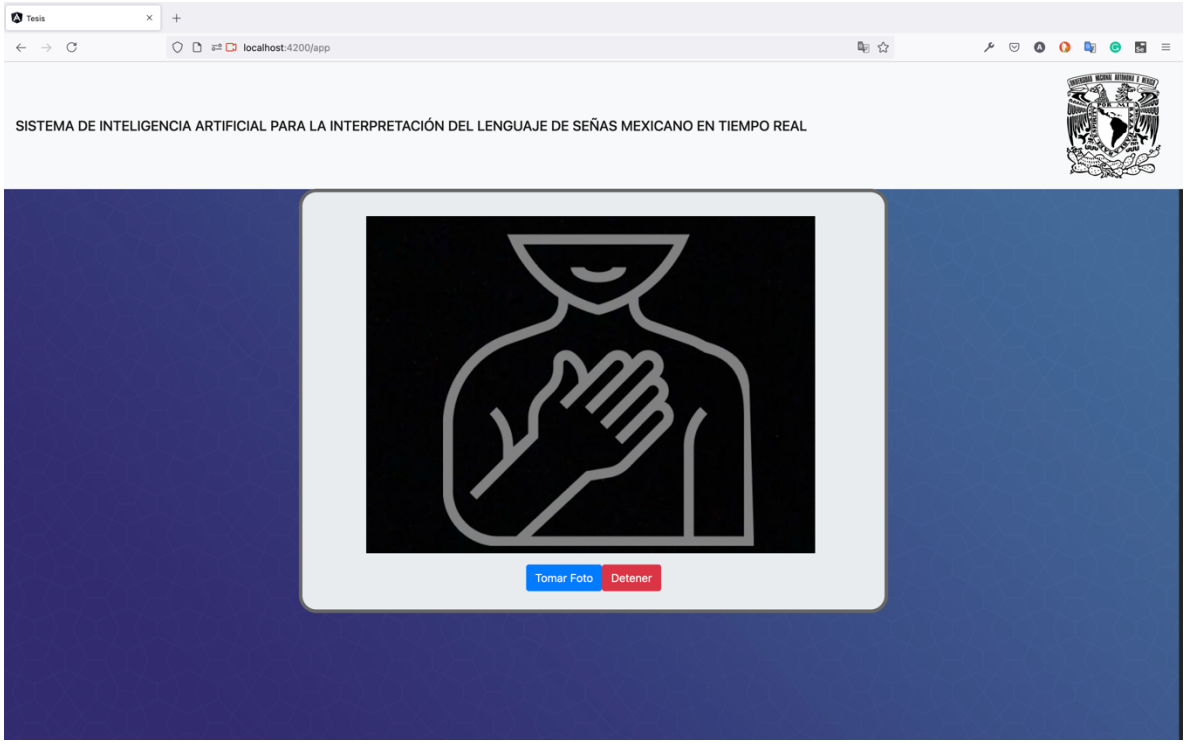
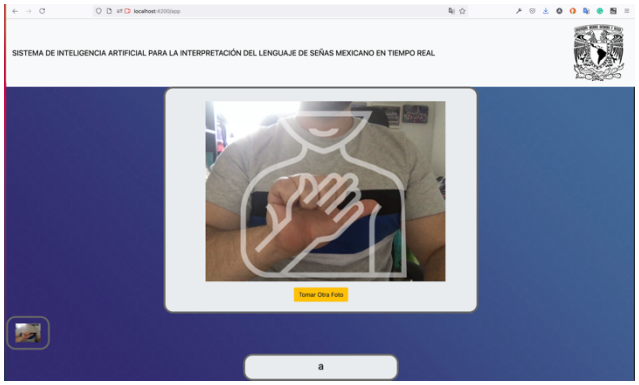
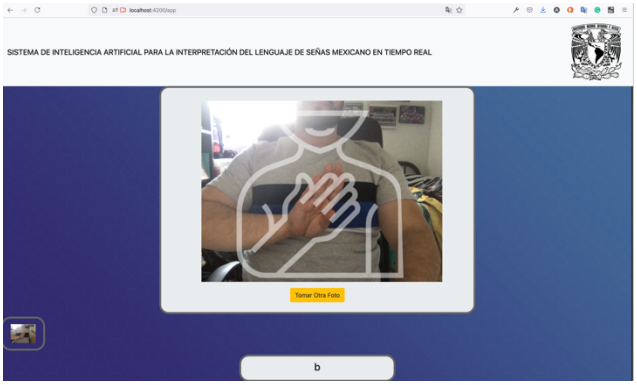


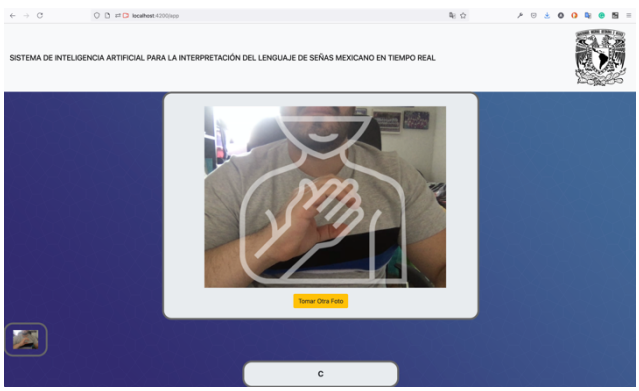
Figura 5.41. Componente Photo-capturer; Modo Automatico.

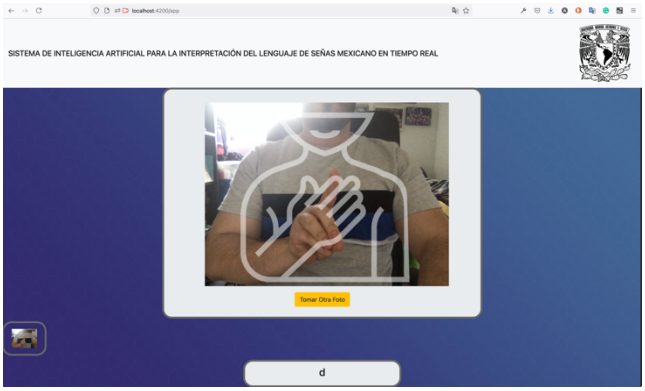
5.5.3 Pruebas del sistema

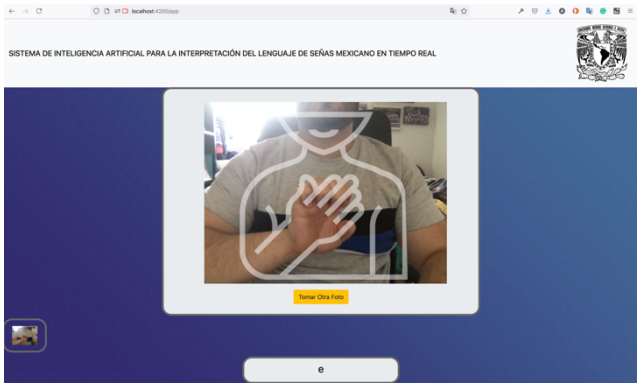
A continuación, se realizan pruebas para todas las letras pertenecientes al conjunto de datos “Lenguaje de Señas Mexicano” generado.

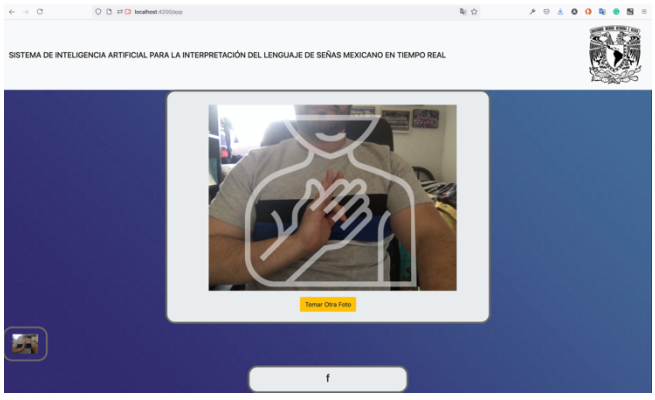


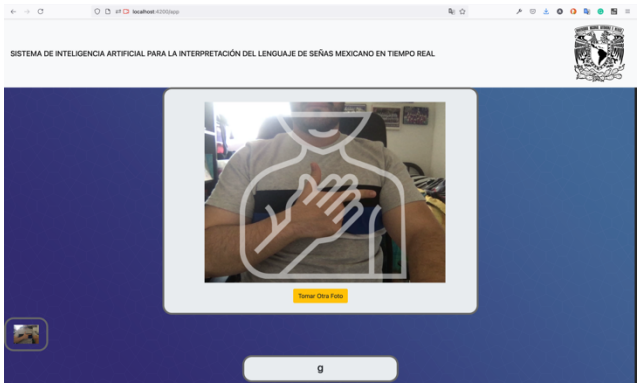


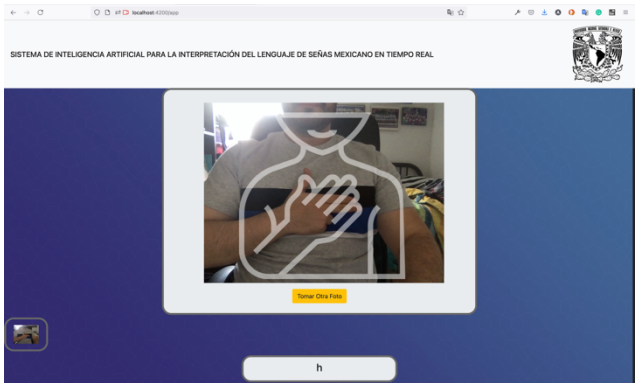


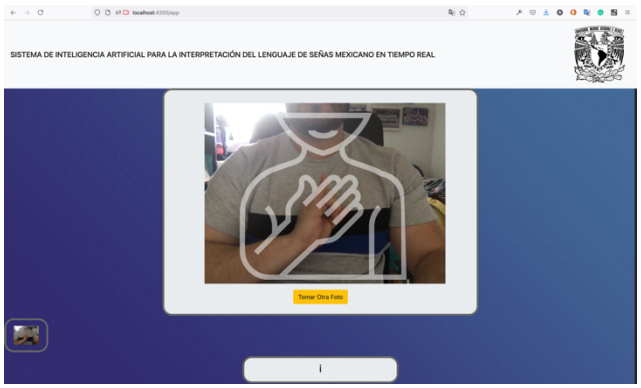


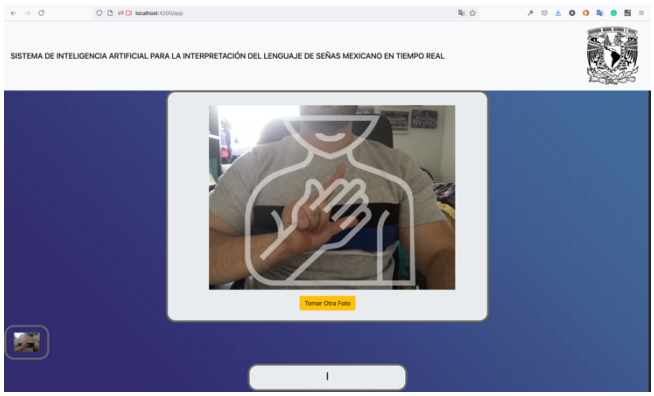


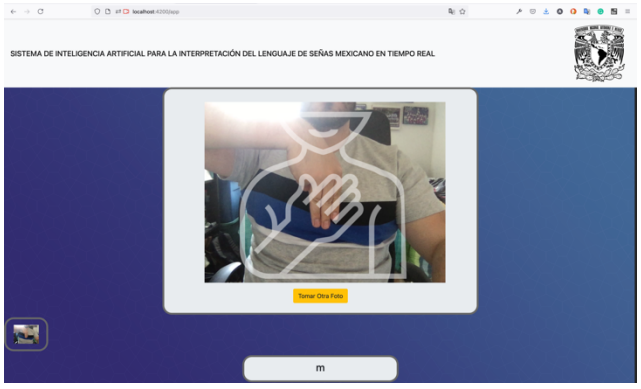


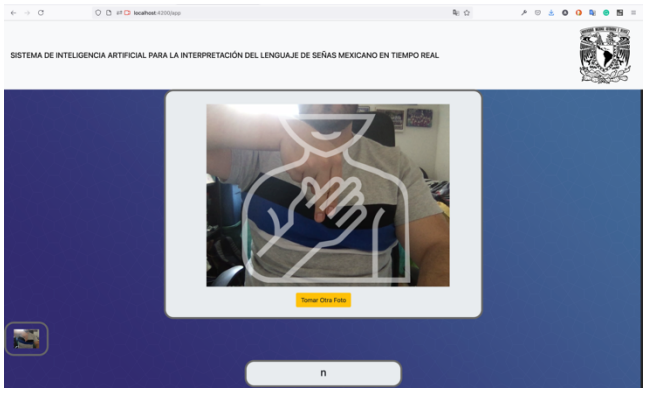

SISTEMA DE INTELIGENCIA ARTIFICIAL PARA LA INTERPRETACIÓN DEL LENGUAJE DE SEÑAS MEXICANO EN TIEMPO REAL
Tomar Otra Foto
g

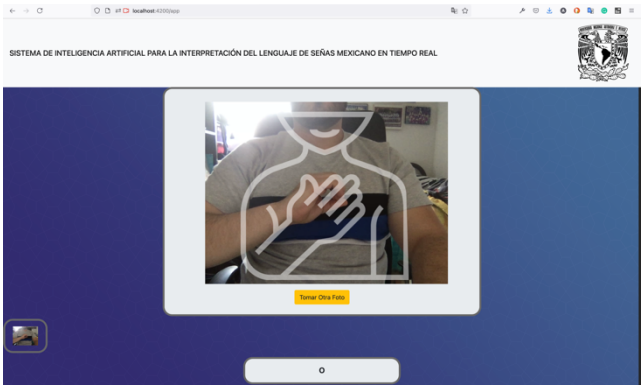

SISTEMA DE INTELIGENCIA ARTIFICIAL PARA LA INTERPRETACIÓN DEL LENGUAJE DE SEÑAS MEXICANO EN TIEMPO REAL
Tomar Otra Foto
h

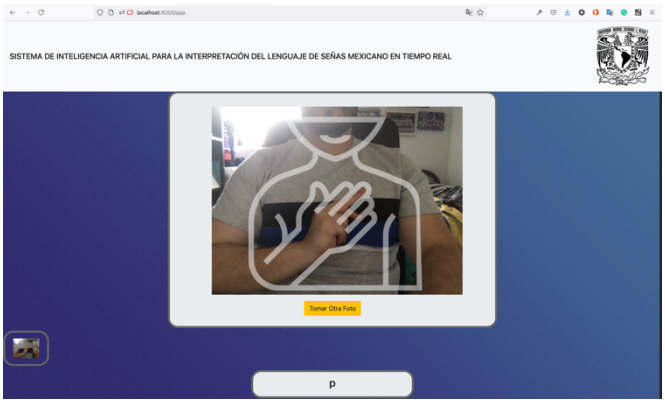

SISTEMA DE INTELIGENCIA ARTIFICIAL PARA LA INTERPRETACIÓN DEL LENGUAJE DE SEÑAS MEXICANO EN TIEMPO REAL
Tomar Otra Foto
i

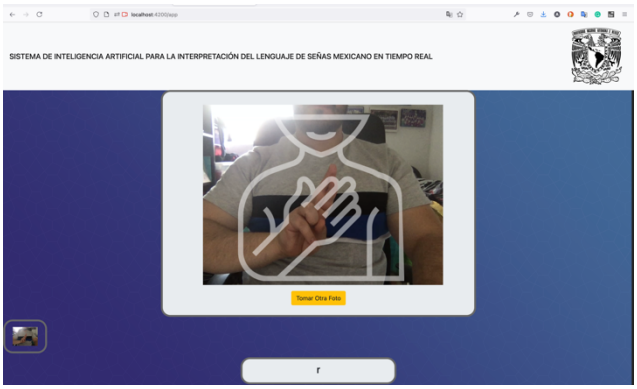

SISTEMA DE INTELIGENCIA ARTIFICIAL PARA LA INTERPRETACIÓN DEL LENGUAJE DE SEÑAS MEXICANO EN TIEMPO REAL
Tomar Otra Foto
l

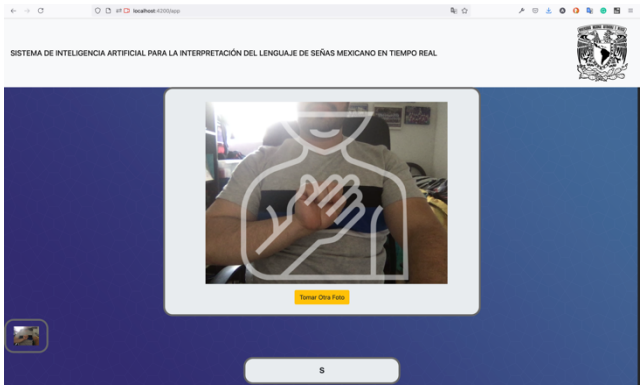

SISTEMA DE INTELIGENCIA ARTIFICIAL PARA LA INTERPRETACIÓN DEL LENGUAJE DE SEÑAS MEXICANO EN TIEMPO REAL
Tomar Otra Foto
m

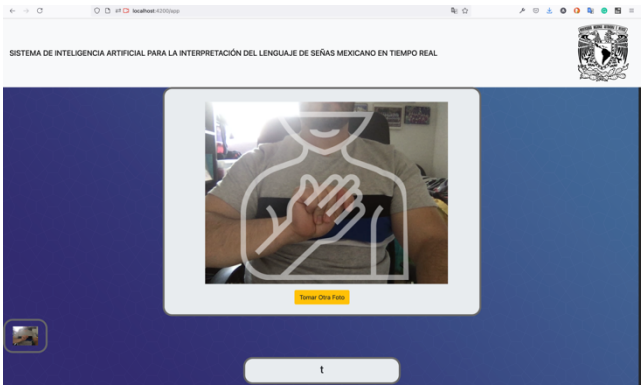

SISTEMA DE INTELIGENCIA ARTIFICIAL PARA LA INTERPRETACIÓN DEL LENGUAJE DE SEÑAS MEXICANO EN TIEMPO REAL
Tomar Otra Foto
n

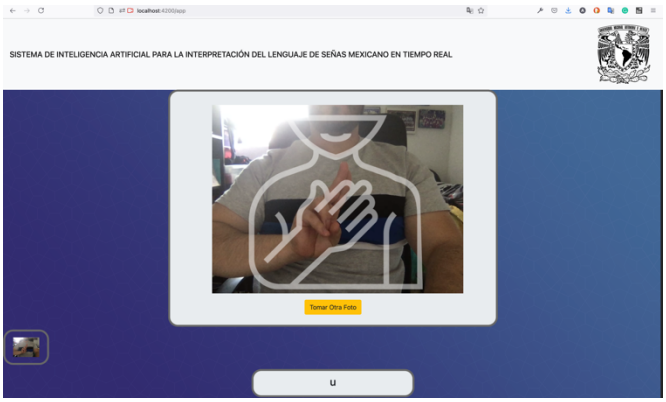

SISTEMA DE INTELIGENCIA ARTIFICIAL PARA LA INTERPRETACIÓN DEL LENGUAJE DE SEÑAS MEXICANO EN TIEMPO REAL
Tomar Otra Foto
o

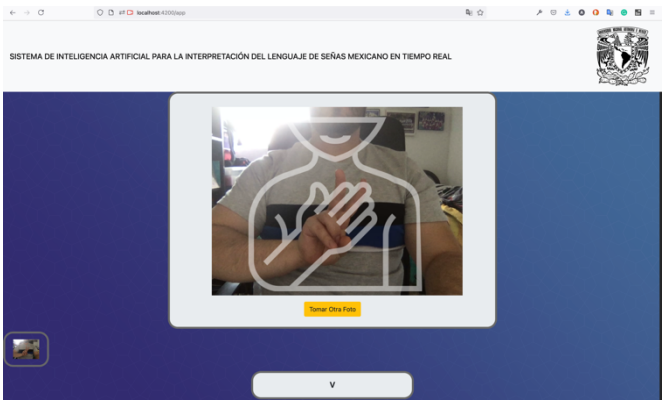

SISTEMA DE INTELIGENCIA ARTIFICIAL PARA LA INTERPRETACIÓN DEL LENGUAJE DE SEÑAS MEXICANO EN TIEMPO REAL
Tomar Otra Foto
p

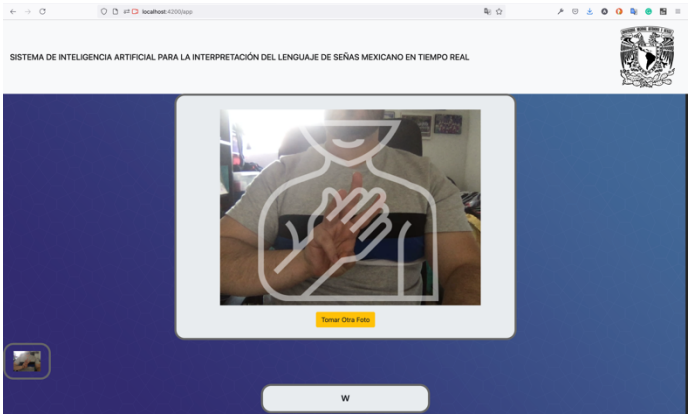

SISTEMA DE INTELIGENCIA ARTIFICIAL PARA LA INTERPRETACIÓN DEL LENGUAJE DE SEÑAS MEXICANO EN TIEMPO REAL
Tomar Otra Foto
r


SISTEMA DE INTELIGENCIA ARTIFICIAL PARA LA INTERPRETACIÓN DEL LENGUAJE DE SEÑAS MEXICANO EN TIEMPO REAL
Tomar Otra Foto
s


SISTEMA DE INTELIGENCIA ARTIFICIAL PARA LA INTERPRETACIÓN DEL LENGUAJE DE SEÑAS MEXICANO EN TIEMPO REAL
Tomar Otra Foto
t


SISTEMA DE INTELIGENCIA ARTIFICIAL PARA LA INTERPRETACIÓN DEL LENGUAJE DE SEÑAS MEXICANO EN TIEMPO REAL
Tomar Otra Foto
u


SISTEMA DE INTELIGENCIA ARTIFICIAL PARA LA INTERPRETACIÓN DEL LENGUAJE DE SEÑAS MEXICANO EN TIEMPO REAL
Tomar Otra Foto
v


SISTEMA DE INTELIGENCIA ARTIFICIAL PARA LA INTERPRETACIÓN DEL LENGUAJE DE SEÑAS MEXICANO EN TIEMPO REAL
Tomar Otra Foto
w

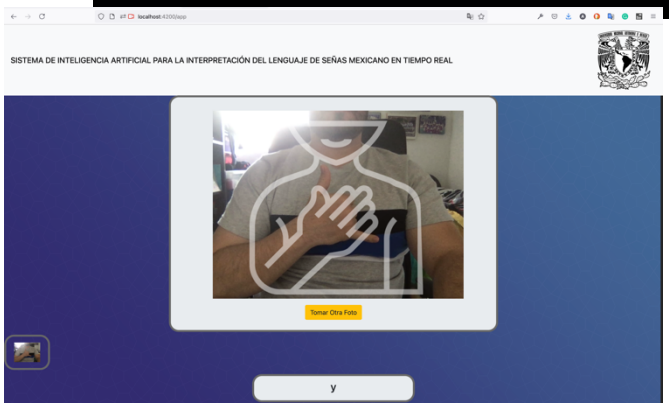


Figura 5.42. Modo manual para interpretar letra por letra los valores incluidos en el conjunto de datos “Lenguaje de Señas Mexicano”.

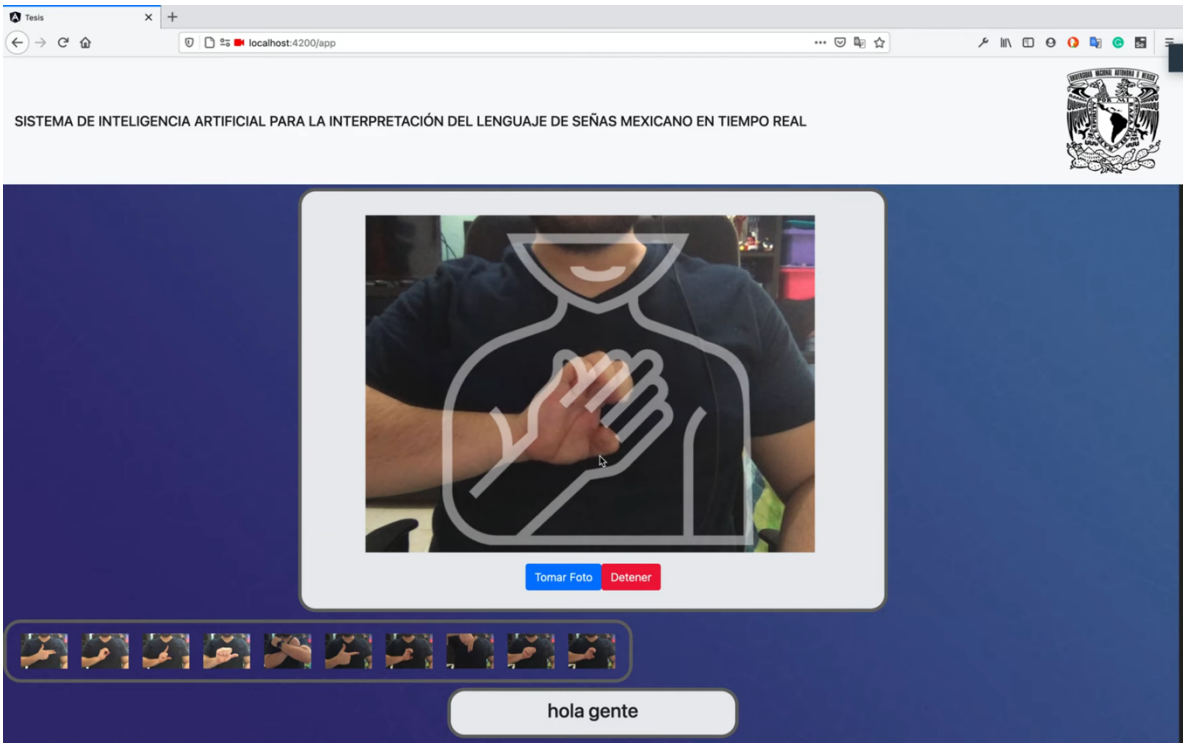


Figura 5.43. Modo automático para interpretar la frase “hola gente” del lenguaje de señas mexicano a lenguaje escrito.

Capítulo 6. Conclusiones y trabajo futuro

" Algunas personas llaman a esto inteligencia artificial, pero la realidad es que esta tecnología nos mejorará. Entonces, en lugar de inteligencia artificial, creo que aumentaremos nuestra inteligencia." - Ginni Rometty

CAPITULO 6. CONCLUSIONES Y TRABAJO

6.1 Conclusión

Este proyecto de tesis ha podido demostrar que el aprendizaje por transferencia es una herramienta muy eficaz para abordar problemas en los que se dispone de pocos datos. El enfoque de utilizar el aprendizaje por transferencia para aumentar la precisión del modelo será replicable en muchos otros lenguajes de señas. Es importante considerar que, con el uso de redes neuronales, el impacto de la supervisión humana se minimiza y se es posible derivar el significado de algunos patrones que pueden llegar a ser demasiado complejos para la percepción humana. No obstante, el uso de redes neuronales genera una capa de abstracción en la que los humanos no tienen control para comprender o cambiar. Como resultado, el entrenamiento del algoritmo podría resultar en una situación en la que este clasifica erróneamente o deriva demasiado significado de una situación binaria. Por eso no se debe descuidar el uso de métodos más tradicionales sobre la base de técnicas nuevas y más complejas.

Se propuso la comparación de distintos modelos de aprendizaje automático de extremo a extremo para traducir imágenes del lenguaje de señas mexicano. Se ha demostrado que el problema de tener un pequeño conjunto de datos se puede resolver mediante el aprendizaje por transferencia con un modelo previamente entrenado: el modelo con una precisión más alta es capaz de clasificar imágenes del lenguaje de señas mexicano con una precisión del 91%. En este caso, la lengua de señas mexicana no se ha incluido lo suficiente en el sector tecnológico para poder tener un conjunto de datos con más validaciones y el trabajo de más personas. Es deseable que el conjunto de datos y modelos generados entusiasme a más personal académico a aportar mas investigación hacia este tema. Gracias a que se ha planteado que este proyecto sea accesible a todo el público, a largo plazo, este proyecto lograra beneficiar a las personas sordas que tienen acceso a la tecnología para mejorar la buena salud, la educación, el trabajo y ayudara a la reducción de las desigualdades.

6.2 Oportunidades

Una limitación respecto al modelo utilizado para interpretar los signos es el uso de modelos basados en arquitecturas convolucionales, ya que estas solo logran realizar clasificación sobre imágenes y existen 6 letras que requieres de movimientos dinámicos (“J”,” K”,” Ñ”,”Q”,”X”,”Z”). Para realizar la inclusión de esos caracteres, es necesario plantear una arquitectura hibrida, combinación de modelos basados en redes neuronales convolucionales para realizar la extracción de características y modelos basados en redes neuronales recurrentes que logren llevar una cuenta de los tiempos para los datos extraídos por la arquitectura convolucional. Este proyecto se centra en el uso del aprendizaje por transferencia para interpretar el alfabeto de señas mexicano. Por lo tanto, no sugiere necesariamente que se pueda hacer lo mismo con el resto del lenguaje de signos o con los alfabetos manuales de otros lenguajes de signos. Finalmente, el conjunto de datos utilizado para lenguaje de señas mexicano se desarrolló específicamente para este proyecto en función de obtener el grado académico, por lo tanto, no se han utilizado otras posibles fuentes de datos. También es necesario que se incluya videos de estos movimientos dinámicos representantes del conjunto de letras no incluidos en el conjunto de datos mencionado en este documento. Debido a la naturaleza del proyecto, solo se ha generado una base de datos con 10 sujetos, sin embargo, la inclusión de mas sujetos en la base de datos lograría mejor la capacidad de generalización de los modelos implementados en esta tesis.

Bibliografía

[1] Organization, W.H., 2018. Addressing the rising prevalence of hear-ing loss.https://apps.who.int/iris/bitstream/handle/10665/260336/9789241550260-eng.pdf?sequence=1&isAllowed=y. [En linea; accedido 30-Diciembre-2020].

[2] Emmorey, K., 2002. Language, Cognition, and the Brain: InsightsFrom Sign Language Research. Lawrence Erlbaum Associates Inc.

[3]"Flask." http://flask.pocoo.org/.

[4]Grinberg, Miguel. Flask Web
Development: Developing Web Applications with Python." O'Reilly Media, Inc.", 2014.

[5]Berson, Alex. "Client-server architecture." No. IEEE-802. McGraw-Hill, 1992

[6] Berners-Lee, T., Fielding, R., & Frystyk, H. (1996). Hypertext Transfer Protocol -- HTTP/1.0. *MIT/LCS*.

[7] I. H. Sarker, K. Apu, “MVC Architecture Driven Design and Implementation of Java Framework for Developing Desktop Application,” International Journal of Hybrid Information Technology, Vol.7, No.5, pp. 317-322, 2014.

[8] Card, D., Page, G., & McGarry, F. (1985). Criteria for software modularization. *Computer Sciences Corporation, Silver Spring.*

[9] D. L. Parnas, "On the Criteria to be Used in Decomposing Systems into Modules,' ACM Communications, December 1972, vol. 15, no. 12, pp. 1053-1058

[10] B. W. Rernighan and P. S. Plauger, The Elements of Proqramminq Style. New York: HCGraw Hill, 1974, p. 126

[11] J. D. Bowen, .Module Size: A Standard or Heuristic?,- Journal of Systems and Soft- =, 1984, no. 4, pp. 327-332

[12] Yangqing "Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick,Sergio Guadarrama, and Trevor" Darrell. "caffe: Convolutional architecture for fast featureembedding"."arXiv preprint arXiv:1408.5093", "2014".

[13] Frank Seide and Amit Agarwal. Cntk: Microsoft's open-source deep-learning toolkit. InProceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discoveryand Data Mining, KDD '16, pages 2135–2135, New York, NY, USA, 2016. ACM.

[14] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro,Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow,Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser,Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, DerekMurray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, KunalTalwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, PeteWarden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[15] Theano Development Team. Theano: A Python framework for fast computation of mathematicalexpressions.arXiv e-prints, abs/1605.02688, May 2016.

[16] Seiya Tokui, Kenta Oono, Shohei Hido, and Justin Clayton. Chainer: a next-generation opensource framework for deep learning. InProceedings of Workshop on Machine Learning

Systems(LearningSys) in The Twenty-ninth Annual Conference on Neural Information ProcessingSystems (NIPS), 2015.

[17] Ronan Collobert, Samy Bengio, and Johnny Mariéthoz. Torch: a modular machine learningsoftware library. Technical report, Idiap, 2002.

[18] G. Neubig, C. Dyer, Y.
Goldberg, A. Matthews, W. Ammar, A. Anastasopoulos, M. Balles-teros, D. Chiang, D. Clothiaux, T. Cohn, K. Duh, M. Faruqui, C. Gan, D. Garrette, Y. Ji,L. Kong, A. Kuncoro, G. Kumar, C. Malaviya, P. Michel, Y. Oda, M. Richardson, N. Saphra,S. Swayamdipta, and P. Yin. DyNet: The Dynamic Neural Network Toolkit.ArXiv e-prints,January 2017.

[19] F. Pérez and B. E. Granger, "Ipython: a system for interactive scientificcomputing," Computing in Science & Engineering, vol. 9, no. 3, 2007.

[20] B. M. Randles, I. V. Pasquetto, M. S. Golshan, and C. L. Borgman, "Usingthe jupyter notebook as a tool for open science: An empirical study," inACM/IEEE Joint Conference on Digital Libraries (JCDL).IEEE, 2017,pp. 1–2

[21] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," nature, vol. 521, no.7553, p. 436, 2015.

[22] A. Brodtkorb, C. Dyken, T. Hagen, J. Hjelmervik, and O. Storaasli, "State-of-the-art in heterogeneous computing," Scientific Programming, vol. 18,no. 1, pp. 1–33, 2010.

[23] NVIDIA Corporation, "Tesla V100 performance guide: deep learning andHPC applications," NVIDIA Corporation Whitepaper, 2016.

[24] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski,G. Lee, D. A. Patterson, A. Rabkin, I. Stoica et al., “Above the clouds: Aberkeley view of cloud computing,” Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley, Tech. Rep.,2009.

[25] Carneiro, T., Medeiros Da Nobrega, R., Nepomuceno, T., Bian, G., De Albuquerque, V., & Filho, P. (2018). Performance Analysis of Google Colaboratory as a Tool for Accelerating Deep Learning Applications. *IEEE Access*, *6*, 61677-61685. doi: 10.1109/access.2018.2874767

[26] Hernandez-Rebollar, J., Kyriakopoulos, N., Lindeman, R., 2004. Anew instrumented approach for translating american sign languageinto sound and text. Sixth IEEE International Conference on Auto-matic Face and Gesture Recognition .

[27] Glenn, C.M., Mandloi, D., Sarella, K., Lonon, M., 2005. An imageprocessing technique for the translation of asl finger-spelling to digitalaudio or text, in: Instructional Technology and Education of the deafSymposium, Rochester, NY, pp. 1–7

[28] Akram, S., Beskow, J., Kjellstrom, H., 2012. Visual Recogni-tion of Isolated Swedish Sign Language Signs. arXiv e-prints ,arXiv:1211.3901arXiv:1211.3901.

[29] Sergiou, C., Siganos, D., 1996. Neural networks. urveys and Presen-tations in Information Systems Engineering (SURPRISE) 96.

[30] Weissmann, J., Salomon, R., 1999. Gesture recognition for virtualreality applications using data gloves and neural networks. IJCNN’99.International Joint Conference on Neural Networks 3, 2043–2046.

[31] Pugeault, N., Bowden, R., 2011. Spelling it out: Real-time asl fin-gerspelling recognition, in: 2011 IEEE International Conference onComputer Vision Workshops (ICCV Workshops), pp. 1114–1119.

[32] Koller, O., Ney, H., Bowden, R., 2016. Deep hand: How to train a cnnon 1 million hand images when your data is continuous and weaklylabelled, in: 2016 IEEE Conference on Computer Vision and PatternRecognition (CVPR), pp. 3793–3802.

[33] Mocialov, B., Turner, G., Lohan, K., Hastie, H., 2017. Towards con-tinuous sign language recognition with deep learning.

[34] Quirk, T., Kamaal, K., 2018. How we used ai to translate signlanguage in real time.https://blog.coviu.com/2018/09/21/how-we-used-ai-to-translate-sign-language-in-real-time/.[En linea; accedido 12-Enerol-2021

[35] Mocialov, B., Hastie, H., Turner, G., 2018. Transfer learning forbritish sign language modelling, in: Proceedings of the Fifth Work-shop on NLP for Similar Languages, Varieties and Dialects (VarDial2018), pp. 101–110.

[36] https://pytorch.org/tutorials/beginner/basics/data_tutorial.html

[37] https://pytorch.org/docs/1.6.0/torchvision/transforms.html

[38] tensorflow.org/tensorboard.

[39] Jadhav, M., Sawant, B., & Deshmukh, A. (2015). Single Page Application using AngularJS. *International Journal Of Computer Science And Information Technologies*, *6*(3), 2876-2879.

[40] Paul, P., 2018. What's it like to be deaf? reflections on signed lan-guage, sustainable development, and equal opportunities. AmericanAnnals of the Deaf Gallaudet University Press 163. doi:10.1353/aad.2018.0030.

[41] Emmorey, K., 2002. Language, Cognition, and the Brain: InsightsFrom Sign Language Research. Lawrence Erlbaum Associates Inc.

[42] Halvardsson, G., Peterson, J., Soto-Valero, C., & Baudry, B. (2021). Interpretation of Swedish Sign Language Using Convolutional Neural Networks and Transfer Learning. *SN Computer Science*, *2*(3). doi: 10.1007/s42979-021-00612-w

[43] Serafín de Fleishmann, M., & González Pérez, R. (2011). *Manos con Voz. Diccionario de Lengua de Señas Mexicana* (pp. 15-19). México: Consejo Nacional para Prevenir la Discriminación.

[44] Qin X, Wen Z, Vivian J. Image processing based on Open CV[J]. Electronic Test, 2011, 07: 39-41.

[45] Wang F, Li Y, Liu J. Implementation of machine vision image processing technology based on Open CV[J]. Mechanical and Electronic, 2010, (06): 54-57.

[46] Shafranovich, Y. (2005). Common Format and MIME Type for Comma-Separated Values (CSV) Files. *Solidmatrix Technologies, Inc.*.

[47] Freed, N., Klensin, J., and J. Postel, "Multipurpose Internet
Mail Extensions (MIME) Part Four: Registration Procedures", BCP
13, RFC 2048, November 1996.

[48] J¨urgen Umbrich, Sebastian Neumaier, and Axel Polleres. Quality assess-ment & evolution of open data portals. InIEEE International Conferenceon Open and Big Data, Rome, Italy, August 2015

[49] Y. Shafranovich. Common Format and MIME Type for Comma-SeparatedValues (CSV) Files. RFC 4180 (Informational), October 2005.

[50] Mitlohner, J., Neumaier, S., Umbrich, J., & Polleres, A. (2016). Characteristics of Open Data CSV Files. *2016 2Nd International Conference On Open And Big Data (OBD)*. doi: 10.1109/obd.2016.18

[51] Krogh, A. (2008). What are artificial neural networks?. *Nature Biotechnology*, *26*(2), 195-197. doi: 10.1038/nbt1386

[52] C. M. Bishop,Pattern Recognition and Machine Learning(Springer, Berlin, 2006).

[53] Y. LeCun, Y. Bengio, and G. Hinton, Deep learning,Nature521,436(2015).

[54] Y. Bengio, I. J. Goodfellow, and A. Courville,Deep Learning(MIT Press, Cambridge, MA, 2015)

[55] Mehta, D., Zhao, X., Bernal, E., & Wales, D. (2018). Loss surface of XOR artificial neural networks. *Physical Review E*, *97*(5). doi: 10.1103/physreve.97.052307

[56] W. Zhang, "A Generalized ADALINE Neural Network for System Identification," 2007 IEEE International Conference on Control and Automation, 2007, pp. 2705-2709, doi: 10.1109/ICCA.2007.4376853.

[57] Widrow, B., & Lehr, M. (1990). Perceptrons, Adalines, and backpropagation.

[58] B. Widrow and M. A. Lehr, "30 years of adaptive neural networks: perceptron, Madaline, and backpropagation," in Proceedings of the IEEE, vol. 78, no. 9, pp. 1415-1442, Sept. 1990, doi: 10.1109/5.58323.

[59] Sibi, P., ALLWYN JONES, S., & SIDDARTH, P. (2013). ANALYSIS OF DIFFERENT ACTIVATION FUNCTIONS USING BACK PROPAGATION NEURAL NETWORKS. *Journal Of Theoretical And Applied Information Technology*, *47*(3), 1264 - 1268.

[60] Chenghao Cai, Yanyan Xu, Dengfeng Ke, and K. Su, "Deep neural network with multistate activation functions," 2015.

[61] Bing Xu, Ruitong Huang, and M. Li, "Revised saturated activation functions," in International conference on Learning Representation, 2016.

[62] Xavier Glorot, Antoine Bordes, and Y. Bengio, "Deep Sparse Rectified Neural Network," in International conference on Artificial Intelligence and Statistics (AISTATS), Fort Lauderdale, FL, USA, 2011.

[63] Andrew L. Maas, Awni Y. Hannun, and A. Y. Ng, "Retifier Nonlinearities Improve Neural Network Acostic Models," in International conference on Machine Learning, Atlanta, Georgia, USA, 2013.

[64] Xiaojie Jin, Chunyan Xu, Jiashi Feng, Yunchao Wei, Junjun Xiong, and S. Yan, "Deep Learning with s-shaped rectified linear activation units " 2015

[65] Kemal Adem , Serhat Kilic ̧arslan , Onur C ̈omert , Classification and Diag-nosis of Cervical Cancer with Softmax classification with stacked autoencoder,Expert Systems WithApplications(2018), doi: https://doi.org/10.1016/j.eswa.2018.08.050

[66] S. Albawi, T. A. Mohammed and S. Al-Zawi, "Understanding of a convolutional neural network," 2017 International Conference on Engineering and Technology (ICET), 2017, pp. 1-6, doi: 10.1109/ICEngTechnol.2017.8308186.

[67] Kleinberg et al. An Alternative View: When Does SGD Escape Local Minima?, 2018

[68]Zhu et al. The Anisotropic Noise in Stochastic Gradient Descent: Its Behavior of Escaping from Sharp Minima and Regularization Effects, 2019.

[69]Keskar et al. On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima, 2017.

[70]]http://www.deeplearningbook.org/contents/convnets.html

[71] https://www.opendatascience.com/blog/an-intuitive-explanation-fconvolutionalneuralnetworks/?utm_source=Open+Data+Science+Newsletter&utm_campaign=f4ea9cc60fEMAIL_CAMPAIGN_2016_12_21&_medium=email&utm_term=0_2ea92bb125-f4ea9cc60f-245860601.

[72] D. Stutz and L. Beyer, “Understanding Convolutional Neural Networks,” 2014

[73] K. Teilo, “ An Introduction to Convolutional Neural Networks,” no. NOVEMBER 2015, pp. 0–11, 2016.

[74]V. Dumoulin and F. Visin, “A guide to convolution arithmetic for deep learning,” pp. 1–28, 2016.

[75] N. Kwak, “Introduction to Convolutional Neural Networks (CNNs),” 2016.

[76] Y. Guo, Y. Liu, A. Oerlemans, S. Wu, and M. S. Lew, “Author ’ s Accepted Manuscript Deep learning for visual understanding : A review To appear in : Neurocomputing,” 2015.

[77] I. Kokkinos, E. C. Paris, and G. Group, “Introduction to Deep Learning Convolutional Networks, Dropout, Maxout 1,” pp. 1–70.

[78] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, C. V Jan, J. Krause, and S. Ma, “ImageNet Large Scale Visual Recognition Challenge.”.

[79] N. Srivastava et al: “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”. Journal of Machine Learning Research, Vol. 15, pp. 1929- 1958, (2014).

[80] G. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neu-ral networks by preventing co-adaptation of feature detectors.http://arxiv.org/abs/1207.0580,2012.

[81] Baldi, P., & Sadowski, P. (2015). Understanding Dropout. *Department Of Computer Scienceuniversity Of California, Irvine*.

[82] Dumoulin, Vincent & Visin, Francesco. (2016). A guide to convolution arithmetic for deep learning.

[83] https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/tesla-product-literature/NV-tesla-p100-pcie-PB-08248-001-v01.pdf

[84] A. Kumar and S. P. Panda, "A Survey: How Python Pitches in IT-World," 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon), 2019, pp. 248-251, doi: 10.1109/COMITCon.2019.8862251.

[85] X. Cai, H. Langtangen and H. Moe, "On the Performance of the Python Programming Language for Serial and Parallel Scientific Computations," Scientific Programming, vol. 13, no. 1, pp. 31-56, 2005.

[86] A. Nagpal and G. Gabrani, "Python for Data Analytics, Scientific and Technical Applications," 2019 Amity International Conference on Artificial Intelligence (AICAI), 2019, pp. 140-145, doi: 10.1109/AICAI.2019.8701341.

[87] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin,A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation inpytorch,"Neural Information Processing Systems, 2017.

[88] F. Laporte, J. Dambre and P. Bienstman, "Photontorch: Simulation and Optimization of Large Photonic Circuits Using the Deep Learning Framework PyTorch," 2019 IEEE Photonics Society Summer Topical Meeting Series (SUM), 2019, pp. 1-2, doi: 10.1109/PHOSST.2019.8794941.

[89] T. E. Oliphant,A guide to NumPy, vol. 1. Trelgol Publishing USA, 2006

[90] Y Bengio and X Glorot. Understanding the difficulty of training deep feedforward neural networks.International Conference on Artificial Intelligenceand Statistics, pages 249–256, 01 2010.

[91] Datta, L. (2020). A Survey on Activation Functions and their relation with Xavier and He Normal Initialization. *Delft University Of Technology*.

[92] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed,D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabi-novich, "Going deeper with convolutions," inProceed-ings of the IEEE conference on computer vision andpattern recognition, 2015, pp. 1–9.

[93] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residuallearning for image recognition,” inProceedings of theIEEE conference on computer vision and pattern recog-nition, 2016, pp. 770–778.

[94] J. Hu, L. Shen, and G. Sun, “Squeeze-and-excitationnetworks,”arXiv preprint arXiv:1709.01507, 2017.

[95] J. Duchi, E. Hazan, and Y. Singer, “Adaptive subgradientmethods for online learning and stochastic optimization,”Journal of Machine Learning Research, vol. 12, no. Jul,pp. 2121–2159, 2011.

[96] M. D. Zeiler, “Adadelta: an adaptive learning ratemethod,”arXiv preprint arXiv:1212.5701, 2012.

[97] T. Tieleman and G. Hinton, “Lecture 6.5—RmsProp:Divide the gradient by a running average of its recentmagnitude,” COURSERA: Neural Networks for MachineLearning, 2012.

[98] D. Kingma and J. Ba, “Adam: A method for stochasticoptimization,”arXiv preprint arXiv:1412.6980, 2014.

[99] Zhang, Z. (2018). *Improved Adam Optimizer for Deep Neural Networks. 2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS).* doi:10.1109/iwqos.2018.8624183

[100] D. P. Kingma and J. L. Ba, “Adam: A Method for Stochastic Optimization,” Proc. of 3rd International Conference for Learning Representations, arXiv:1412.6980, 2015.

[101] Matsumura, S., & Nakashima, T. (2017). *Incremental learning for SIRMs fuzzy systems by Adam method. 2017 Joint 17th World Congress of International Fuzzy Systems Association and 9th International Conference on Soft Computing and Intelligent Systems (IFSA-SCIS).* doi:10.1109/ifsa-scis.2017.8023307

[102] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Deep Residual Learning for Image Recognition.Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770-778

[103] Lin, H., & Jegelka, S. (2018). ResNet with one-neuron hidden layers is a Universal Approximator.

[104] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenetclassification with deep convolutional neural networks. InAdvances in neural information processing systems, pages1097–1105, 2012

[105] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh,S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein,et al.Imagenet large scale visual recognition challenge.2014.

[106] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed,D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich.Going deeper with convolutions. In Proceedings of the IEEEConference on Computer Vision and Pattern Recognition,pages 1–9, 2015.

[107] C. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the Inception Architecture for Computer Vision. *2016 IEEE Conference On Computer Vision And Pattern Recognition (CVPR)*. doi: 10.1109/cvpr.2016.308

[108] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep intorectifiers: Surpassing human-level performance on imagenetclassification.arXiv preprint arXiv:1502.01852, 2015

[109] Y. Movshovitz-Attias, Q. Yu, M. C. Stumpe, V. Shet,S. Arnoud, and L. Yatziv. Ontological supervision for finegrained classification of street view storefronts. InProceedings of the IEEE Conference on Computer Vision and PatternRecognition, pages 1693–1702, 2015.

[110] Z. Huang, Z. Pan and B. Lei, "Transfer Learning with Deep Convolutional Neural Network for SAR Target Classification with Limited Labeled Data," Remote Sensing, vol. 9, no. 9, p. 907, 2017.

[111] Z. Huang, Z. Pan and B. Lei, "Transfer Learning with Deep Convolutional Neural Network for SAR Target Classification with Limited Labeled Data," Remote Sensing, vol. 9, no. 9, p. 907, 2017.

[112] M. A. Mufti, E. A. Hadhrami, B. Taha and N. Werghi, "Automatic target recognition in SAR images: Comparison between pre-trained CNNs in a tranfer learning based approach," 2018 International Conference on Artificial Intelligence and Big Data (ICAIBD), 2018, pp. 160-164, doi: 10.1109/ICAIBD.2018.8396186.