Universidad Andina del Cusco
Repositorio Digital

UNIVERSIDAD ANDINA DEL CUSCO

FACULTAD DE INGENIERÍA Y ARQUITECTURA

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

TESIS

PROTOTIPO DE TRADUCTOR DE LENGUAJE DE SEÑAS PERUANAS BÁSICAS USANDO MACHINE LEARNING

Línea de Investigación: Ciencia de Datos

Presentado por:

- Bach. Jimenez Moreano Anaid
- Bach. Qquecho Ccachainca Brenda Alexandra

Para optar el Título Profesional de:

Ingeniero de Sistemas

Asesor:

Ing. Ivan Molero Delgado

CUSCO-PERÚ

2020

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

DEDICATORIA

El presente trabajo de investigación está dedicado principalmente a Dios, por ser el inspirador y darnos fuerza para continuar en este proceso de obtener uno de los anhelos más deseados.

A mis padres Juan Jimenez Benavides y Claudia Moreano Casquino, por su amor, trabajo y sacrificio en todos estos años, gracias a ustedes he logrado llegar hasta aquí y convertirme en una persona llena de valores. Ha sido el orgullo y el privilegio de ser su hija, son los mejores padres.

A mis hermanos, mi hermano Jürgen mi gran fuente de inspiración, el personaje que me acompaño y alentó a incurrir en el mundo de la ciencia y la tecnología y a mi hermano Erwin por enseñarme la fragilidad de la vida y aprender a apreciarla a cada minuto.

A mi amiga y compañera de tesis Brenda, por haber tenido la paciencia y la predisposición de trabajar en equipo para lograr un objetivo común.

Finalmente, todas las personas que me han apoyado y han hecho que el trabajo se realice con éxito en especial a aquellos que me abrieron las puertas y compartieron sus conocimientos conmigo, amistades como la de Talia T., Luis C, Pilar H.

Anaid Jimenez Moreano

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

DEDICATORIA

El presente trabajo de investigación está dedicado principalmente a Dios, por ser el inspirador y darnos fuerza para continuar por el camino de la vida.

En memoria de padre Feliciano y mi abuelita Genoveva, quien en vida fueron una gran fortaleza para seguir adelante, inculcándome valores, llenándome de amor, sabiduría, consejos y enseñanzas, gracias por su amor, su trabajo y su sacrificio en todos los años que me acompañaron en vida, se que desde donde se encuentren se sienten orgullosos, pensar que están felices por el anhelo que tenían de que llegara este día, para festejar este logro.

A mi madre por asumir la responsabilidad de ser padre y madre para mis hermanos y para mí, gracias por sacarnos adelante, también por tu comprensión, amor, trabajo y sacrificio, gracias a ti pude llegar a cumplir una de mis metas más anheladas, también darte las gracias por haberme apoyado incondicionalmente en cada situación difícil y por convertirme en una persona llena de valores.

A mis hermanos Freddy y Brayan por estar siempre conmigo, apoyándome incondicionalmente, dándome sabios consejos y sobre todo por ese cariño que me tienen y gracias por todas las motivaciones que me dan para seguir adelante.

A mi amiga y compañera de tesis Anaid, por haber cumplido una de nuestras metas juntas, y poder haber sido una integrante más de mi familia, gracias por tu comprensión y paciencia, sobre todo tu liderazgo para poder avanzar en equipo y terminar nuestro objetivo en común.

Brenda Alexandra Qquecho Ccachainca

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

AGRADECIMIENTO

Agradezco a Dios por bendecirnos la vida, por guiarme a lo largo de mi existencia, ser el apoyo y fortaleza en aquellos momentos de dificultad y de debilidad.

Gracias a mis padres: Claudia Moreano Casquino y Juan Jimenez Benavides, por ser los principales promotores de mis sueños, por confiar y creer en mis expectativas, por los consejos, valores y principios que me han inculcado.

Gracias a mi hermano Jürgen Jimenez Moreano mi amigo y confidente, quien me apoyo siempre que estuvo en sus manos, por cuidarme siempre.

Agradezco a nuestros docentes de la Escuela profesional de Ingeniería de Sistemas de la Universidad Andina del Cusco, por haber compartido sus conocimientos a lo largo de la preparación de nuestra profesión, de manera especial, al Ingeniero Ivan Molero Delgado por ser gran tutor y amigo en el camino de nuestro proyecto de investigación quien ha guiado con su paciencia, y su rectitud como docente nuestra investigación. También a nuestros dictaminantes Ing. Isabel Acurio e Ing. Hugo Espetia pues “Un buen instructor puede crear esperanza, encender la imaginación e inspirar amor por el aprendizaje”.

Anaid Jimenez Moreano

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

AGRADECIMIENTO

Agradecer a Dios por la vida y las bendiciones que me ha dado en mi vida, sobre todo por darme fuerzas cuando más lo necesitaba para poder seguir adelante.

Agradezco a mi mamá, a mis hermanos y a toda mi familia por estar siempre conmigo por motivarme a terminar mis metas, gracias por todos sus consejos que siempre me dan para no rendirme y seguir adelante, agradecer a mi primo Dario Qquecho por sus consejos y por haberme apoyado moralmente.

Agradecer a la señora Claudia y el señor Juan por haberme acogido como una hija más en su familia, por sus consejos y motivaciones para llegar a una de nuestras metas junto a Anaid.

A todos mis docentes de la escuela profesional de Ingeniera de Sistemas de la Universidad Andina del Cusco, por haberme brindado sus conocimientos y consejos a lo largo de nuestra preparación profesional. Especialmente a nuestro asesor Ing. Ivan Molero Delgado, por habernos guiado en este proceso de nuestro trabajo de investigación, por su paciencia y sabiduría, por apoyarnos incondicionalmente para culminar nuestra meta. También agradecer a Ing. Pilar Hidalgo León por habernos aconsejado y guiado en nuestro trabajo de investigación, a nuestros dictaminantes Ing. Isabel Acurio e Ing. Hugo Espetia por su tiempo y paciencia en este proceso del trabajo de investigación.

Agradecer a nuestros amigos que nos apoyaron incondicionalmente, por sus conocimientos y sus consejos para terminar nuestro trabajo. También nos dieron sus palabras de aliento para no rendirnos en cada obstáculo que se nos presentaba y poder cumplir nuestra meta anhelada.

Brenda Alexandra Qquecho Ccachainca

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

ÍNDICE GENERAL

DEDICATORIA 2

AGRADECIMIENTO 4

ÍNDICE GENERAL 6

ÍNDICE DE TABLAS 8

ÍNDICE DE FIGURAS 8

INTRODUCCIÓN 11

ABSTRACT 12

1. Problema de investigación 13

1.1. Ámbito de influencia 13

1.1.1. Ámbito de influencia teórica 13

1.2. Planteamiento del problema 13

1.2.1. Descripción de la situación actual del lugar de intervención 13

1.2.2. Descripción del problema 14

1.2.3. Formulación del problema 15

1.2.4. Objetivos 15

General 15

Específicos 15

1.2.6. Alcances y limitaciones 16

2. Marco Teórico 18

2.1. Antecedentes del desarrollo, implementación o transferencia tecnológica 18

2.2. Base Teórico-Científicos 24

3. Desarrollo, Implantación o Transferencias Tecnológica 50

4 Resultados 90

4.1. Comprobación de la prospectiva 90

4.1.1. Matriz de Confusión 91

4.2. Cumplimiento de objetivos 94

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

4.3. Contribuciones 96

4.4. Recursos y Presupuesto 97

GLOSARIO 98

CONCLUSIONES 102

RECOMENDACIONES 103

REFERENCIAS 105

ANEXOS 107

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

ÍNDICE DE TABLAS

Tabla 1.Definición de Usuarios para el prototipo 51
Tabla 2.Historia de Usuarios..... 52
Tabla 3.Entrenamiento de imágenes estáticas 77
Tabla 4.Entrenamiento de videos de 5 segundos a 10 segundos como máximo (Procesador el Core i5) 79
Tabla 5..Características de la computadora para la instalación del prototipo..... 86
Tabla 6.Resultados de clasificar una letra en diferentes equipos de cómputo..... 90
Tabla 7.Tabla de costos de elaboración del prototipo..... 97

ÍNDICE DE FIGURAS

Figura 2. 1. Un diagrama de Venn que describe Deep Learning. 28
Figura 2. 2. Arquitectura de una red neuronal artificial simple 29
Figura 2. 3. Support Vector Machine.....31
Figura 2. 4.Sistema de clasificación de imagen. 32
Figura 2. 5. Paradigma de realizar prototipos 36
Figura 2. 6. Proceso de metodología cualitativa de investigación 37
Figura 2. 7.Función de activación ReLU 38
Figura 2. 8.Los 21 puntos claves de la mano 39
Figura 2. 9.Las tres capas de la red de Zimmermann. 40
Figura 2.10. Arquitectura propuesta para la red PosePrior. 41
Figura 2. 11.Ejemplo del Dataset.....43
Figura 2. 12. Pseudocódigo para la obtención de la pose de la mano usando geometría de dedos 45
Figura 2. 13. Pseudocódigo para la clasificación de poses mediante SVM.....46
Figura 2. 14. Pseudocódigo para la clasificación de poses mediante Red Neuronal estándar 47
Figura 2. 15.Ajuste de Red Neuronal Estándar usando keras 48
Figura 2. 16 Diagrama de la metodología general del algoritmo de investigación desarrollada 49

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

Figura 3. 1.Interfaz del prototipo 53
Figura 3. 2.Agregando archivos entenados de las letras a la red de Zimmermann..... 57
Figura 3. 3. Pseudocódigo del algoritmo de procesamiento para correr el prototipo 58
Figura 3. 4.Dorso de la mano..... 62
Figura 3. 5.Imagen de salida con los filtros realizados..... 62
Figura 3. 6.Imagen de entrada..... 63
Figura 3. 7.Imagen de salida 63
Figura 3. 8.Reconociendo de la mano mediante video 64
Figura 3. 9.Recogimiento del contorno de la mano en video, posición correcta..... 65
Figura 3. 10.Utilizando filtro de cv2.GaussianBlur..... 66
Figura 3. 11. Imagen de salida con los filtros utilizados 66
Figura 3. 12. Resultados obtenidos para el entrenamiento y evaluación de la red de Zimmermann..... 69
Figura 3. 13.Estructura de las posiciones de la mano 70
Figura 3. 14.Posición del dedo en “NoCurva”..... 71
Figura 3. 15.Posición del dedo en “MediaCurva” 72
Figura 3. 16.Posición del dedo en “FullCurva” 72
Figura 3. 17.Posición del dedo “HaciaArriba” 73
Figura 3. 18.Posición del dedo “HaciaAbajo”..... 73
Figura 3. 19.Posición de dedo “ApuntaIzquierda” 74
Figura 3. 20.Posición del dedo “ApuntaDerecha”..... 74
Figura 3. 21.Posición del dedo “EsquinaDerecha”..... 75
Figura 3. 22.Posición del dedo “EsquinaIzquierda”..... 75
Figura 3. 23.Posición del dedo “EsquinaInfIzquierda” 76
Figura 3. 24.Posición del dedo “EsquinaInfDerecha” 76
Figura 3. 25.Obtención de los .csv como dataset para la fase II..... 81
Figura 3. 26.Procesamiento de los .csv..... 81
Figura 3. 27.Procesamiento de los .csv..... 82
Figura 3. 28.Procesamiento de los .csv..... 82
Figura 3. 29.Código para determinar número de tiempo para capturar los frames 83
Figura 3. 30.Pruebas en tiempo real del reconocimiento de las letras del abecedario..... 83
Figura 3. 31.Obtención de los frames de la captura del video de 6 segundos 84
Figura 3. 32.Procesamiento de los frames para la detección de la seña 84
Figura 3. 33.Resultado del procesamiento del video 85

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

Figura 3. 34. Instalador de Python 3.6.8 86
Figura 3. 35. Instalación de Python 3.6.8 en Windows 87
Figura 3. 36. Verificación de la version de Python instalado 87
Figura 3. 37. Bibliotecas requeridas para el funcionamiento del prototipo 88
Figura 3. 38. Línea de comandos para ejecutar prototipo con SVM 89
Figura 3. 39. Icono de prototipo..... 89
Figura 4. 1. Accuracy por letra y accuracy overall…… ……91
Figura 4. 2. Matriz de confusión de resultados 92
Figura 4. 3.Cuadro de Test versus Entrenamiento con overfitting 93
Figura 4. 4.Cuadro de Test versus entrenamiento, resuelto el ajuste de overfitting 93

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

INTRODUCCIÓN

La investigación se enfoca en el desarrollo de un prototipo que permita reconocer una señal básica (cualquier letra estática del abecedario de lenguaje de señas peruano) capturada por webcam basándose en la lengua de señas del Perú dentro de un tiempo corto estimado.

El prototipo utiliza la tecnología de Machine Learning usando una Convolutional Neural Network (CNN), en español llamada red neuronal convolucional de tres capas y Support Vector Machine (SVM) para la segunda fase; La primera capa de la CNN definida como "HandSegNet" ubica y aísla a la mano a detectar, la segunda capa definida como "PoseNet" que usa algoritmos de detección de puntos clave obtenidos de una imagen 2D RGB y la tercera capa deriva estos puntos base más el punto de vista de la cámara, considerando iluminación, perspectiva y giro a un plano 3D obtenidos a partir del paper "Learning to Estimate 3D Hand Pose from Single RGB Images" de Christian Zimmermann & Thomas Brox investigadores de la universidad de Freiburg en Alemania.

La limitante de la red de Zimmermann se reduce a entregar coordenadas de los dedos de la mano 3D, para lo cual se desarrolló una etapa que pueda definir las poses del abecedario peruano usando umbrales de confianza, por ángulos y rizo, a la salida de estas coordenadas relativas usa la máquina de vectores de soporte (SVM), también la compara con una red neuronal estándar usando como función de activación la regresión lineal, finalmente escribe la letra sobre la seña correspondiente definida en el algoritmo.

ACAT es un sistema desarrollado por S. Hawking, que tras su muerte se volvió de código abierto, este sistema fue nuestra inspiración, buscamos desarrollar un prototipo para ayudar de la misma forma a las personas con discapacidad auditiva del CEBE Don José de San Martin, así los niños con discapacidad auditiva (D.A) sean capaces de transmitir algunos pensamientos clave mediante las letras del abecedario estáticas, usando solo una webcam común y corriente a comparativa de sistemas ya hechos que usan cámaras complejas y de alto costo como la Kinect.

A un trabajo futuro se espera una mejora considerable del prototipo y que los niños puedan comunicarse con sus amigos o familiares de manera libre y feliz como funciona un traductor actual de idiomas.

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

ABSTRACT

The thesis focuses on the development of a prototype that allows us to recognize a basic and static signal of a peruvian signal language captured by webcam within a short estimated time.

The prototype uses Machine Learning as main technology; First phase uses three-layer Convolutional Neural Network (CNN) and for the second phase uses Support Vector Machine (SVM).

The first layer of the CNN defined as “HandSegNet” that locates and isolates the hand to be detected for the webcam, the second layer defined as “PoseNet” that uses key point detection algorithms obtained from a 2D RGB image and positioning to a 3D plane and the third layer derives these points base to the camera's point of view, this part represents the image’s depth, considering lighting, perspective and rotation to a 3D plane obtained from the paper “Learning to Estimate 3D Hand Pose from Single RGB Images” by Christian Zimmermann & Thomas Brox researchers from the University of Freiburg in Germany that has free access for use.

The limitation of the Zimmermann network is reduced to delivering coordinates of the 3D planeo f fingers of the hand, we créate a Class to define the poses of the Peruvian alphabet using confidence thresholds, by angles and curls, at the exit of these Relative coordinates we use the support vector machine (SVM), finally write the letter over the corresponding sign defined in the Class.

ACAT is a system developed by S. Hawking, which after his death became open source software, the system was our inspiration, we sought to develop the prototype to help people with hearing disabilities from CEBE Don José de San Martin for children with hearing impairment (DA) , so they’ll be able to transmit some key thoughts through the static letters of the alphabet, using only an ordinary webcam to comparison of ready-made systems that use complex cameras like the Kinect.

In a future job, we consider a considerable improvement of the prototype, so children could communicate with their friends or family freely and happily as a current language translator Works with the system working over AWS service.

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

1. Problema de Investigación

1.1. Ámbito de influencia

1.1.1. Ámbito de influencia teórica

Área de Dominio

Según dominio de la Escuela Profesional de Ingeniería de Sistemas de la Universidad Andina del Cusco, se encuentra en el área de tecnologías de información.

Línea de Investigación

Ciencia de Datos: Al desarrollar el prototipo de traductor de lengua de señas hacemos uso de la tecnología de "Machine Learning", perteneciendo actualmente a la ciencia de datos, pues usaremos datos, estadísticas, metodología científica, ingeniería de datos, computación avanzada y demás características que se encuentran definidas en la Data Science.

1.2. Planteamiento del problema

1.2.1. Descripción de la situación actual del lugar de intervención

El tema de investigación se desarrolla principalmente para el CEBE (Centro de Educación Básica Especial) Don José de San Martín ubicado en Urb Ttio Mz. D-3 en el distrito de Wanchaq, cuenta con 42 años al servicio de la comunidad, liderado por su Director Helmer del Pozo Cruz, cuyo objetivo principal se basa en la construcción de una sociedad inclusiva y solidaria para las personas con discapacidad, especialmente para los niños, anualmente atienden a más de 210 estudiantes con diversas discapacidades, cuentan con docentes calificados en las especialidades de discapacidad intelectual, discapacidad auditiva y multidiscapacidad.

Nuestro tema de investigación se concentrará con aquellos estudiantes con dificultad auditiva teniendo aproximadamente una población de 28 alumnos desde inicial hasta sexto de primaria los cuales se encuentran en dos salones (niños de tres a siete años aproximadamente), el primer salón incluye inicial hasta segundo grado de primaria a cargo de la docente Mery Rodriguez Huaman y el segundo salón incluye tercer grado de primaria hasta sexto de primaria a cargo de la docente María del Carmen Celis Gómez.

Asimismo, para la comunidad de ciudadanos con dificultad auditiva del Departamento, Región Cusco.

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

1.2.2. Descripción del problema

La Lengua de señas en el Perú con ISO 639-3 PRL es una lengua reconocida por parte del estado peruano, y como tal posee una sintaxis, gramática y léxico. Actualmente está reconocida por el estado mediante la LEY 29535, se registra según el censo nacional del 2017 más de diez mil usuarios aproximadamente de lengua de señas peruanas.

El problema principal sobre el ámbito tecnológico - informático, es que actualmente en la ciudad del Cusco, no se cuenta con una herramienta de comunicación tecnológica para la comprensión de los niños y sus familiares cercanos, la realidad que afronta el CEBE Don José de San Martin.

La tecnología actual existente para la comunicación entre personas comunes y personas con discapacidad auditiva es muy limitada, una persona que desee comunicarse con una persona que tiene discapacidad auditiva (D.A), tiene que aprender necesariamente la lengua de señas peruano.

La población de persona con discapacidad auditiva (D.A) viene en aumento, pero el número de centros como el CEBE es limitado por lo que un apoyo informático sería de gran ayuda, un sistema que facilite la comunicación, evitando que personas con discapacidad auditiva (D.A) se enfrenten a un mundo lleno de dificultades y barreras de comunicación efectiva.

Por tal motivo el propósito de este proyecto es desarrollar un prototipo que refuerce la comunicación entre una persona que no aprendió y no quiere aprender Lenguaje de señas peruano (LSP); por lo que sería de valiosa ayuda contar con un prototipo que les permita reforzar las ideas que desean transmitir y que logre comunicarse y poder integrarse en un entorno de oyentes.

En el 2018 según la INEI, la dificultad de oír en todo el Perú es del 7.6 % (232176 personas) y en el Cusco es del 8.4 % (11226 personas).

El total de personas en el Perú que tienen dificultad para hablar o comunicarse es del 3.1 % (93088 personas), en el Cusco es del 3% (4011 personas).

La población según lengua materna aprendida en la niñez es de 24 624 personas que no escuchan ni hablan y 10447 que aprendieron en su niñez como lengua materna la lengua de señas peruanas. (Instituto Nacional de Estadística e Informática-INEI, 2018)

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

Según encuestas realizadas (Ver anexo N°6 y N°7) en el CEBE, a las docentes principales de los salones de niños con discapacidad auditiva (D.A) obtuvimos las siguientes problemáticas:

De todos los niños del primer salón de inicial, ningún padre de familia o familiar cercano sabe lengua de señas, a comparativa del otro salón donde solamente un padre de familia puede comunicarse con su hijo usando lengua de señas.

El curso de lengua de señas se imparte en el CEBE cada 2 meses de marzo a diciembre, pero lamentablemente los padres o familiares no pueden asistir por diferentes motivos.

En algunos municipios de la ciudad del Cusco se imparte un curso breve de lengua de señas, pero tampoco se le da la debida importancia por la sociedad en general.

En la mayoría de los casos, según nuestras encuestas, para aprender la lengua de señas se necesita mucho tiempo, convivir con la persona con discapacidad auditiva y la dificultad del idioma.

Viendo la problemática en general, observamos que no son los cursos de capacitación o el niño, sino la falta de comprensión de las personas, el tiempo que toma aprenderlo o la dificultad de la lengua, puede ser también la falta de sensibilización y desconocimiento por parte de la población como otros factores.

1.2.3. Formulación del problema

¿Cómo desarrollar un prototipo de traductor de lengua de señas peruanas básicas estáticas utilizando machine learning?

1.2.4. Objetivos

General

Desarrollar un prototipo de traductor de lengua de señas peruanas básicas estáticas utilizando machine learning.

Específicos

a) Obtener los requerimientos principales para la construcción del prototipo.
b) Realizar el diseño rápido del prototipo.
c) Construir prototipo inicial.
d) Evaluar el prototipo en función al usuario.
e) Refinar el prototipo
f) Implementar el prototipo en el CEBE.

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

1.2.5. Justificación

Con el desarrollo del prototipo, lograremos fortificar y ampliar la poca comunicación que existe entre los padres y los niños con dificultad auditiva, rompiendo las brechas sociales que tienen con sus padres o familiares, fomentará la inclusión de las personas con discapacidad auditiva dentro del CEBE y con otras personas oyentes que los rodeen.

El prototipo conectara a las personas con discapacidad auditiva (D.A) a un entorno externo donde una persona que no tiene mínima idea de lengua de señas pueda entenderle al menos de una forma básica en tiempo real y atender alguna urgencia.

El prototipo también tiene el propósito de fomentar tecnología nueva en la ciudad del Cusco, desarrollando nuevas tecnologías como es “Machine Learning”.

Esta propuesta de prototipo ayudará a incrementar las habilidades tecnológicas de las personas con dificultad auditiva, familiares y personas afines; Además en el CEBE Don José de San Martin se implementará esta tecnología nueva hacia los estudiantes y docentes.

En conclusión, con este prototipo ayudaremos y facilitaremos la comunicación entre los niños con dificultad auditiva, sus familiares y otros de su entorno más cercano, siendo un sistema sin fines de lucro.

1.2.6. Alcances y limitaciones

Alcances

El trabajo de investigación consta de la implementación del prototipo funcional de un traductor de lengua de señas básicas estáticas peruanas, sistema local que se plantea implementar en el CEBE Don José de San Martin, cuyo objetivo principal es el reconocimiento del abecedario estático, usando una colección de frames tomados por la webcam en un “tiempo real” sin uso de algún aditamento electrónico especial como un Kinect.

Utilizaremos principalmente la tecnología de Machine Learning con un algoritmo de red neuronal Convolucional y máquina de vectores de soporte (VSM -Vector Support Machine), la red neuronal Convolucional consta de tres capas (red neuronal de Zimmermann), con un arreglo de filtros en la segunda capa y creación de un algoritmo de poses y rizos de la mano realizada por las tesistas (la llamaremos geometría de dedos).

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

Limitaciones

El prototipo se desarrolla para el CEBE (Centro de Educación Básica Especializado) Don José de San Martín, exclusivamente para los dos salones donde se enseñan la lengua de señas, la primera aula cuenta con 28 alumnos desde inicial hasta sexto de primaria, el segundo salón cuenta con niños de 3 a 7 años aproximadamente; En la primera aula se enseña desde el nivel inicial hasta segundo grado de primaria, el segundo salón incluye tercer grado de primaria hasta sexto grado de primaria, salones con los que se trabaja y donde se implementaría el sistema.

El prototipo no incluye las señas peruanas dinámicas como la “J”, “Ñ” y la “Z”, porque eso implicaría hacer un algoritmo adicional para el reconocimiento de la variación del movimiento diferenciado de un frame a otro y el algoritmo mencionado en el trabajo de investigación realiza el procesamiento de un frame estático haciendo el cálculo de profundidad de la imagen, reconocimiento de contornos y una estimación de pendientes, ángulos definidos para la seña estática, el desarrollar un algoritmo para la señas dinámica implicaría una confusión en el procesamiento del algoritmo realizado en el trabajo de información.

Ya que el trabajo de investigación es prototipo no podemos adquirir un PaaS (Platform as a Service) o como subir nuestro aplicativo sobre Amazon Web Services (AWS) pues el costo del servicio es muy elevado para la elaboración de este trabajo.

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

2. Marco Teórico

2.1. Antecedentes del desarrollo, implementación o transferencia tecnológica

2.1.1 Antecedentes Nacionales

a) CLASIFICACIÓN Y RECONOCIMIENTO DE GESTOS ESTÁTICOS DE LA MANO BASADO EN EL ALFABETO DACTILOLÓGICO DE LA LENGUA DE SEÑAS DEL PERÚ APLICANDO REDES PROFUNDAS BAJO CARACTERÍSTICAS INVARIANTES - UNIVERSIDAD NACIONAL DE SAN ANTONIO ABAD DEL CUSCO – 2017 - ESCUELA PROFESIONAL DE INGENIERÍA INFORMÁTICA Y DE SISTEMAS-, JOSÉ LUIS FLORES CAMPANA

Se afirma que:

La investigación que realizaron está enfocada en clasificar y reconocer los 24 gestos estáticos de la mano basados en el alfabeto dactilológico de la lengua de señas del Perú (LSP), no hizo uso de los gestos dinámicos debido a que el enfoque es el uso de las redes profundas para imágenes. El objetivo de la investigación que desarrollaron se basó en comparar una arquitectura de una Convolutional Neural Network (CNN) y otra de Stacked Denoising Autoencoder (SDAE) para clasificar el conjunto de 24 gestos estáticos de la mano de la LSP obtenidos de una base de datos local desarrollada (No especifica detalles de dataset). Para el desarrollo del proyecto de investigación usaron un tipo de investigación explorativa, teórica y aplicativa. Propusieron un algoritmo efectivo para detectar la región que contiene el gesto de la mano, el algoritmo incorpora una etapa de pre procesamiento y segmentación. Compararon los resultados de precisión y error obtenido entre las dos arquitecturas de red profunda diseñadas (CNN y SDAE) con técnicas usuales de extracción y clasificación de machine learning.

Comentario:

La investigación que realizaron cuenta con una base de datos de imágenes de prueba estáticas introducidas manualmente y la salida de resultados se queda en tan solo una apreciación del algoritmo descrito. Los gestos no se clasifican adecuadamente ante invariaciones de escala, rotación y traslación, debido a que

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

procesa posturas y puntos a nivel 2D, no es robusta ante ruido y cambios de iluminación.

Se enfoca ante la comparativa de las redes profundas y compara los resultados de precisión y error obtenido entre las dos arquitecturas de red profunda diseñadas (CNN y SDAE).

b) ANÁLISIS DE MÉTODOS DE VISIÓN COMPUTACIONAL Y MACHINE LEARNING PARA LA CLASIFICACIÓN DE IMÁGENES DE VARIEDADES DE PAPA NATIVA - UNIVERSIDAD NACIONAL DE SAN ANTONIO ABAD DEL CUSCO – 2018 - ESCUELA PROFESIONAL DE INGENIERÍA INFORMÁTICA Y DE SISTEMAS – YENNIFER ZULMA & CARLA DORIS CARDOSO

Se afirma que:

En la Universidad Nacional de San Antonio Abad del Cusco (Calvo & Cardoso, 2018) realizaron un análisis de métodos de Visión Computacional y Machine Learning para la clasificación de imágenes de variedades de papa, la clasificación de la papa se realiza manualmente, por lo cual las tesistas proponen un extractor que contempla dichas características, donde se obtienen características de color en los espacios RGB, CMYK, HSV, HSI, YCbCr, YIQ, LUV y Colores Oponentes. Dentro de los métodos de Maching Learning utilizan los clasificadores Support Vector Machine y Random Forest, donde llegaron a la conclusión que el mejor método para la clasificación de la papa fue Random Forest que obtuvo mayor tasa de aciertos.

Comentario:

La investigación que realizaron nos ofrece una visión clara sobre la comparativo al usar un clasificador tipo Random Forest versus uno de Support Vector Machine (SVM), claramente el reconocimiento de una papa es diferente al de una mano, pero gracias a los datos que obtuvieron nos impulsaron a realizar el trabajo de investigación a continuación tomando en consideración el clasificador tipo SVM, debido a que Random Forest permite detectar mediante textura una imagen por ser invariantes a la rotación, traslación y escalamiento, sin perder la información de esta, mientras que SVM hace lo contrario, considerando tanto la rotación, traslación y escalamiento, adecuado para nuestro trabajo de investigación.

También tomamos en consideración aspectos del paper (Sharma et al., 2016) y del paper (Gavali & Banu, 2019) , donde se puede ver unas tablas comparativa de

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

mejores técnicas de clasificadores para imágenes, además de beneficios y limitantes en el anexo n° 65.

c) DIAGNÓSTICO AUTOMÁTICO DE ROYA AMARILLA EN HOJAS DE CAFETO APLICANDO TÉCNICAS DE PROCESAMIENTO DE IMÁGENES Y APRENDIZAJE DE MÁQUINA– PONTÍFICE UNIVERSIDAD CATÓLICA DEL PERÚ – 2018 – FACULTAD DE CIENCIAS E INGENIERÍA – INGENIERÍA INFORMÁTICA – ALFONSO CARLOS BARRIGA POZADA & CARLOS ARRASCO ORDOÑEZ

Se afirma que:

La Roya Amarilla. La Roya Amarilla se propaga fácilmente a través del aire, una vez que cae en un cultivo de café, ataca directamente en las hojas, almacenándose en forma de esporas en el envés de estas, y al paso de días consume las hojas hasta defoliar completamente la planta infectada.

Propone una solución a través del aprendizaje máquina y procesamiento de imágenes, con el fin de automatizar el proceso de detección de la Roya en las hojas y calcular de manera más precisa la severidad del hongo. El proceso comienza en tomar fotografías a las hojas en un espacio semi controlado (con fondo blanco), luego se guardan todas las imágenes de las que se quiera conocer el porcentaje de severidad y ejecutar el programa propuesto, al término de ello el software muestra un reporte estadístico con el grado de incidencia por hoja según la clasificación de severidad que corresponda. Finalmente, destacar que, de manera funcional, el aprendizaje máquina será vital para descartar si hay presencia de roya en la hoja analizada, y luego si la hoja está infectada, con el método de procesamiento de imágenes se calculará de manera más precisa el porcentaje de severidad considerando el área de la hoja examinada.

Comentario:

La investigación que hicieron utiliza igualmente SVM para la clasificación de la hoja infectada, las imágenes de las hojas podrían asemejarse a las rugosidades que se tiene en la palma de la mano, se observa detalles de la calibración de contraste y segmentación de manchas.

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

d) DISEÑO DE UN GUANTE ELECTRÓNICO PARA EL MAPEO Y RECONOCIMIENTO DE GESTOS UTILIZANDO REDES NEURONALES - 2017 - FACULTAD DE INGENIERÍA - 2017 - FACULTAD DE CIENCIAS E INGENIERÍA - INGENIERÍA ELECTRÓNICA - DULANTO RAMOS & LUIS ENRIQUE

Se afirma que:

La investigación que realizaron tiene como objetivo diseñar un sistema de reconocimiento de gestos manuales que traduzca gestos a necesidades básicas de la persona, con el fin de ayudar a personas de la tercera edad que padecen problemas del habla. Dicho sistema de reconocimiento de gestos es aplicado en un guante electrónico que extrae señales de la mano. Para el desarrollo de la tesis, se hace uso de dos ejes de modelamiento: modelamiento directo y modelamiento inverso. Primero, se modela por análisis paramétrico (modelamiento directo) a una mano antropomórfica que responde a los movimientos del guante electrónico; luego, se aplica identificación de sistemas por red neuronal mediante algoritmos de retro propagación a los modelos obtenidos por análisis paramétrico, con el fin de utilizar estos modelos basados en redes neuronales en el sistema de reconocimiento de gestos. Este segundo paso es el modelamiento inverso y la razón de su aplicación se fundamenta en el hecho de que se desea obtener un sistema que posea tolerancia a fallas, la cual es una propiedad de las redes neuronales. Finalmente, se diseña un núcleo de reconocimiento de gestos, el cual reconoce patrones en la data resultante de las redes neuronales. Dicho núcleo de reconocimiento de gestos es también una red neuronal y, a diferencia de otros tipos de sistemas de reconocimiento de patrones, tiene la capacidad de aprender a partir de data experimental y generar soluciones en base a lo aprendido.

Comentario:

El sistema que implementaron usa datos de sensores del guante electrónico, por lo que sus sistemas de redes neuronales trabajan con otro tipo de data trabajada a nivel analógico y transformado en datos digitales, el tratamiento es diferente al de un sistema de reconocimiento de imágenes informático.

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

2.1.2 Antecedentes Internacionales

a) RECONOCIMIENTO DE GESTOS DINÁMICOS TESINA - UNIVERSIDAD NACIONAL DE LA PLATA ARGENTINA – 2014 - FACULTAD DE INFORMÁTICA - FACUNDO MANUEL QUIROGA

Se afirma que:

En la Universidad Nacional de la Plata Argentina, (Facundo Manuel, 2014), proyecto de investigación de tesina, definieron un modelo de gestos a reconocer, generaron una base de datos de prueba con gestos llamados LNGH, basados en máquinas de vectores de soporte (SVM), redes neuronales feedfoward (FF) y redes neuronales competitivas (CPN), donde utilizaron representaciones locales y globales para caracterizar los gestos. Donde los gestos a reconocer son movimientos de la mano, con invariancia a la velocidad, la rotación, la escala y la traslación. La captura de la información se realizó mediante el dispositivo Kinect y su SDK correspondiente, que reconoce las partes del cuerpo y determina sus posiciones en tiempo real.

Comentario:

La investigación realizada en la Universidad de la Plata hace uso de herramientas tecnológicas como la Kinect y nos muestra un claro ejemplo de los algoritmos de clasificación usados, demostrando en esta investigación que una de las más eficientes para correr nuestro algoritmo en la fase II sería la máquina de vectores de soporte (SVM).

b) DISEÑO DE UN SISTEMA DE RECONOCIMIENTO DE GESTOS NO MÓVILES MEDIANTE EL PROCESAMIENTO DIGITAL DE IMÁGENES-TESIS-UNIVERSIDAD AUTÓNOMA DEL CARIBE - 2014 - ESCUELA PROFESIONAL INGENIERÍA MECATRÓNICA- VALERIA VALENCIA & BETSY VILLA

Se afirma que:

En la Universidad Autónoma del Caribe,(Valencia & Villa, 2014), desarrollaron e implementaron un sistema de reconocimiento de gestos no móviles mediante el entorno de Matlab, por medio del cual se visualiza la imagen de la letra adquirida, junto con la traducción en el lenguaje de señas colombiano, el sistema se desarrolló para reconocer 20 letras no móviles del lenguaje de señas colombiano, obteniendo

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

un promedio de error del 23% concluyendo así que dicha aplicación resulto ser eficaz para la traducción de letras adquiridas con la mano derecha utilizando un guante quirúrgico, como método de parametrización de las imágenes de entrada capturadas.

c) FREE-HAND GESTURE RECOGNITION WITH 3D-CNNS FOR IN-CAR INFOTAINMENT CONTROL IN REAL TIME - 2017 – SACHARA F., KOPINSKI T., GEPPERTH A. & HANDMANN U.

Se afirma que:

Muestra un enfoque novedoso para transformar datos de sensores de time-of-flight (ToF) para que sean interpretables por redes neuronales convolucionales (CNN). Como ToF los datos tienden a ser demasiado ruidosos dependiendo de varios factores, como como iluminación, coeficiente de reflexión y distancia, la necesidad de se hace evidente un enfoque algorítmico robusto. Al abarcar una cuadrícula tridimensional de tamaño fijo alrededor de cada nube de puntos que son capaces de transformar la entrada tridimensional para que sea procesable por CNN. La metodología demuestra que las CNN son realmente capaces de extraer la información relevante y aprende un conjunto de filtros, lo que les permite diferenciar un conjunto complejo de diez gestos diferentes obtenidos de 20 individuos diferentes y que contiene 600.000 muestras en general. La validación cruzada de 20 veces muestra la generalización del rendimiento de la red, logrando una precisión de hasta el 98,5% en conjuntos de validación que comprenden 20.000 muestras de datos. La aplicabilidad en tiempo real del sistema se demuestra mediante una validación interactiva en un sistema de infoentretenimiento que funciona con hasta 40 fps en un iPad en el interior del vehículo.

d) SKELETON-BASED DYNAMIC HAND GESTURE RECOGNITION – 2016 - QUENTIN DE SMEDT, HAZEM WANNOUS & JEAN PHILIPPE VANDEBORRE

Se afirma que:

En este artículo, propone un nuevo enfoque basado en esqueletos para el reconocimiento de gestos de la mano en 3D. En concreto, explota la forma geométrica de la mano para extraer un descriptor efectivo de las articulaciones conectadas del esqueleto de la mano devueltas por la Intel RealSense depth camera (sensor parecido al Kinect), luego cada descriptor es codificado por una

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

representación de Fisher Vectorial obtenida usando un modelo de mezcla gaussiana. Una representación multinivel de Vectores Fisher y otras características geométricas basadas en esqueletos, está garantizado por una pirámide temporal para obtener el vector de características final, utilizado posteriormente para lograr la clasificación mediante un clasificador SVM lineal.

Evalúa en un desafío conjunto de datos de gestos de la mano que contiene 14 gestos, realizados por 20 participantes realizando el mismo gesto con dos números de dedos diferentes. Los resultados experimentales muestran que el enfoque basado en esqueletos logra consistentemente rendimiento sobre un enfoque basado en la profundidad.

2.2. Base Teórico-Científicos

2.2.1. Lengua de Señas

Según la Ley N° 29535, ley que otorga el reconocimiento oficial a la lengua de señas peruana, esta ley tiene como propósito de otorgar reconocimiento oficial y regular la lengua de señas peruana como una lengua natural de las personas con discapacidad auditiva en todo el territorio nacional, esta disposición no afecta la libre elección del sistema que desee utilizar la persona con discapacidad auditiva para comunicarse en su vida cotidiana.(*LEY N° 29535 - Norma Legal Diario Oficial El Peruano*, 2010)

Es un lenguaje que se percibe a través de la vista donde requiere el uso de las manos como articuladores activos, e uso del espacio como lugar de articulación (estructura fonológica) y referencia temporal, el cual es un sistema de comunicación. Para el aprendizaje de la lengua de señas las personas deben ejercitar la atención, la percepción y la memoria visual, así como la agilidad manual, uso del espacio, así como la expresión facial y la expresión corporal.

Se deben considerar:

a) Aspectos Visuales: La atención visual, la discriminación visual y la memoria visual.
b) Aspectos gestuales: La expresión corporal, la expresión facial y la motricidad digital y manual.

Donde estos aspectos ayudan a desarrollar habilidades para discriminar visualmente las señas y tener memoria visual para identificarlas. (Perú. Ministerio de Educación.

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

Dirección General de Servicios Educativos Especializados. Dirección de Educación Básica Especial, 2015).

2.2.2. Audición

La audición es muy importante en la vida de las personas desde que nacen, el sentido de la audición es uno de los canales de recepción sensorial con la que cuenta cada persona. La audición es el medio por donde los niños y niñas entran en contacto con el mundo que los rodea, que se va integrando poco a poco al código lingüístico compuesto por diferentes sonidos y estructura propia que constituyen el lenguaje. La pérdida de la función auditiva, ya sea total o parcial, mínima se va ver reflejada en el desarrollo del lenguaje de la persona, sobre todo en el aprendizaje y la relación con su entorno. (Perú. Ministerio de Educación. Dirección General de Servicios Educativos Especializados. Dirección de Educación Básica Especial, 2013)

Discapacidad Auditiva. Se da por la causa de la deficiencia auditiva, donde se presenta la disminución en la capacidad para poder oír, entender y escuchar, sobre todo los sonidos del lenguaje hablado, con la misma habilidad con que lo realiza una persona sin deficiencia auditiva. Se denomina persona sorda porque tiene deficiencia auditiva importante que afecta la adquisición del lenguaje hablado, existen deficiencias unilaterales que se da en un solo oído, no se presentan necesariamente alteraciones en el lenguaje.

La discapacidad auditiva no es igual en todas las personas; se puede dar desde los problemas leves de audición a deficiencias severas o sordera total. Los niños con deficiencia auditiva, no todos tienen el mismo grado de discapacidad, esto va a depender de la intensidad de la deficiencia auditiva, donde en función de esta pérdida existirá mayor o menor capacidad para poder identificar sonidos y desarrollar el lenguaje oral. (Perú. Ministerio de Educación. Dirección General de Servicios Educativos Especializados. Dirección de Educación Básica Especial, 2013)

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

2.2.3. CEBE

CEBE (Centro de Educación Básica Especial) brinda atención con un enfoque inclusivo a estudiantes con necesidades especiales asociadas a discapacidad severa y multidiscapacidad, que, por la naturaleza de sus limitaciones, no pueden ser atendidos en las instituciones inclusivas de otras modalidades y formas de educación donde los estudiantes desarrollan sus potenciales, en un ambiente flexible, apropiado y no restrictivo, mejorando sus posibilidades para lograr una calidad de vida con miras a su inclusión social, educativa, familiar y laboral. (Ministerio de Educación Perú, 2018).

2.2.4. Machine Learning

Machine Learning es una forma de Inteligencia Artificial (IA) que permite a un sistema aprender de los datos en lugar de a través de la programación explícita. Sin embargo, el aprendizaje automático no es un proceso simple.

Machine Learning utiliza una variedad de algoritmos que aprenden iterativamente de datos para mejorar, describir datos y predecir resultados. A medida que los algoritmos ingieren datos de entrenamiento, es posible producir modelos más precisos basados en esos datos. Un modelo de aprendizaje automático es el resultado generado cuando entrena su algoritmo de aprendizaje automático con datos. Después del entrenamiento, cuando proporciona un modelo con una entrada, se le dará una salida. Por ejemplo, un algoritmo predictivo creará un modelo predictivo. Luego, cuando proporcione datos del modelo predictivo, recibirá una predicción basada en los datos que entrenaron al modelo. El aprendizaje automático es ahora esencial para crear modelos analíticos. (Hurwitz & Kirsch, 2018).

En el proceso de toma de datos Machine Learning nos ayuda a extraer ciertas características para calificar o describir la imagen como el color, la textura y la forma del objeto.

Tipos de Machine Learning:

a) Aprendizaje supervisado

   Un set de entrenamiento con ejemplos y respuestas correctas o targets, son provistos. Basados en este set de entrenamiento, el algoritmo generaliza la respuesta correcta para las posibles entradas también llamado entrenamiento por ejemplos.

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

b) Aprendizaje no supervisado

Las respuestas correctas no son provistas, en este caso el algoritmo identifica similitudes entre las entradas, entonces las salidas son categorizadas por estas cosas en común. En el enfoque estadístico se usa para saber la estimación de densidad.

c) Aprendizaje reforzado

Este define un punto entre aprendizaje supervisado y no supervisado. El algoritmo dice cuando la respuesta es incorrecta, pero no te dice como seria correcta, también llamado aprendizaje con crítica, porque el monitor calcula por scores la respuesta correcta pero no sugiere mejoras.

d) Aprendizaje evolutivo

La evolución biológica se ve como un proceso evolutivo, los organismos se adaptan para mejorar y así sobrevivir, este mismo modelo se adapta a este tipo de algoritmo que corresponde a poner scores en cuanto a cuan buena es la solución. (Hand, 2010)

Deep Learning. Deep Learning es un método de representación-aprendizaje con múltiples niveles de representación, obtenida componiendo módulos simples, pero no lineales que cada uno transforma la representación en un nivel, comenzando con la entrada sin formato, en una representación a un nivel más alto, un poco más abstracto.

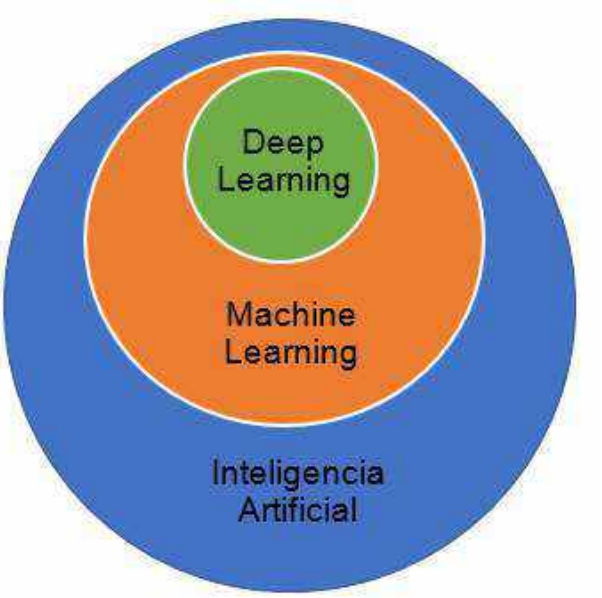
La clave de Deep Learning es que las capas no están diseñadas por ingenieros humanos, sino que se aprenden de los datos mediante un procedimiento de aprendizaje de propósito general, además Deep Learning es un subcampo Machine Learning, donde Machine Learning es un subcampo de la Inteligencia Artificial. (Rosebrock, 2017).

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

Figura 2. 1.

Un diagrama de Venn que describe Deep Learning.



Nota. Como un subcampo del Machine Learning que se encuentra en un subcampo de la Inteligencia Artificial. Fuente:(Rosebrock, 2017)

¿Cuántas capas necesitan una red neuronal para ser considerada deep?

Según el Dr. Adrian Rosebrock en su libro "Deep Learning for Computer Vision with Python" necesita tener al menos dos capas para ser considerado como deep en una CNN.

Red Neuronal. Llamadas también redes neuronales artificiales (ANNs – Artificial Neural Networks), no se refiere a modelos reales de las redes neuronales cerebrales del ser humano, sino más a cómo funcionan estas al momento de reconocer algo abstracto como una cara o una imagen, ¿cómo es que nuestro cerebro logra realizar estas distinciones?, las redes neuronales vienen de McCulloch y Pitts en 1943, este fue el primer clasificador binario (llamado "Binary Classifier") capaz de reconocer dos diferentes categorías basados en una entrada.

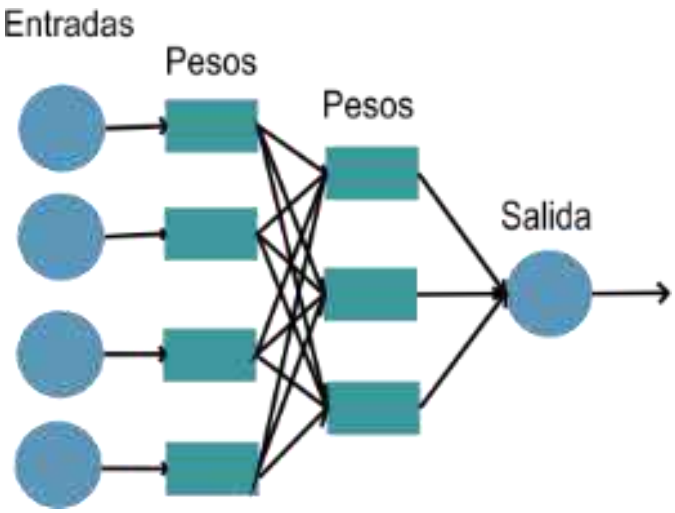
Después de muchas investigaciones y experimentos finalmente en el año 1986 el algoritmo de propagación regresiva por Werbos, Rumelhart y LeCun habilitaba las famosas "multi capas de alimentación (multilayer feedforward)" para que la red pudiera ser entrenada y aprenda de los errores.

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

Figura 2. 2.

Arquitectura de una red neuronal artificial simple



Nota. Las entradas entran a la red, cada conexión lleva la señal a través de dos capas ocultas en la red, la función final calcula la etiqueta de la clase de salida. Fuente: Propia

Las redes neuronales se usan para el aprendizaje de tareas supervisadas, no supervisadas y semisupervisadas. Desafortunadamente, las redes neuronales estándar no obtienen una alta precisión de clasificación cuando se trabaja con conjuntos de datos de imágenes desafiantes que exhiben variaciones en la traslación, rotación, punto de vista, etc. Para obtener una precisión razonable en estos conjuntos de datos, necesitaremos trabajar con un tipo especial de redes neuronales de retroalimentación llamadas Redes neuronales convolucionales (CNN). (Rosebrock, 2017).

La forma en la que una red neuronal artificial aprende se basa en la modificación de valores o pesos de las conexiones, para asi lograr las salidas deseadas por el usuario. Las redes neuronales artificiales pueden clasificarse de acuerdo con el tipo de aprendizaje que utilizan. (Mercado Polo et al., 2015)

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

Red Neuronal Convolucional. Llamadas CNNs (Convolutional Neural Networks), hoy en día son consideradas los clasificadores de imagen más poderosos y responsables de darle impulso a la rama de Computer Visión un subcampo del Machine Learning, las redes neuronales convolucionales comparadas con otras tienen mayor poder para la clasificación de imágenes debido a las convoluciones o filtros que usa en cada capa y el comportamiento como trabaja para clasificar estas.

Estos filtros o "convoluciones" pueden ser difuminados, extracción de contornos, suavizados, etc. En términos de aprendizaje profundo la convolución es una multiplicación por elementos de dos matrices seguido de una suma. (Rosebrock, 2017)

Support Vector Machine (SVM). Clasificador de precisión alta comparada con Regresión logística y los árboles de decisión, genera un hiperplano que mejor divida el conjunto de datos en clase, los vectores de soporte son los puntos de datos más cercanos al hiperplano, estos puntos definen de mejor manera la línea de separación de clase a clase calculando los márgenes.

El algoritmo supervisado segrega el conjunto de datos con mayor cantidad de márgenes posible entre clases y vectores de soporte, generando diferentes tipos de hiperplanos, pero selecciona el que mejor separe los datos.

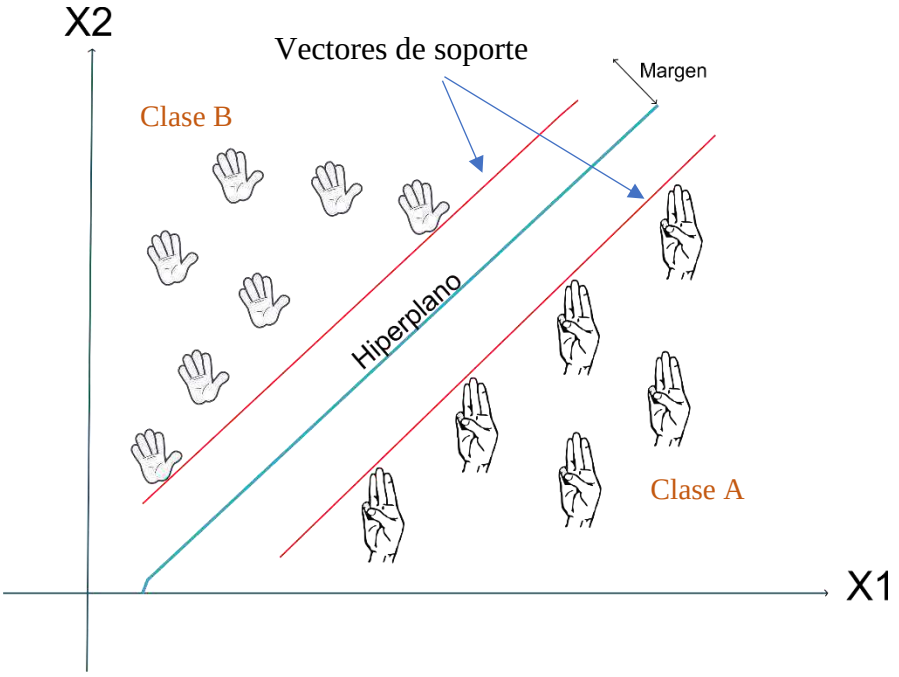
La máquina de vectores de soporte (SVM) es lo primero que usan muchos investigadores de optimización en Machine Learning debido a su formulación clásica como un programa cuadrático convexo.

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

Figura 2. 3.

Support Vector Machine



Nota. Perspectiva del algoritmo (Fuente propia)

2.2.5. Numpy

Numpy es una librería fundamental usada en la ciencia computacional que trabaja con Python, esta contiene un potente array N-dimensional para objetos, funciones de broadcasting, herramientas para la integración con C/C++ y Fortran y hace uso del algebra lineal, las transformadas de Fourier para la discretización de las señales y tiene una capacidad muy alta de números aleatorios.

Además de su uso con características muy similares a las que cuenta Matlab, esta es de código abierto y de uso libre, las grandes capacidades científicas ofrecen ser un óptimo contenedor de datos de diferentes tipos, esto permite a Numpy integrarse con gran variedad de bases de datos. (Rosebrock, 2017).

Imagen como un Numpy Array. Si bien es cierto que conocemos el concepto de pixel, o color basado en RGB, en este caso Numpy como librería define una imagen en los siguientes parámetros altura, ancho y profundidad (height, width, depth), usado para ser procesador por OpenCV como una representación scikit-image en RGB, donde la profundidad es definida por el número de canales que usa; Ejemplo: Una

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

imagen de 300x200 pixeles donde 300 es el ancho o width, 200 el heigth o altura y 3 canales de profundidad por ser una imagen RGB (canales rojo, verde y azul), cabe aclarar que OpenCV usa este modelo de imagen pero invertido llamado BGR.

Las variaciones que considera la imagen para ser trabajadas en un clasificador son de escala, deformación, oclusión, iluminación, desorden de fondo, variación intraclase, punto de vista y deformación.

Figura 2. 4.

Sistema de clasificación de imagen.

Nota. Para ser trabajado en un clasificador o una CNN. Fuente:(Rosebrock, 2017)

2.2.6 OpenCv

OpenCV (Open Source Computer Vision Library) es una biblioteca de software de visión de computadora y de aprendizaje automático de código abierto. OpenCV fue construido para proporcionar una infraestructura común para las aplicaciones de visión artificial y para acelerar el uso de la percepción de la máquina en los productos comerciales. Al ser un producto con licencia BSD, OpenCV facilita que las empresas utilicen y modifiquen el código.

La biblioteca cuenta con más de 2500 algoritmos optimizados, que incluyen un conjunto completo de algoritmos de aprendizaje por ordenador y de aprendizaje por ordenador tanto clásicos como de vanguardia. Estos algoritmos se pueden usar para detectar y reconocer rostros, identificar objetos, clasificar acciones humanas en videos, rastrear movimientos de cámara, rastrear objetos en movimiento, extraer modelos 3D de objetos, producir nubes de puntos 3D desde cámaras estéreo, unir

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

imágenes para producir una alta resolución imagen de una escena completa, encuentre imágenes similares de una base de datos de imágenes, elimine los ojos rojos de las imágenes tomadas con flash, siga los movimientos de los ojos, reconozca paisajes y establezca marcadores para superponerlos con realidad aumentada, etc. (OpenCV team, 2019).

2.2.7 Tensorflow

Tensorflow y Theano son librerías de definición algo abstracta, con propósito general de uso en la visión computarizada, usando Deep learning en su procesamiento, podría compararse como frameworks para Deep Learning, siendo una librería de gran alcance. (Rosebrock, 2017)

2.2.8 Matplotlib

Matplotlib es una libreria para la generación de gráficos a partir de datos contenidos en listas o arrays y su extensión matemática NumPy. (*Matplotlib: Python plotting — Matplotlib 3.3.0 documentation*, s. f.)

2.2.9 Scikit – learn

Es una biblioteca para aprendizaje automático de software libre para el lenguaje de programación Python.1 incluye varios algoritmos de clasificación, regresión y análisis de grupos entre los cuales están máquinas de vectores de soporte, bosques aleatorios, Gradient boosting, K-means y DBSCAN. Está diseñada para interoperar con las bibliotecas numéricas y científicas NumPy y SciPy.(*Scikit-learn: machine learning in Python — scikit-learn 0.23.1 documentation*, s. f.)

2.2.10 SciPy

SciPy es una biblioteca libre y de código abierto para Python. Se compone de herramientas y algoritmos matemáticos. Se creó a partir de la colección original de Travis Oliphant, que se componía de módulos de extensión para Python y fue lanzada en 1999 bajo el nombre de Multipack, llamada así por los paquetes netlib que reunían a ODEPACK, QUADPACK, y MINPACK.(*SciPy.org — SciPy.org*, s. f.)

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

2.2.11 Pillow

Pillow es una librería que permite la edición de imágenes directamente desde Python en formatos GIF, JPEG y PNG y otro.(*Pillow — Pillow (PIL Fork) 7.2.0 documentation*, s. f.)

2.2.12 PyTesseract

¿Qué es tesseract? es un potente motor de OCR de uso libre. Si bien en sus orígenes era software propietario desarrollado por Hewlett Packard, tras años sin evolución, en 2005 terminó siendo liberado el código. Actualmente es desarrollado por Google.(Lee, s. f.)

2.2.13 Cmake

cmake-python-distributions fue desarrollado inicialmente en septiembre de 2016 por Jean-Christophe Fillion-Robin para facilitar la distribución del proyecto usando scikit-build y dependiendo de CMake.(*Documentation | CMake*, s. f.)

2.2.14 Keras

Keras a comparativa de TensorFlow, si es un framework de Deep learning que provee un API muy bien desarrollado, que facilita la construcción de redes neuronales profundas (Deep neuronal networks). (Rosebrock, 2017).

2.2.15 Paradigma del Prototipo

Utilizaremos en esta investigación el “paradigma del prototipo tipo parches”, según un tipo de investigación cualitativa, pero antes una breve introducción del paradigma para su mejor entendimiento:

El ciclo de vida del desarrollo de sistemas (SDLC) es una metodología de siete fases para el análisis y diseño de sistemas, la cual sostiene que los sistemas se desarrollan mejor mediante el uso de un ciclo específico de actividades de analistas y usuarios.

La construcción de un prototipo es un proceso rápido e interactivo entre los usuarios, analistas para crear y refinar las partes del nuevo sistema. Se puede emplear como parte del ciclo de vida del desarrollo de sistemas para determinar los requerimientos, o como una alternativa al ciclo de vida del desarrollo de sistemas (SDLC). La información que se recolecta en la fase de un prototipo permite al analista establecer

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

prioridades y redirigir los planes sin sufrir consecuencias graves. (Kendall & Kendall, 2011)

El prototipo de software se puede utilizar de varias maneras en un proceso de software las cuales son:

a) Se utiliza en el proceso de ingeniería de requerimientos, donde el prototipo puede ayudar en la obtención y validación de los requerimientos del sistema.
b) Se utiliza en el proceso de diseño del sistema, donde el prototipo sirve para explorar soluciones de software particulares y apoyar el diseño de las interfaces de usuario.
c) Se utiliza en el proceso de pruebas, el prototipo sirve para realizar pruebas back-to-back con el sistema que se entregara al cliente.

Tipos de prototipos

Existen cuatro tipos de prototipos:

a) Prototipo de parches: consiste en la construcción de un sistema funcional, parchado o construido totalmente con partes. Este tipo de prototipo es un modelo funcional, con todas las características necesarias pero que es ineficiente. En esta instancia del prototipo los usuarios pueden interactuar con el sistema, acostumbrarse a la interfaz y a los tipos de salidas disponibles
b) Prototipo no operacional: Es un modelo a escala no funcional, utilizado para probar ciertos aspectos del diseño. Sería conveniente un modelo de escala no funcional para un sistema de información cuyas aplicaciones requieran una codificación demasiado extensa como para incluirla en el prototipo. Se debe considerar la idea de entrada y de salida únicamente en el sistema para realizar el prototipo.
c) Prototipo primero de una serie: Es la creación de un modelo a escala completa de un sistema, a lo que comúnmente se le conoce como piloto. El modelo funcional permite a los usuarios experimentar una iteración realista con el nuevo sistema, al tiempo que minimiza el costo de solucionar problemas que se presentan
d) Prototipo de características selectas: Es la creación de un modelo operacional que incluya solo algunas características del sistema final. Los prototipos de esta forma es posible que incluyan solo algunas características esenciales, los

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

sistemas se desarrollan en módulos, estos modelos de prototipo forman parte del sistema actual.

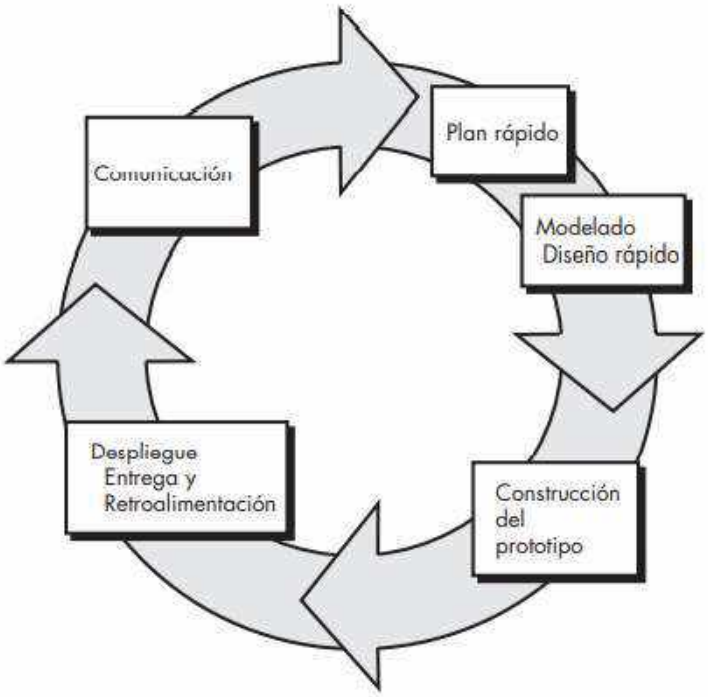
(Kendall & Kendall, 2011)

Para la construcción de un prototipo las etapas son:

a) Comunicación: Consiste en la reunión con los otros usuarios para definir los objetivos generales del software y definir los requerimientos.
b) Plan rápido: Se planifica rápidamente una iteración para hacer el prototipo.
c) Modelado: Consiste en la representación de aquellos aspectos del software que serán visibles para los usuarios finales.
d) Diseño rápido: Lleva a la construcción de un prototipo, aquí se trata de dar una idea general al usuario.
e) Construcción del Prototipo: En esta etapa se diseña el prototipo real basado en la información recolectada en el diseño rápido.
f) Despliegue entrega y Retroalimentación: Se entrega y es evaluado por los participantes, que dan una retroalimentación para mejorar los requerimientos.

Figura 2. 5.

Paradigma de realizar prototipos



Fuente:(Pressman, 2010)

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

Los beneficios de usar el prototipo rápido son:

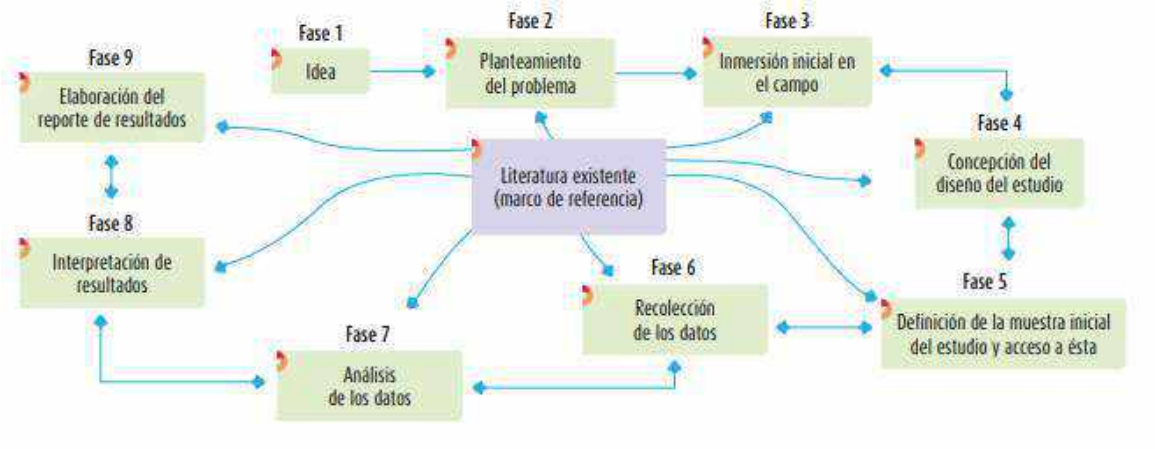
a) Mejora en la usabilidad del sistema.
b) Genera concordancia entre el sistema y las necesidades del usuario.
c) Mejora la calidad del diseño.
d) Mejora en el mantenimiento.
e) Reduce el esfuerzo de desarrollo.
f) La creación rápida de los prototipos es demostrar las posibilidades rápidamente mediante la creación de una serie de maquetas, para que el diseño sea eficaz, donde cumplan con las expectativas de los requisitos de los usuarios.

2.2.16 Sobre la metodología de investigación

Nuestra investigación se basa sobre el enfoque cualitativo definido como: *"El enfoque cualitativo utiliza la recolección de datos sin medición numérica para descubrir o afinar preguntas de investigación en el proceso de interpretación."*(Hernández Sampieri et al., 2014)

Figura 2. 6.

Proceso de metodología cualitativa de investigación



Fuente:Hernández Sampieri et al., 2014)

Mientras que el enfoque cuantitativo se ocupa de describir, explicar y predecir los fenómenos, generando y probando teorías, el enfoque cualitativo permite describir, comprender e interpretar los fenómenos a través de las percepciones y significados producidos por las experiencias de los participantes, en este caso definidos por las historias de usuarios.

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

2.2.17 PyCharm

PyCharm es un entorno de desarrollo más completo para Python. Es parte del suite de herramientas de programación ofrecida por JetBrains.(«PyCharm», 2018)

2.2.18 Git

Git, es un software de control de versiones inventado por Linus Torvalds, fue creado pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones, nos proporciona las herramientas para desarrollar un trabajo en equipo de manera rápida y eficaz.(*Qué es Git*, s. f.)

2.2.19 Función de Activación

Una función de activación asigna valores de un dominio a un rango, ejemplo de rango y dominio:

$$Y = \text{sen}(x) \rightarrow f(x)$$

Donde el rango iría de 1 a -1 y el dominio serio, todos los números reales (R), las funciones también pueden variar entre ser lineales o no lineales.

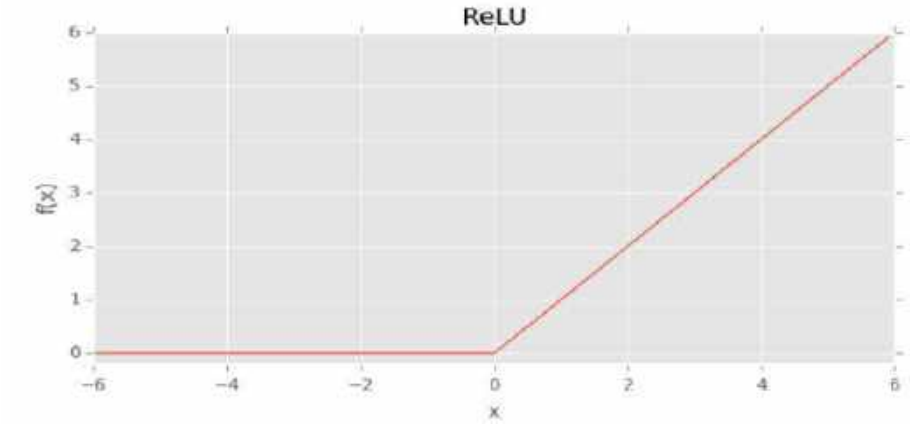
Las funciones de activación en redes neuronales, suelen ser funciones no lineales que ayudan a clasificar muestras no lineales; Existen diferentes funciones de activación según propósito que se requiera.

Para una CNN la función de activación más adecuada suele ser ReLU, definida como:

$$\text{ReLU}(X) = 0 \text{ si } X = 0$$

Figura 2. 7.

Función de activación ReLU



Fuente: Rosebrock, 2017

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

Existen variaciones en ReLU para mejoras y optimizaciones en redes neuronales convolucionales:

- Leaky ReLU
- parametric rectified linear (PReLU)
- randomized rectified linear (RReLU)

(Xu et al., 2015)

2.2.20 Red Neuronal de Zimmermann y Thromas Brox

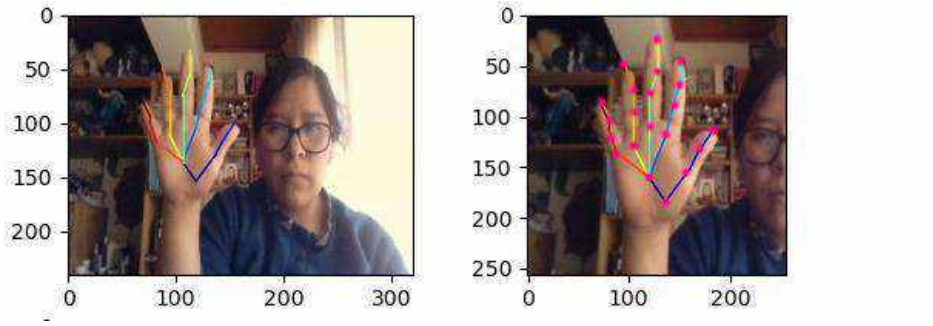
La red Neuronal de Zimmerman y Thomas Brox nos ofrece la solución al dilema de la estimación de la pose de la mano en un plano 3D a partir de imágenes 2D (a partir de una imagen .jpg), sin la necesidad de usar algún equipo en especial como un Kinect u otro.

El enfoque global de la Red de Zimmermann y Thomas Brox (desde este punto la llamaremos Red de Zimmermann) consiste en tres capas profundas que cubren subtareas importantes que se pueden observar en la figura 2.9.

a) La primera capa segmenta y localiza la mano en la imagen.
b) La segunda capa localiza los puntos clave en las imágenes 2D de la mano ya segmentada, estos puntos son 21 en total (como se ve en la figura 2.8, puntos en color rosado) comparado con el algoritmo de Landmark que define 68 puntos clave en el rostro; La clave de esta capa, es la profundidad de la imagen 2D que puede ofrecer o definir un espacio 3D usando un mapa de profundidad.

Figura 2. 8.

Los 21 puntos claves de la mano



Fuente: Propia

DERECHOS DE AUTOR RESERVADOS

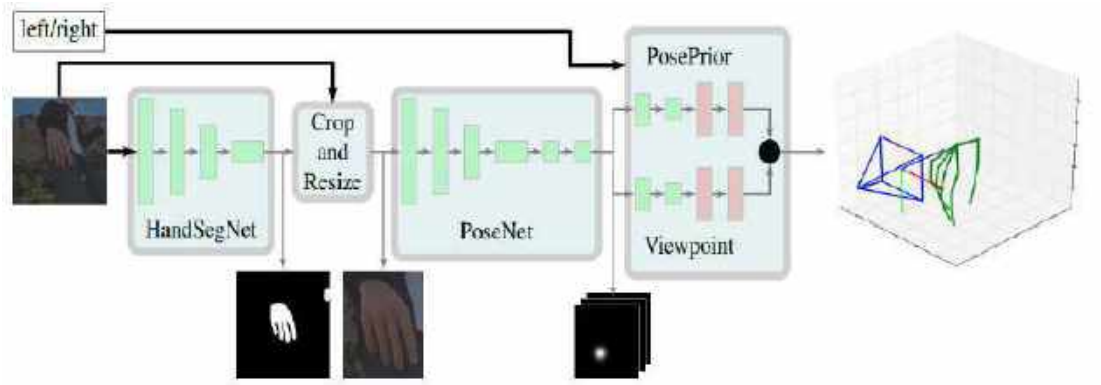
Universidad Andina del Cusco
Repositorio Digital

c) Finalmente, la tercera capa define los puntos como vectores en un plano tridimensional (3D), cogiendo los puntos clave 2D de la segunda capa obtenidos en el procedimiento anterior, combinados con la profundidad de cámara.

Debemos aclarar que esta red desarrollada por Zimmermann consta de una licencia de uso libre para estudios e investigaciones (“GNU Lesser General Public License”).

Figura 2. 9.

Las tres capas de la red de Zimmermann.



Nota. La primera capa localiza la mano dentro de la imagen mediante una red de segmentación (HandSegNet). De acuerdo con la máscara de mano definida por contornos, la imagen de entrada se recorta y sirve como entrada a PoseNet, que es la segunda capa.. Esta localiza un conjunto de puntos clave de la mano representados como mapas de puntuación. Posteriormente, la red PosePrior estima la más probable Estructura 3D condicionada a los mapas de score o puntuacion. Fuente:(Zimmermann & Brox, 2017)

Hace uso del poder discriminativo de los enfoques actuales de Redes Neuronales Convolucionales (CNN) y luego intentan elevar el conjunto de detecciones 2D al espacio 3D.

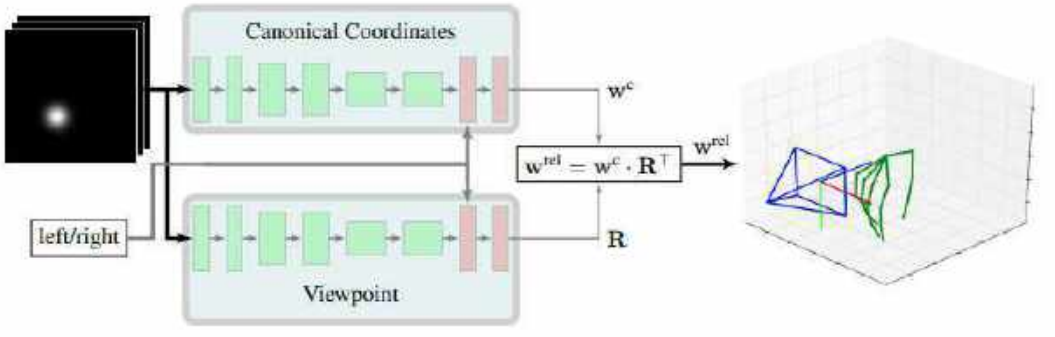
En la estimación de pose manual, la pose en 3D se recupera resolviendo un problema de optimización cinemática inversa como se puede observar en la figura 2.10.

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

Figura 2.10.

Arquitectura propuesta para la red PosePrior.



Nota. Dos corrientes casi simétricas estiman las coordenadas canónicas y el punto de vista relativo a este sistema de coordenadas. La combinación de las dos predicciones produce una estimación de la relativa coordenada normalizadas w^{rel}. Fuente:(Zimmermann & Brox, 2017)

Para la representación del conjunto de coordenadas relativas, donde $w_i = (x_i, y_i, z_i)$ describe las coordenadas de los puntos clave J en el espacio 3D, es de decir i ∈ $[1, J]$ con J=21 para este caso.

Con respecto al punto de vista podría considerarse como el peso que tiene las coordenadas canónicas de salida multiplicadas por la matriz de rotación, dándonos como resultado una estimación aproximada 3D de la pose de la mano en coordenadas relativas (W^{rel}), tomando en consideración la rotación para la mano derecha e izquierda, definida como R^T multiplicadas por las coordenadas canónicas definidas como W^c.

Para realizar la representación de la pose de la mano el algoritmo de Zimmermann trabaja de la siguiente manera:

- El problema de inferir un vector observado a nivel 2D en un plano 3D, tiene ambigüedades, para ello infiere una estructura 3D invariante de escala entrenando una red para estimar una coordenada ya normalizada, en esta fase llegaron a la conclusión que el punto clave en el medio de la palma de la mano era el más estable.
- Con respecto a la segmentación de la mano, despliega una arquitectura, que inicializa detectando a la persona y la representa como un mapa de score (score map), obtiene la posición central de la persona, hace el recorte de la misma forma desde la posición central de la mano para hacer el recorte, este depende mucho

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

de la articulación, ellos usan como una versión menor del algoritmo de red de Wei et al. (Para mayor detalle de esta, revisar el anexo n° 62.a).

- Cada mapa de score contiene la información acerca de la probabilidad de que cierto punto clave este representado en una ubicación espacial. Para esto usa una arquitectura del tipo encoder-decoder parecida al de "Pose Network" definido por Wei et al. (Para mayor detalle de esta, revisar el anexo n° 62.a).
- Para calcular la matriz de rotación (sobre plano 3D), el algoritmo lo realiza en 2 pasos:
    - Busca la rotación con respecto a los ejes X y Z, después es alineado con el eje Y en el frame de coordenadas canónicas, el cálculo lo realiza tanto para la mano derecha como para la izquierda.
    - Al estimar la transformación R de rotación, equivale a predecir el punto de vista dado de una muestra con respecto a un frame canonico, y lo llama como "estimación del punto de vista".
- La red de Zimmermann usa una serie de 6 convoluciones con ReLU (ReLU non-linearities – Rectified Linear Unit como función de activación para la red).
- Para el entrenamiento de la red de HandSegNet, se aplicó softmax como función de clasificación con entropía cruzada. (Para mayor detalle de esta, revisar el anexo n° 62.a).

El DataSet. El DataSet que se usó contiene 41258 muestras entrenadas y 2728 muestras de prueba. Cada muestra contiene imágenes con las siguientes características:

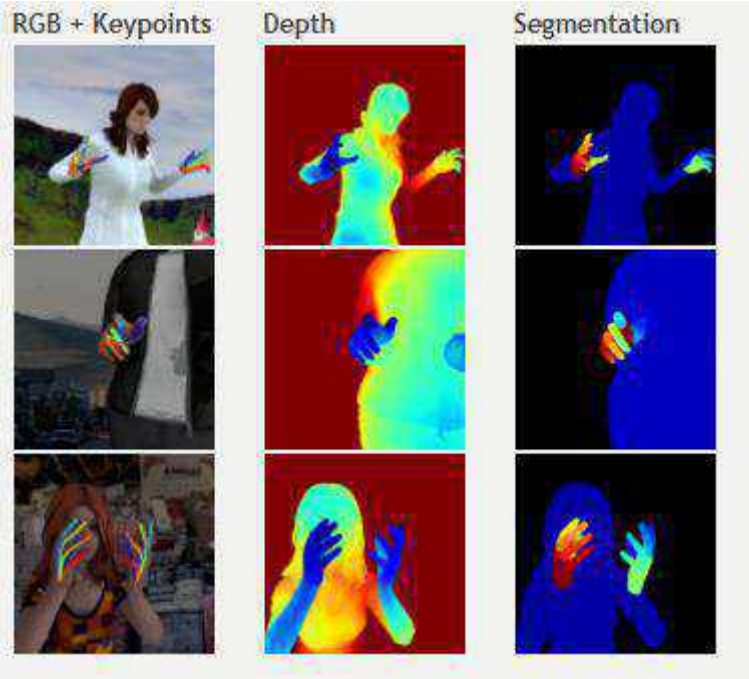
a) Imágenes RGB (320x320 pixeles)
b) Imágenes con mapa de profundidad extraída (320x320 pixeles)
c) Imágenes con mascara de segmentación (fondo, persona, indicadores para palma y dedos)
d) 21 puntos clave para cada mano con sus coordenadas canónicas (cuatro puntos por dedo y uno en el medio de muñeca)
e) Matriz de cámara intrínseca K.

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

Figura 2. 11

.Ejemplo del Dataset



Nota. Obtención de la imagen con mapa de segmentación y profundidad, usa 4 puntos clave por dedo. Fuente:(Zimmermann & Brox, 2017)

2.2.21 Metodología general del algoritmo de investigación desarrollado

Claramente el objetivo de la investigación es el desarrollo del prototipo de traductor de lenguaje de señas peruanas y no el de la explicación del funcionamiento de la red de Zimmermann, por lo que se enfocó más en el desarrollo del algoritmo para el reconocimiento de las señas peruanas, haciendo uso de la red de Zimmermann para la obtención de las coordenadas de los puntos clave o vectores de los dedos sobre el plano 3D, también se implementó una mejora en la capa de segmentación de la mano comparando diferentes filtros tipo gausiano (Gaussian) para el difuminado y de umbralizacion (Threshold) para contoneados más precisos.

Por último, esta salida se procesa mediante una máquina de vectores de soporte, también se hace la comparativa con el algoritmo de geometría de dedos y con una red neuronal artificial estándar; Para definir la geometría de dedos, se definió cada tipo de dedo y orientación, cada umbral de rizo y orientación va de 0.0 – 1.0.

La red en la fase II se entrena con videos de 7 a 10 segundos de duración aproximadamente para la red neuronal estándar, consecutivamente hace una comparativa

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

de letra a letra, dando como salida la información de la letra con mayor peso (esto varia de 0.0 – 10.0), el resultado es una colección de frames procesados que se unen para formar un video en formato .mp4 de salida para el usuario final.

En la fase II el algoritmo puede clasificar la pose usando una red neuronal simple, la máquina de vectores de soporte o el algoritmo de geometría de dedos directamente, los resultados se reflejan mejor en la tabla n°4.1, tanto la red neuronal como la máquina de vectores de soporte usan para su funcionamiento la geometría de dedos.

A continuación, se detalla el Pseudocódigo de la fase II del algoritmo de la geometría de dedos, de la red neuronal y de SVM.

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

a) Pseudocódigo de la geometría de dedos en la fase II:

Figura 2. 12.

Pseudocódigo para la obtención de la pose de la mano usando geometría de dedos

```
Algoritmo:  Determinacion de la pose definido para cada letra
            del abecedario estatico de LSP.

Entrada:    -Uso de la clase "CurvaDeDedo" del archivo CurvaDedo.py, "PosicionDedo" del
archivo
            PosicionDedo.py y la clase "DataFormacionDedo" del archivo
            FormaciondedatosDedos.py.
            -Frames capturados por la webcam durante 8 segundos, pudiendo este tiempo ser
            modificable,
            almacenados temporalmente en la carpeta "test_data".
Salida:     Conjunto de caracteristicas de cada pose.

Metodo determinar_posicion (posicciones_curvatura,posiciones_dedo,posiciones_dedo_conocidas,
min_umbral)
Metodo get_nombrePosicion_id (pose_id, pose_dedo)

Funcion principal crear_poses_conocidasDedos()
    poses_dedo_conocidas <- []

    #Configuracion que se debe realizar para cada dedo variando valores de cruvatura y
    posiciones_dedo
    letra_definida <- DataFormacionDedo()
    letra_definida.nombre_posicion = "Nombre de la letra"
    letra_definida.curva_posicion <- [
        #puede obtener una o muchas curvaturas, conviene declarar solamente 1 o 2
        [CurvaDeDedo.Tipo_de_curva], #tipo de curva para el dedo pulgar
        [CurvaDeDedo.Tipo_de_curva], #tipo de curva para el dedo indice
        [CurvaDeDedo.Tipo_de_curva], #tipo de curva para el dedo medio
        [CurvaDeDedo.Tipo_de_curva], #tipo de curva para el dedo anular
        [CurvaDeDedo.Tipo_de_curva]  #tipo de curva para el dedo meñique
    ]

    #umbral por cada tipo de curvatura definido
    letra_definida.curva_posicion_determinacion <- [
    [x.0], #este valor puede ir de 0.0 - 1.0
    [x.0], #este valor puede ir de 0.0 - 1.0
    [x.0], #este valor puede ir de 0.0 - 1.0
    [x.0], #este valor puede ir de 0.0 - 1.0
    [x.0]  #este valor puede ir de 0.0 - 1.0
    ]

    #en este punto define la posiciones_dedo
    letra_definida.dedo_posicion <- [
    #puede obtener una o muchas posiciones, conviene declarar solamente 2
    [posicionDedo.Tipo_de_Posicion, posicionDedo.Tipo_de_Posicion,], #tipo de posicion para
    el dedo pulgar
    [posicionDedo.Tipo_de_Posicion, posicionDedo.Tipo_de_Posicion,], #tipo de posicion para
    el dedo indice
    [posicionDedo.Tipo_de_Posicion, posicionDedo.Tipo_de_Posicion,], #tipo de posicion para
    el dedo medio
    [posicionDedo.Tipo_de_Posicion, posicionDedo.Tipo_de_Posicion,], #tipo de posicion para
    el dedo anular
    [posicionDedo.Tipo_de_Posicion, posicionDedo.Tipo_de_Posicion,]  #tipo de posicion para
    el dedo meñique
    ]

    #umbral por cada tipo de posicion definido
    letra_definida.determinar_posicion_dedo <-[
    [x.0], #este valor puede ir de 0.0 - 1.0
    [x.0], #este valor puede ir de 0.0 - 1.0
    [x.0], #este valor puede ir de 0.0 - 1.0
    [x.0], #este valor puede ir de 0.0 - 1.0
    [x.0]  #este valor puede ir de 0.0 - 1.0
    ]

    #Aqui declara que id va enlazada a esta posicion
    letra_definida.posicion_id = x #este valor puede variar de 0 - 23
    poses_dedo_conocidas.adiciona_a(letra_definida)

    retorna poses_dedo_conocidas
```

Nota. Aquí define la posición de cada dedo, así como su curvatura y rizo

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

b) Pseudocódigo de SVM fase II:

Figura 2. 13.

Pseudocódigo para la clasificación de poses mediante SVM

```
Algoritmo:  Determinacion de la pose usando SVM definido para cada letra
            del abecedario estatico de LSP (se halla en entrenamiento/SVM.py)

Entrada:    -Usa la lectura de files mediante un metodo de captura de los archivos y la
cantidad de entradas
            permitidas por tipo, definido como leer_data().
Salida:     Conjunto de caracteristicas de cada pose.


Metodo_declaracion_de_argumentos:
    -archivos .csv, delimitados por comas,
    -ruta de salida,
    -radio de entrenamiento del test,
    -numero maximo de muestras por clase permitido.

metodo_de_clasificacion_usando_SVM (data_entrenamiento, data_de_test, salida)
    x_entrenamiento, y_entrenamiento <- data_entrenamiento
    x_test, y_test <- data_de_test

    #Realizacion del entrenamiento por SVM, definiendo un kernel tipo lineal por default
    #se usa la libreria de sklearn para SVM, se sigue pasos como sklearn declara para hacer uso
    de su algoritmo de SVM
    variablea <-svm.SVC()
    variablea.fit(x_entrenamiento,y_entrenamiento)

    y_test_prediccion <- variablea.predice(x_entrenamiento)
    test_accuracy <- saca_score_accuracy(y_entrenamiento,y_test_prediccion)

    #creacion del archivo svc.pickle como data de entrenamiento de salida para su uso posterior.
    crear svc.pickle, 'wb'


Modulo principal
    //define argumentos
    args <- parse_args()

    //declaracion de archivos
    data_archivos <- ruta_absoluta[para variable_p en argumentos_de_csv.definidospor(',')]

    //data de entrenamiento y de test
    x_data,y_data <- leer_data(archivos,argumentos)
    data_entrenamiento, data_de_test <- separacion_De_data

    //si no menciona el folder de salida, entonces crea uno nuevo y limpia y crea otro para
    siguiente iteracion
    salida <-carpeta_checkpoint

    //corre modulo de clasificacion por svm usando sklearn
    metodo_de_clasificacion_usando_SVM(data_entrenamiento,data_de_test,salida)
```

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco

Repositorio Digital

c) Pseudocódigo de la red neuronal fase II:

Figura 2. 14.

Pseudocódigo para la clasificación de poses mediante Red Neuronal estándar

```
Algoritmo:  Determinacion de la pose usando Red Neuronal (Perceptron Multicapa) definido
para cada letra
            del abecedario estatico de LSP (se halla en entrenamiento/RedNeuronal.py)

Entrada:    -Uso la lectura de files mediante un metodo de captura de los archivos y la
cantidad de entradas
            permitidas por tipo, definido como leer_data().
Salida:     Conjunto de caracteristicas de cada pose.

Define N_puntosClave <- 63

Metodo declaracion_de_argumentos:
    -archivos .csv, delimitados por comas,
    -ruta de salida,
    -Prueba de test dividida,
    -Rango de aprendizaje,
    -Cantidad de epocas para entrenar la data,
    -Tamaño del batch,
    -Numero máximo de muestras,
    -Punto de cumplimiento del entrenamiento

Metodo definicion_funcion_De_activacion()
    ##Funcion de activación ReLU

Metodo Red_clasificacion_pose(X, n_clases)
    //peso,capa y parcialidad de la capa 1 de la red
    //peso,capa y parcialidad de la capa 2 de la red

Metodo de_evaluacion(X_data, Y_data,accuracy,x,y,tamaño_batch)
    suma <- 0
    Inicio para
        fin <- offset + tamaño batch
        batch_x, batch_y <- X_data[Desde offset hasta fin], Y_data[Desde offset hasta fin]

        Sesion_de_evaluacion <- get_sesion_por_default
        accuracy <- Sesion_de_evaluacion.inicia()
        suma <- accuracy* longitud(batch_y)
    fin para

    retorna suma/longitud(batch_y)


Metodo de_construccion_Y_entrenamiento_red(data_entrenamiento, data_de_test, parametros,
salida)
    x_entrenamiento, y_entrenamiento <- data_entrenamiento
    x_test, y_test <- data_de_test

    //genera grafico con Tensorflow usando los puntosclave (N_puntosClave) con X & Y

    Metodo Red_clasificacion_pose(X,parametros[n_clases])

    Obtencion de la perdida()
    Optimizacion()
    Obtencion del Accuracy o precision()
    Guardar modelo()
    Establece tamaño del batch()
    batch_tamaño <- parametros [batch]

        Inicia Sesion
            sesion.inicia()
            accuracy_De_entrenamiento <- []
            accuracy_De_test <- []

            Inicia progresbar()

            accuracy_entrenamiento <-
            de_evaluacion(X_entrenamiento,accuracy,y_entrenamiento,X,y,tamaño_batch)
            accuracy_test <- de_evaluacion(x_test,y_test,X,y,accuracy,tamaño_batch)

            accuracy_De_entrenamiento.agregarAlArray(accuracy_entrenamiento)
            accuracy_De_test.agregarAlArray(accuracy_test)

            si accuracy_test >= parametros
                rompa el entrenamiento()

            crear carpeta de modelo guardado llamado checkpoint()
            Dibuje: grafica de entrenamiento VS test()
        fin Sesion


Metodo obtiene_valor()
    //obtiene la sesion con tensorflow

Metodo para_Crar_Y_exportar_grafico()
    //Genera un archivo final llamado 'graph.pb'


Modulo principal
    //define argumentos
    args <- parse_args()

    archivos <- ruta_De_los_Archivos_con_extension_CSV
    X_data, Y_data <- leer data(archivos,numero maximo de muestras)
    data_entrenamiento, data_de_test <- Dividir(X_data,Y_data,prueba de test dividida)

    //Agrega parametros
    parametros <- (tamaño_batch, epocas, radio_aprendizaje, n_clases, rompe_entrenamiento_en)

    //Guarda la salida en un folder llamado "Checkpoint", y si no existe lo crea

    //ejecuta construccion y entrenamiento de la Red
    de_construccion_Y_entrenamiento_red(data_entrenamiento,data_de_test,parametros,salida)

    //Exporta grafico creado
```

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

Ajuste de código en Red Neuronal Estándar usando keras:

En este bloque se detalla un ajuste para el mejor funcionamiento, en correlación con la optimización del código, se actualizo Tensorflow a la versión 2.2, para usar la librería de Keras y reducir tamaño en código, además de generar resultados más rápido. A continuación, se detalla el ajuste:

Figura 2. 15.

Ajuste de Red Neuronal Estándar usando keras

```
Algoritmo: Determinacion de la pose usando Red Neuronal standard o perceptron
           multicapa con Keras, version tensorflow 2.2, actualizacion de librerias

Entrada:    Carga las combinaciones del archivo .csv detallados previamente
        usa: Keras.models (sequential), keras.layers.core (Dense) como
        librerias de arramnque para la opptimizacion de codigo

Salida: Conjunto de caracteristicas de cada pose.

Define N_puntosClave <- 63

Metodo de declaracion de variables
    -(El mismo detallado previamente en el bloque anterior)

Definicion de funciones de uso, funcion de activacion()
Red_clasificatoria_pose(),
Evalucaion()
(
Aqui aplica  model.evaluate() de keras optimizando el proceso previo
detallado en el bloque anterior para esta funcion.
),
Metodo de contruccion de ted y entrenamiento(),
Metodo de obtencion de valores(),
Metodo de creacion de grafico(),

Crea un modelo secuencial con RELU (Aqui entra la optimizacion de codigo
implementada con la libreria de Keras yel update de tensorflow 2.2)
```

Tras construir el video de salida, vacía las carpetas de almacenamiento para los frames de entrada capturados por la webcam y los frames procesados en la carpeta de salida. (Ver metodología del algoritmo final en la figura 2.9).

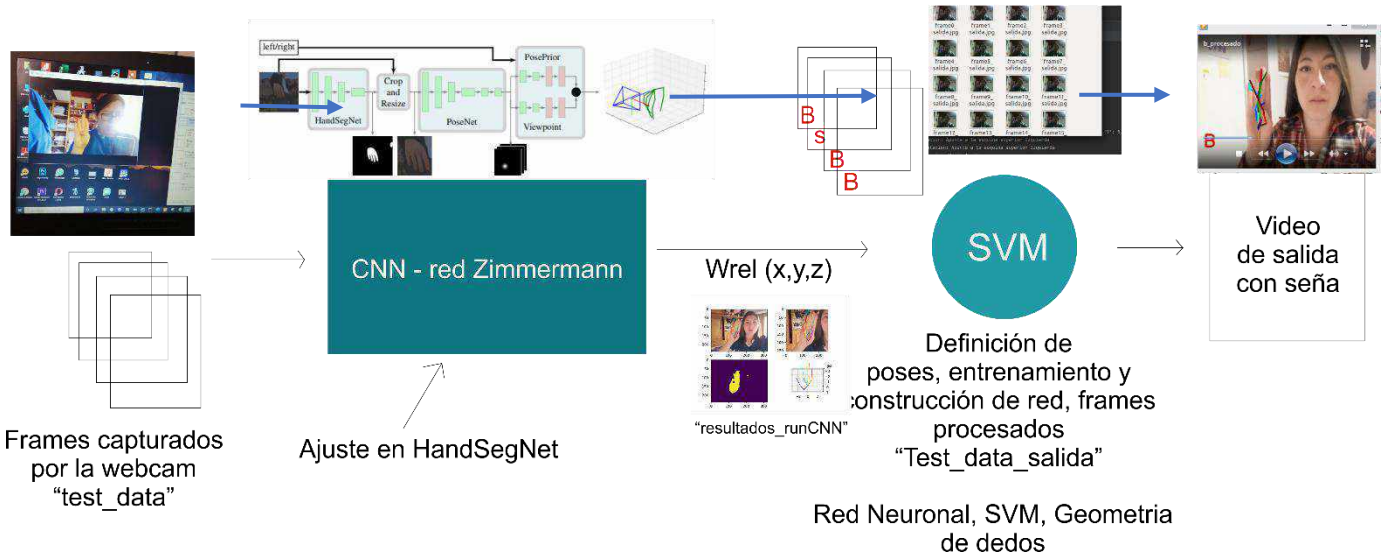
La veracidad de los filtros se consideró gracias a las salidas, para la obtención más clara de los vectores de los 21 puntos clave sobe el plano tridimensional de salida.

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

Figura 2. 16

Diagrama de la metodología general del algoritmo de investigación desarrollada



Fuente: Propia

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

3. Desarrollo, Implantación o Transferencias Tecnológica

La metodología del prototipo, la cual ya fue detallada en el marco teórico del presente trabajo utilizara el prototipo de tipo parches, aquí se especifica las etapas que se deben realizar siguiendo este tipo de prototipado:

3.1. Comunicación

Para la etapa de comunicación se realizaron visitas y reuniones con los usuarios involucrados que son el director y las docentes encargadas de los salones de discapacidad auditiva para definir los objetivos generales y requerimientos. En las reuniones con las docentes se obtuvieron los requerimientos que se necesitaba para la construcción del prototipo.

Las fechas visitadas al CEBE Don José de Martin son las siguientes:

a) Se realizó la primera reunión con el director del CEBE Don José de San Martín quien es el Lic. Helmer del Pozo Cruz, donde se conversó el tema de autorización para poder trabajar con la institución, el día 06/05/2019. (Anexo n°1).
b) La segunda reunión se realizó con el director del CEBE Don José de San Martín, el Lic. Helmer del Pozo Cruz, donde se nos hizo entrega del libro de lengua de señas peruana, el día 15/05/2019. (Anexo n°2)
c) La tercera reunión se realizó con el director del CEBE Don José de San Martín, el Lic. Helmer del Pozo, en esta reunión se pidió la autorización para tener reuniones con las docentes, el día 20/05/2019. (Anexo n°3)
d) Se realizó la cuarta reunión con la docente María del Carmen Celis Gomez, quien enseña a niños con discapacidades auditivas, ella dicta clases a niños de tercer grado de primaria a sexto de primaria, el día 12/06/2019 se recolectó datos realizando una encuesta (Anexo n°5)
e) Se quinta reunión se realizó con la docente Mery Rodrigues Huaman, quien enseña a niños con discapacidades auditivas, ella dicta clases a niños de inicial hasta segundo de primaria, el día 13/06/2020 se realizó una encuesta a la docente para poder recolectar información. (Anexo n°4)
f) Se envió una solicitud al CEBE Don José de San Martin el día 03/12/2019, solicitando un acta de confirmación para poder trabajar con el CEBE Don José de San Martín. (Anexo n°8)

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

Al realizar todas estas visitas obtuvimos los siguientes requerimientos:

a) Reconocimiento de Lenguaje de Señas Básicas como mínimo el alfabeto, no incluyen las señas dinámicas.
b) Reconocimiento de lenguaje de señas en tiempo real o un tiempo comprensible para la comunicación.
c) Interfaz sencilla.

3.1.1. Reconocimiento de los usuarios para el uso del prototipo – Historias de usuario

A. Definición de Usuarios

Tabla 1.

Definición de Usuarios para el prototipo

Nombre	**Características**
Director	Persona quien dirige la el CEBE Don José de San Martin.
Docente de lengua de señas	Persona quien enseña el lenguaje de señas
Alumnos	Persona matriculada en el CEBE Don José de San Martin, alumnos con discapacidad auditiva.
Padres de familia/de alumnos con problemas auditivos	Personas responsables de los alumnos

Fuente: Propia

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

B. Historias de Usuario

Se realizó las historias de usuario al Director, Docentes, Alumnos, Padres de Familia, que son nuestros usuarios finales.

Tabla 2.

Historia de Usuarios

	Historia de Usuario
Nùmero:1	**Usuario:** Director, Docentes de lengua de señas, Alumnos, Padres
Nombre historia: Reconocer el alfabeto de lengua de señas básicas estáticas	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Baja
Programador responsable: Anaid Jimenez Moreano y Brenda Qquecho Ccachainca	
Descripción: Reconocer el alfabeto de lenguaje de señas básicas en tiempo real, con el significado de la seña	
Validación: El director, los docentes de lenguaje de señas, alumnos, y padres de familia como usuarios finales al realizar una seña del abecedario con el uso del prototipo traductor, este deberá reconocer y mostrarle el significado de la seña.	

Fuente: Propia

3.2.Plan rápido

Para el desarrollo del plan rápido, se describe la secuencia para el funcionamiento del prototipo, en este caso la secuencia es la siguiente:

a) Tener la interfaz del prototipo para el reconocimiento de la seña básica del abecedario peruano.
b) La interfaz debe tener acceso a la cámara web, para el reconocimiento de la seña del abecedario.
c) El algoritmo del reconocimiento de la seña del abecedario debe procesar frame por frame para tener una precisión más exacta de la seña del abecedario y hacer pruebas respectivas.

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

d) Una vez terminado el procesado, nos debe mostrar el resultado de la seña del abecedario reconocida.

3.3.Modelado

El modelado es una representación definitiva que se le mostrara al usuario final. Dicho modelado de la interfaz se mostrar a nuestro usuario final.

En la Figura 3.1 se realiza una seña estática durante un aproximado de 8 segundos ante la cámara web del ordenador, y como resultado nos muestra la letra que le corresponde a la seña superpuesta en color rojo en nuestra interfaz.

Figura 3. 1.

Interfaz del prototipo

Nota. Interfaz de traductor de lenguaje de señas peruanas, reconociendo la letra "c" del abecedario peruano. Fuente: Propia

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

3.4.Diseño rápido

Para la etapa de diseño rápido, se define bosquejos para las primeras versiones del algoritmo del prototipo y así se realiza pruebas según requerimientos:

3.4.1. Bosquejo de definición Versión n°1: Reconocimiento de fotos en formato .jpg o .png

Para el reconocimiento mediante fotos en formato ".jpg", ".jpeg", ".png", se necesita entrenar la red de Zimmermann con el dataset, el dataset nos proporciona 41258 imágenes para la formación y 2728 imágenes para la evaluación con una resolución de 320x320 pixeles. Todas las muestras vienen con la anotación completa de un modelo de esqueleto de 21 puntos clave de cada mano y adicionalmente 33 máscaras de segmentación, una para cada palma y cada dedo que está compuesta por 3 segmentos, cada dedo está representado por 4 puntos clave que son:

a) La punta del dedo
b) Dos puntos clave intermedio
c) El extremo situado en la palma
d) Un punto clave final al medio de la palma de la mano

El dataset está basado en imágenes 3D renderizadas, realizadas en Mixamo y Blender.

Para entrenar la red de Zimmermann, se hizo con un sistema operativo Ubuntu 18.04, pero puede ser ejecutado en Windows igualmente instalando los paquetes correspondientes:

a) Python versión 3
b) tensorflow versión 1.30
c) spicy versión 0.18.1
d) matplotlib versión 1.5.3

Teniendo estas librerías como mínimo, se pasa a entrenar fotos elaboradas por las tesistas con la red neuronal para obtener sus posiciones en el plano 3D, se creó una clase (nota: desde este punto se habla de clase como concepto de programación orientada a objetos a diferencia de uso en el contexto de SVM) que denominaremos "CurvaDeDedo", en esta se define el tipo de curvatura del dedo, siendo:

- No Curva
- Medio Curveado
- Full Curva

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

Luego se procede a crear otra clase llamada “Dedo”, esta clase se define los dedos de la mano como “Pulgar”, “Index”, “Medio”, “Anular” y “Menhique”, en seguida se crea otra clase llamada “EstimacionPoseDedo” en la que se saca la pendiente de los dedos, el ángulo de orientación y la curvatura, en consecuencia creamos la clase “PosicionDedo” la que determina si apunta hacia arriba, abajo, a la derecha, a la izquierda, a la esquina derecha inferior, esquina derecha superior, esquina izquierda superior y esquina izquierda inferior, finalmente se denomina otra clase llamada “determinar_posicion” en la que se crea poses conocidas para cada letra, la clase más importante porque aquí definimos las poses para cada letra del abecedario del Lenguaje de señas peruano(LSP).

3.4.2. Bosquejo de definición Versión n°2: Reconocimiento de video, velocidad de procesamiento y frames.

Para el reconocimiento de video, primero debemos haber realizado todo el entrenamiento con nuestro dataset (fotos de cada seña del abecedario peruano), se determina que posiciones va tomar nuestra mano para cada letra del abecedario, en este punto se hizo un ajuste en la etapa de reconocimiento de la mano en la capa HandSegNet, teniendo todo esto listo debemos entrenar la red neuronal de la fase II con videos cortos de diez segundos aproximadamente en formato “.mp4”, se tuvo listos los videos de cada letra del abecedario a estos videos les llamaremos videos de entrada, seguidamente debemos ejecutar la línea de comando en nuestra terminal para procesar nuestros videos de entrada ,para esto se ejecuta cada video individualmente por el plazo aproximado de dieciocho minutos por letra usando un procesador Intel core i5 de segunda generación (2.50 GHz), y tarjeta gráfica NVIDIA GeForce GT 630M/PCIe/SSE2 como instrumento para la fase de entrenamiento de la red neuronal de la fase II y posteriormente comparar con la máquina de vectores de soporte.

Con el número de posición correspondiente a la letra del abecedario detallada en el archivo “DeterminacionPosicion.py”, que hemos declarado anteriormente en el paso del procesado de la imágenes, terminado el proceso del procesamiento se verificó los resultados donde tendremos el video de salida como “videoletra_procesado.mp4” y un archivo .csv, siendo este un array multidimensional tipo Numpy de tamaño [x, 63], siendo “x” un valor cambiante; En el video de salida debemos verificar si se ha reconocido la letra del abecedario y que el archivo de extensión “letra.csv” se haya

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

generado correctamente con la cantidad de data necesaria o en todo caso modificar el algoritmo para que lo reconozca, finalmente después de verificar el archivo del .csv que se haya realizado correctamente con el tamaño adecuado y datos válidos para uso, se ejecuta la instrucción para el procesamiento del video siguiente.

Los valores del archivo con extensión .csv son los puntos clave que tienen cada postura de la mano para el reconocimiento de cada letra y la posición de cada dedo dentro de la mano, estos archivos nos servirán posteriormente para entrenar la red neuronal, usando la geometría de dedos, también usaremos estos datos para entrenar la máquina de vectores de soporte.

3.4.3. Bosquejo de definición Versión n°3 - Final: Sistema con detección webcam en tiempo real aceptable.

Para el reconocimiento en tiempo real o streaming, se procesa los resultados de los archivos .csv, esta colección de archivos se agrega al entrenamiento de la red neuronal y de SVM de la fase II como se ve en la figura 3.2, se creó un algoritmo que permita que la cámara este activa en tiempo real, esta pasa a un método que envía los frames totales capturados por la webcam almacenados en una carpeta llamada "test_data" a un método para ser luego procesados frame por frame en la red neuronal, devuelve los valores estimados (coordenadas) de cada mano en el plano 3D, que seguidamente envía estas coordenadas a la clase de "determinar_posicion" para evaluar la pose, teniendo estos valores en coordenadas Wrel (coordenadas 3D relativas de la pose), hace posteriormente una comparativa de puntos definidos en nuestro archivo "EstimacionPoseDedo.py" para determinar la inclinación y la curvatura de cada dedo en el plano, compara con la posición definida para cada letra y los umbrales por ángulos en el archivo "FormaciondedatosDedos.py", inmediatamente envía al procesamiento una vez se obtiene los valores para cada letra, este enviará el frame procesado para que se le asigne la letra y OpenCv la escriba en la pantalla en la parte inferior izquierda del archivo procesado tipo .jpg que guarda en la carpeta de salida donde todos los frames procesados son almacenados, esta carpeta se denomina "test_data_salida", cuando se tienen todos los frames procesados, se unen todas las imágenes de salida para formar un único video que será guardado en la carpeta "video_salida" y una vez esté listo lo mostrara al usuario final, ver la figura 3.3 del Pseudocódigo del algoritmo de procesamiento en tiempo real.

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

Para acabar limpia las carpetas de entrada y salida de datos, libera también el cache de la red y estará listo para su siguiente uso.

Figura 3. 2.

Agregando archivos entenados de las letras a la red de Zimmermann

Fuente: Propia

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

Figura 3. 3.

Pseudocódigo del algoritmo de procesamiento para correr el prototipo

```
Algoritmo:  Determinacion de la pose final, algoritmo final de procesamiento en tiempo
aceptable,
            puede encontrarse en el archivo denominado "analisis_tiemporeal_version_final.py"

Entrada:    -Usa la red neuronal entrenada en la fase II por nuestros videos de 10 segundos,
tambien se
            puede hacer la clasificacion con SVM (definida como metodo principal) y con el
            algoritmo de
            geometria de dedos que es el uso de las poses como tal.
            -Usa red Zimmermann
            -Usa clase DeterminacionPosicion
            -Usa clase EstimacionPosicionDedo

Salida:     Video procesado de la seña con el nombre de la letra que corresponde.


Metodo declaracion_de_argumentos:
    - "--plot-fingers": Para ver las lineas hechas por la red de Zimmermann = 1, si se desea
    esconder estas
        se declara como 0.
    - "--thresh": Es el umbral o nivel de confianza va de 0-1
    -"--solve-by": Es el metodo para la clasificacion a usar, donde 1 es el algoritmo
    directo de geometria de dedos,
        el 2 es usando SVM y 0 para usar la red neuronal de la fase II entrenada.
    -"--pb-file": Es el archivo de graph.pb que se obtiene al entrenar la red neuronal
    usando los archivos de extension
        .csv que usa la red neuronal para correr, se tiene que especificar la ruta de este.
    -"--svc-file": Ruta donde se almacena el archivo del modelo entrenado con SVM con
    extension .pickle necesaria para funcionar.

Metodo prepara_entrada(ruta_entrada,ruta_salida)
    //ruta del destino de las imagenes
    ruta_entrada <- ruta de entrada absoluta
    //archivos dentro de la carpeta
    data_archivos <- listar

    //si la ruta de salida no se declara, se guarda en el mismo folder de entrada
    si ruta_salida != NO existe
        ruta_salida <- ruta_entrada
    sino
        ruta_salida <- ruta de salida absoluta
    fin si

    retornar data_archivos, ruta_salida


#prediccion por geometria de dedos, envia al algoritmo de SVM, declarado en el capitulo II
metodo prediccion_por_geometria(puntos_Clave3D,poses_dedo_conocidas,thresh)
    EstimacionDedoPose <- Calcula_posicion_dedos()
    posiciones_obtenidas <- determinar_posicion(EstimacionDedoPose)

    etiqueta_De_score <- "Indefinido"

    si longitud(posiciones_obtenidas) > 0
        etiqueta_max_pose <- maximo (posiciones_obtenidas.items())
            si posiciones_obtenidas.etiqueta_max_pose >= thresh
                etiqueta_De_score <- etiqueta_max_pose
            fin si
    fin si

    escribir(posiciones_obtenidas)
    retornar etiqueta_De_score


#prediccion por la red neuronal fase II
metodo prediccion_por_RedNeuronal(puntos_Clave3D,poses_dedo_conocidas,thresh,pb_file)
    deteccion_Archivo_pb()
    etiqueta_De_score <- "indefinido"
    leer_archivo_pb()

    deteccion_puntos_clave_planos()
```

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco

Repositorio Digital

```
    index_maximo <- arraynumpy_de_Argumentosmaximos()
    score_index <- index_maximo si index_maximo.salida >= thresh sino -1
    etiqueta_De_score <- "indefinido"
        si score_index == -1 sino get_nombrePosicion_id(score_index, poses_dedo_conocidas)

    escribir(salidas)
    retornar etiqueta_De_score


#prediccion por SVM
metodo prediccion_por_SVM(puntos_Clave3D,poses_dedo_conocidas,svc-file)
    abrir_archivo_svc_picle()

    deteccion_puntos_clave_planos()
    index_maximo<-svc.predict(deteccion_puntos_clave_planos)
    etiqueta_De_score <- get_nombrePosicion_id(index_maximo,poses_dedo_conocidas)
    retornar etiqueta_De_score


Modulo principal
    //declaracion de argumentos

    #este contador para contar los frames
    contador=0
    #tiempo que estara activa la webcam
    tiempo=8
    inicio_tiempo<-time.time()
    #captura de webcam
    video_Captura <- cv2.VideoCapture()

    INICIO BUCLE
        #toma tiempo actual
        tiempo_presente <- time.time()
        #captura frame por frame
        ret,frame <- video_Captura.read()

        #guarda el frame capturado por la webcam en una carpeta llamada test_data
        cv2.imwrite('./pose/test_data/frame%d.jpg',%contador,frame)
        contador++
        tiempo_pasa <- tiempo_presente - inicio_tiempo

        #muestra el resultado del frame
        cv2.imshow('resultado seña',frame)

        si mantiene_la_web_Cam & (tiempo_pasa > tiempo)
            rompe
        fin si

    //libera y destruye ventana
    //declaracion de ruta de entrada y ruta de salida
    //declaracion de poses de dedo conocidas
    poses_dedo_conocidas <- crear_poses_conocidas()

    #entrada de red
    //hace la evaluacion con libreria de Tensorflow

    #Construccion de la red
    //Hace el llamado a la fase I con la red de Zimmermann
    Fase_I_RED_ZIMERMANN()

    #inicia TF
    //modifica opciones del GPU como memoria y sesion

    #inicia Red Neuronal convolucional -zimermann
    net.inicia()

    #alimenta la imagen a traves de la red
    //hace la lectura y un resize de los frames de entrada a 240x320 pixeles
    //las dimensiones del array numpy se declara tipo float
```

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco

Repositorio Digital

```
si argumentos ==1
    //obtener las coordenadas relativas 3D()
    #obtencion de los scoremaps
    //obtener el mapa de scores()

    #pasa al post procesamiento
    //coordenadas y transformacion de coordenadas()

    plotea_las_coordenadas2D()
sino
    puntos_Clave3D <- corre_la_sesion()

#AQUI DECLARA QUE METODO VA USAR PARA LA CLASIFICACION

si argumentos.solve-by == 0
    //llama al metodo prediccion_por_RedNeuronal()
si argumentos.solve-by == 1
    //llama al metodo prediccion_por_geometria()
sino argumentos.solve-by == 2
    //llama al metodo prediccion_por_SVM()

escribir_texto_de_etiqueta_De_score()
crea_frame_de_Salida_procesado_con_etiqueta_De_score()
guarda_frame(nombreframe+'salida.jpg')


#en este punto transforma los frames de salida que estan en la carpeta test_data_salida
a un video
declara_Array()
para imagenesprocesadas en test_data_salida
    leer_frame()
    declara_Array.adjuntarframe()
fin para

#guarda el video procesado en una carpeta llamada "video_procesado"
guardar_video(formato .mp4)

reproducir_video_procesado()

FIN BUCLE

#limpiar carpetas test_data y test_data_salida
//limpiar carpeta test_data()
//limpiar carpeta test_data_salida()
```

Fuente: Propia

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

3.5. Construcción del Prototipo

Para la construcción del prototipo se realizó las siguientes etapas basadas en el bosquejo de definición de la Versión n°3, explicación a detalle algunos elementos de construcción, siendo la versión final.

3.5.1. Etapa de construcción de escalas de grises y contoneados

En esta etapa se desarrolló el algoritmo para convertir una imagen RGB a escalas de grises y el semi contorneado de la forma de la mano para tener una precisión exacta y poder hacer un mejor filtrado en la capa "PoseNet" de la CNN (Red neuronal de Zimmermann).

Para la realización de esta prueba se utilizó la librería de OpenCv que es "Simple Thresholding", esta librería realiza una comparación de pixel a pixel, si es mayor que el valor del umbral de sombras definido por la librería, entonces les asigna un valor que puede ser blanco o negro, definiendo así un contorno en la matriz de la imagen (definidos como cero o uno).

Se utilizó la función del cv2.threshold, esta utiliza tres parámetros, el primer parámetro es el origen de la imagen, que debe estar en una escala de grises, el segundo parámetro es el valor de threshold o umbral de grises que se utiliza para clasificar los valores de los pixeles y por último, el tercer parámetro es el maxVal que es el representa el valor que se le va dar al pixel, determinando si es mayor o menor que el valor del threshold definido, para dar estos valores la librería de Open Cv ofrece diferentes tipos de umbrales para contoneado:

a) cv2.THRESH_BINARY
b) cv2.THRESH_BINARY_INV
c) cv2.THRESH_TRUNC
d) cv2.THRESH_TOZERO
e) cv2.THRESH_TOZERO_INV

Se probó cada uno de estos y decidió usar el cv2.THRESH_BINARY y cv2.THRESH_TOZERO para obtener un contoneado adecuado, según pruebas hechas en la figura 3.5.

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

3.5.1.1 Etapa de pruebas en el dorso de la mano

Para la prueba de dorso de la mano se tomó una foto sobre un fondo negro, se puede apreciar en la figura 3.4, para tener mejores resultados en la salida de la imagen figura 3.5 aplicando el filtro de cv2.THRESH_BINARY y cv2.THRESH_TOZERO.

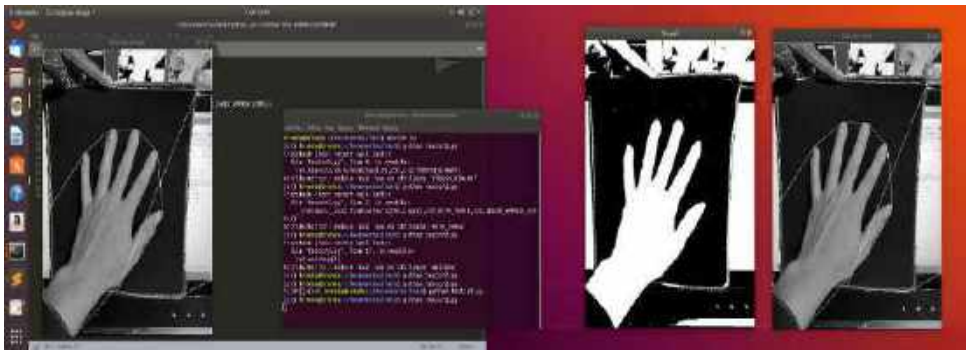
Figura 3. 4.

Dorso de la mano

Nota. Imagen de entrada para aplicar filtros.cv2.THRESH_BINARY y cv2.THRESH_TOZERO.Fuente: Propia

Figura 3. 5.

Imagen de salida con los filtros realizados



Nota. Como se puede observar los filtros nos ayudan a tener mejores resultados, estos filtros aplicados reconocen la forma de la mano con más exactitud. Fuente: Propia

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

3.5.1.2 Etapa de pruebas en la palma de la mano

Para esta etapa de la prueba en la palma de la mano se puede visualizar la figura 3.6 que se define como nuestra imagen de entrada en la cual se aplicó el filtro de tipo cv2.THRESH_BINARY_INV y como resultado tenemos la imagen de salida que se puede apreciar en la figura 3.7.

Figura 3. 6.

Imagen de entrada

Nota. Palma de la mano, en cual se aplicará el filtro de cv2.THRESH_BINARY_INV.Fuente: Propia

Figura 3. 7.

Imagen de salida

Nota. Con el filtro aplica a nuestra imagen de entra, como resultados podemos observar nos delimita la forma de la mano. Fuente: Propia

DERECHOS DE AUTOR RESERVADOS

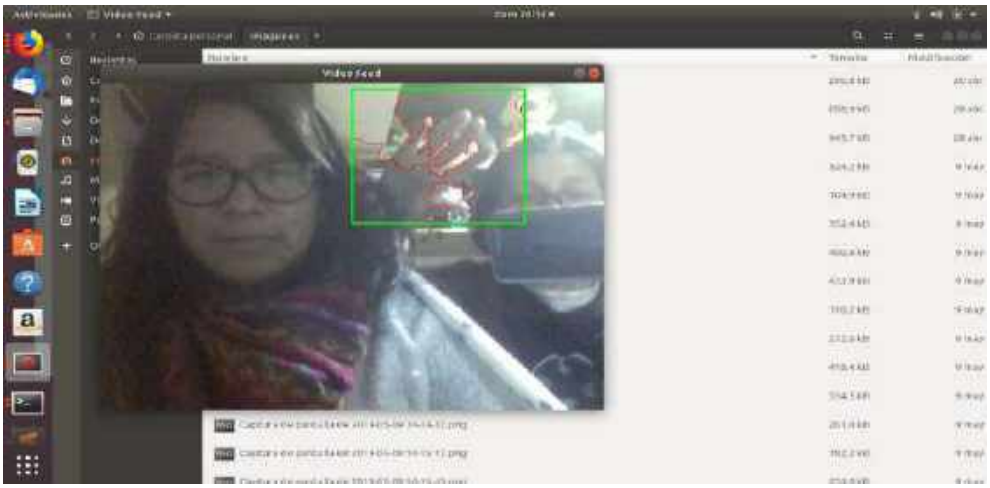
Universidad Andina del Cusco
Repositorio Digital

3.5.1.4 Prueba de reconocimiento de contorno de la mano en vídeo

Para el reconocimiento del contorno de la mano y la mejora que se agrega a la capa de SegNet de Zimmermann (segunda capa), la posición de la mano debe ser visible por la cámara, la posición no debe mostrar el dorso de la mano ni en forma diagonal mostrando la mano como se muestra en la figura 3.8 ya que la cámara no podrá hacer el reconocimiento del contorno de la mano. Para tener una detección adecuada se debe mostrar la palma de la mano como se muestra en la figura 3.9; Teniendo de esa manera la cámara podrá tener un mejor reconocimiento del contorno de la mano, como se hizo en las pruebas.

Figura 3. 8.

Reconociendo de la mano mediante video



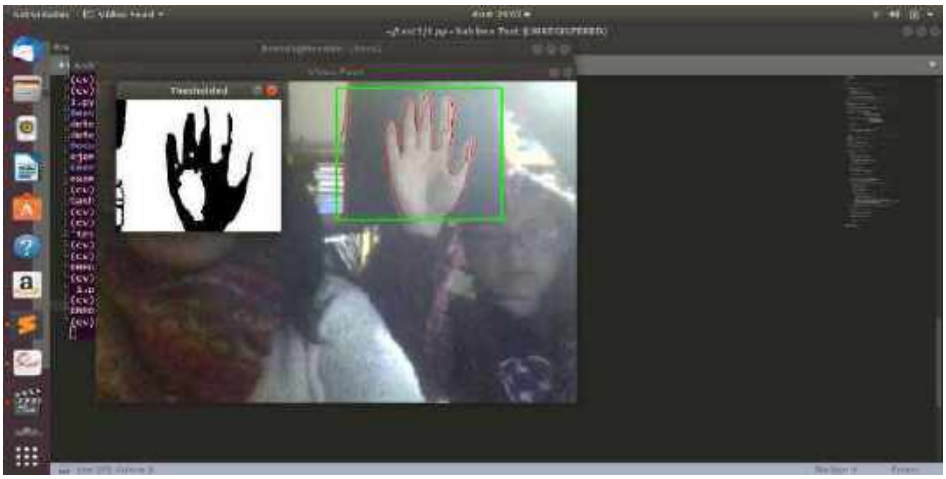
Nota. La posición puesta es inadecuada en la mano, la posición de la mano no debe estar de forma diagonal u otra forma, la mano se debe estar en posición de frente para un reconocimiento en el contorno de la mano. Fuente: Propia

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

Figura 3. 9.

Recogimiento del contorno de la mano en video, posición correcta



Fuente: Propia

3.5.1.4 Etapa de pruebas de aplicación de filtros a la mano

En esta etapa de aplicación de los filtros usamos el filtro tipo "GaussianBlur" de la biblioteca OpenCV2 como se puede observar en la figura 3.10, que nos permitió tener un filtro de desenfoque a una imagen para suavizar algunos aspectos rugosos de la mano.

Asimismo, se aplicó el filtro "Adaptive Thresholding", este calcula el threshold o umbral para pequeñas regiones de la imagen, el resultado de este filtro es el que obtenemos con diferentes threshold para diferentes regiones.

Se obtuvo mejores resultados con este filtro adicional.

El Adaptive Thresholding ofrece:

a) cv2.ADAPTIVE_THRESH_MEAN_C: donde el threshold es la media del área del vecindario.
b) cv2.ADAPTIVE_THRESH_GAUSSIAN_C: el valor threshold es la suma ponderada de los valores del vecindario donde los pesos son una ventana gaussiana.

Para estas pruebas se aplicaron los tipos de cv2.Adaptive_Thresh_Mean_C y cv2.Adaptive_Thresh_Gaussian_C, el resultado de estos filtros se puede observar en la figura 3.11

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

Figura 3. 10.

Utilizando filtro de cv2.GaussianBlur

Nota. Como se puede observar nos da un mejor contorno de la mano. Fuente: Propia

Figura 3. 11. Imagen de salida con los filtros utilizados

Nota. Filtros de cv2.Adaptive_Thresh_Mean_C y cv2.Adaptive_Thresh_Gaussian_C, donde se puede apreciar que tenemos mejores resultados a comparación de la figura 3.9, para un mejor reconocimiento del contorno de la mano. Fuente: Propia

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

3.5.2. Entrenando la red Neuronal de Zimmermann con datos de las tesistas

Para el entrenamiento de la red Neuronal de Zimmermann se debe tener los siguientes requisitos cumplidos:

a) Sistemas Operativo: Ubuntu 16.04.2 en adelante (Nosotras usamos la distribución 18.04 de Ubuntu)
b) Biblioteca de Python 3.5.2 en adelante
c) Biblioteca de pip versión 9.0.1
d) Biblioteca de Tensorflow versión 1.3.0
e) Biblioteca de numpy versión 1.19.1
f) Biblioteca de scipy versión 1.1.0
g) Biblioteca de matplotlib versión 3.2.0
h) Biblioteca de imageio versión 2.9.0
i) Biblioteca de imageio-ffmpeg versión 0.4.1
j) Biblioteca de moviepy versión 1.0.1
k) Biblioteca de opencv-python versión 4.2.0.32
l) Biblioteca de Pillow versión 7.2.0
m) Biblioteca de scikit-learn versión 0.23.2
n) Biblioteca de tensorflow-tensorboard versión 0.1.8

Los pasos para entrenar la red de Zimmermann:

1. En el código de Zimmermann de licencia abierta para investigaciones, buscamos el archivo con el nombre créate_binary_db.py, donde buscamos la siguiente línea de código path_to_db=”nombre_archivo_dataset”, quedaría de la siguiente manera:
   Path_to_db=’./RHD_published_v2’, guardamos los cambios realizados.
   Se usó el dataset de menor peso encontrado (dataset de 7.1 GB, de poses de manos renderizados en 3D ofrecido por Christian Zimmermann, ver anexo n° 84), este dataset proporciona 41258 muestras de entrenamiento y 2728 de pruebas y demás detalles explicados en el marco teórico.
2. En el mismo archivo nos encontramos con el modo ‘training’, el segundo modo que está en ‘evaluation’ que nos permitirá usar la red Zimmermann en modo evaluación para su uso o en modo entrenamiento para entrenarla, o modo evaluación para hacer uso de la red ya entrenada.

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

3. Se compila el código del archivo “create_binary_db.py” para realizar el entrenamiento.

   La red se demora en entrenar un mínimo de 3 horas y un máximo de 6 horas (valores promedio). Para ver todas las imágenes de entrenamiento de la red ver el anexo n°9.a

4. Terminado el entrenamiento con la configuración de ‘training’, continuamos con la siguiente instrucción: python3 training_handsegnet.py para ver la instrucción ir anexo nº 9.b, para obtener el siguiente paso del preprocesamiento al del entrenamiento pasando por la capa uno de la CNN (“HandSegNet”).
5. Terminado el anterior proceso, en el terminal colocamos la siguiente instrucción para finalizar el entrenamiento de la capa dos: python3 training_posenet.py para ver el procesamiento ir al anexo nº 9.c, pasando está a la capa tres de obtención de las coordenadas de “PoseNet” y posteriormente obtener el set entrenado para su uso.
6. Terminado la anterior instrucción, se compila el archivo con el nombre eval2d_gt_cropped.py, esto permitirá reentrenar la red y agregar valores posteriores para nuestras poses y a futuro si se desea agregar otras, poder adicionar las poses a las ya entrenadas.

   Para ver cómo termina la compilación del archivo eval2d_gt_cropped.py y como se hizo el entrenamiento completo ver el anexo nº 9.d.

7. Los resultados deben compararse con los resultados del paper, ya que el paper obtiene según pruebas un accuracy o precisión del 92%, como se detalla a continuación en la figura 3.12, los resultados obtenidos por las tesistas, se igualan a los resultados obtenidos en el paper tanto para el entrenamiento como para la evaluación:

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

Figura 3. 12.

Resultados obtenidos para el entrenamiento y evaluación de la red de Zimmermann

eval2d_gt_cropped.py yields:

```
Evaluation results:
Average mean EPE: 7.630 pixels
Average median EPE: 3.939 pixels
Area under curve: 0.771
```

eval2d.py yields:

```
Evaluation results:
Average mean EPE: 15.469 pixels
Average median EPE: 4.374 pixels
Area under curve: 0.715
```

Nota. Que se espera obtener una vez entrenada la red, tomando en cuenta la media, la mediana y el área debajo de la curva.

8. Para realizar el entrenamiento de la red de Zimmermann en modo evaluación, debemos cambiar en el archivo con nombre "create_binary_db.py" del modo de "training" a "evaluation" una vez realizado y guardados los cambios compilamos el archivo create_binary_db.py. En esta evaluación se trabaja con las 2728 imágenes de pruebas.

   En entrenamiento dura de 1 hora a dos 2 horas como máximo. Para ver cómo termina el entrenamiento ir al anexo n° 9.e.
9. Para obtener nuevos puntos clave para el entrenamiento con nuestro dataset, se compilo los siguientes archivos:
   a) Python3 eval2d_gt_cropped.py
   b) Python3 eval2d.py
   c) Python3 eval3d.py
   d) Python3 eval3d_full.py

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

3.5.3. Obtención de puntos claves para cada pose de la mano

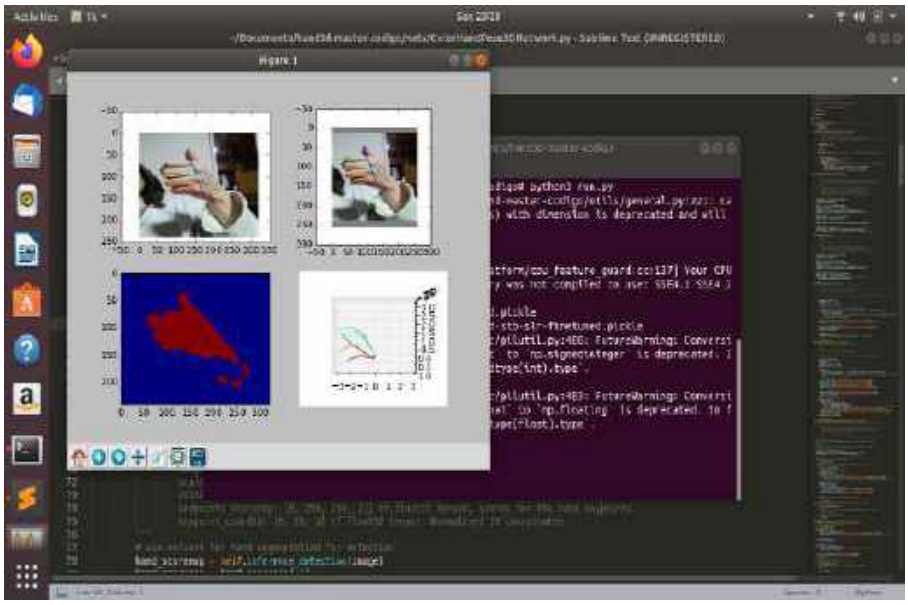
Para la obtención de puntos clave y coordenadas relativas para cada pose de la mano cómo podemos observar en la figura 3.13 se usó una imagen de entrada, detectamos puntos clave en 2D, el primer cuadro muestra los puntos de cada mano, seguidamente en el segundo cuadro muestra otra escala de puntos con más exactitud, en la que primero muestra la coordenadas en función a los ejes X y Z, en el tercer cuadro la mano saca los puntos y lo muestra con el filtro aplicado para tener la imagen, además de obtener la rotación y el punto de vista.

La posición en la que estaría se muestra en el último cuadro, se detecta las coordenadas normalizadas obtenidas a partir de las coordenadas relativas obtenidas, se obtiene la simetría diferencia también la mano derecha de la izquierda, gracias a los puntos claves del 2D, la normalización de las coordenadas relativas 3D y el punto de vista obtenidos a partir de una matriz de mapas de score, que nos permiten estimar una posición de manos normalizadas en 3D teniendo así las coordenadas relativas en los ejes X,Y y Z.

Para mayores detalles matemáticos y otros, revisar el paper declarado en el marco teórico o en el anexo n° 62.a

Figura 3. 13.

Estructura de las posiciones de la mano



Nota. Donde nos permite tener puntos clave en 2D, gracias estos datos obtenemos datos que nos permiten estimar una posición de manos normalizada en 3D, para ver más estructuras de las posiciones de la mano revisar el anexo n°10.Fuente: Propia

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

3.5.4. Usando geometría de los dedos

Para el uso de la geometría de los dedos se debe especificar que posiciones va tomar nuestra mano, para determinar las posiciones de cada dedo de la mano, se definió en dos partes:

3.5.4.1. Curvatura del dedo: Es la curva que va a tener el dedo para cada pose, las que especificamos de la siguiente manera:

a) NoCurva: se puede apreciar en la figura 3.14.

Figura 3. 14.

Posición del dedo en "NoCurva"

Fuente:Propia

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

b) MediaCurva: Se pude apreciar en la figura 3.15.

Figura 3. 15.

Posición del dedo en "MediaCurva"

Fuente: Propia

c) FullCurva: Se puede apreciar en la figura 3.16.

Figura 3. 16.

Posición del dedo en "FullCurva"

Fuente: Propia

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

3.5.4.2. Orientaciones y direcciones: Es la orientación y dirección del dedo, las cuales son las siguientes que definimos:

a) HaciaArriba: Se puede apreciar en la figura 3.17.

Figura 3. 17.

Posición del dedo "HaciaArriba"

Fuente: Propia

b) HaciaAbajo: Se puede apreciar en la figura 3.18.

Figura 3. 18.

Posición del dedo "HaciaAbajo"

Fuente: Propia

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

c) ApuntaIzquierda: Se puede apreciar en la figura 3.19.

Figura 3. 19.

Posición de dedo "ApuntaIzquierda"

Fuente: Propia

d) ApuntaDerecha: Se puede apreciar en la figura 3.20.

Figura 3. 20.

Posición del dedo "ApuntaDerecha"

Fuente: Propia

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

e) EsquinaDerecha: Se puede apreciar en la figura 3.21.

Figura 3. 21.

Posición del dedo "EsquinaDerecha"

Fuente: Propia

f) EsquinaIzquierda: Se puede apreciar en la figura 3.22.

Figura 3. 22.

Posición del dedo "EsquinaIzquierda"

Fuente: Propia

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

g) EsquinaInfIzquierda: Se puede apreciar en la figura 3.23.

Figura 3. 23.

Posición del dedo "EsquinaInfIzquierda"

Fuente: Propia

h) EsquinaInfDerecha: Se puede apreciar en la figura 3.24.

Figura 3. 24.

Posición del dedo "EsquinaInfDerecha"

Fuente: Propia

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

3.5.5. Usando la red Neuronal de Zimmermann entrenada y posiciones definidas para las letras del abecedario del Lenguaje de señas peruano (LSP).

Para ello se crea un archivo “DeterminacionPosicion.py”, en esta definimos los siguientes métodos “determinar_posicion”, “get_nombrePosicionId” y “crear_poses_conocidasDedos” para determinar la geometría de los dedos para cada letra del abecedario y la red neuronal, se especifica la posición de la curvatura del dedo y la orientación del dedo; Prontamente especificamos el umbral de confianza para cada uno.

En la fase final se observa cómo se define la geometría de dedos teniendo en consideración la curvatura, la dirección y orientación, se programa todo este fragmento de código para cada letra; Para mayor detalle revisar el link donde se encuentra todo el proyecto de la tesis, en el archivo de “DeterminacionPosicion.py”, el link se puede hallar en el anexo n° 62.c.

3.5.5.1. Entrenando imágenes estáticas. Usando la red neuronal ya entrenada con nuestro dataset y nuestra clase dentro de “DeterminacionPosicion.py”, se tuvo que hacer en cada prueba modificaciones del algoritmo para obtener tener mejores resultados de las letras del abecedario, el entrenamiento y resultado de las imágenes estáticas se puede apreciar en la tabla n° 3. a continuación:

Tabla 3.

Entrenamiento de imágenes estáticas

LETRAS DEL ABECEDARIO	ANEXOS
Letra A	Ver anexo n° 14
Letra B	Ver anexo n° 15
Letra C	Ver anexo n° 16
Letra D	Ver anexo n° 17
Letra E	Ver anexo n° 18
Letra F	Ver anexo n° 19
Letra G	Ver anexo n° 20
Letra H	Ver anexo n° 21
Letra I	Ver anexo n° 22
Letra K	Ver anexo n° 23

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

Letra L	Ver anexo n° 24
Letra M	Ver anexo n° 25
Letra N	Ver anexo n° 26
Letra O	Ver anexo n° 27
Letra P	Ver anexo n° 28
Letra Q	Ver anexo n° 29
Letra R	Ver anexo n° 30
Letra S	Ver anexo n° 31
Letra T	Ver anexo n° 32
Letra U	Ver anexo n° 33
Letra V	Ver anexo n° 34
Letra W	Ver anexo n° 35
Letra X	Ver anexo n° 36
Letra Y	Ver anexo n° 37

Fuente: Propia

3.5.5.2. Entrenado con videos

3.5.5.2.1. Entrenando con diferentes procesadores. Se hizo pruebas con diferentes procesadores, para realizar una comparativa de tiempos de procesamiento de los videos donde se obtuvo los siguientes resultados:

a) Procesador Intel Core i3: El tiempo que se demoró en entrenar un video de una seña con nuestra red entrenada fue de 59 minutos, para ver el proceso de entrenamiento ir al anexo n° 11.

b) Procesador Intel Core i5: El tiempo que se demoró en entrenar un video de una seña con nuestra red entrenada fue de 35 minutos, para ver el proceso de entrenamiento ir al anexo n° 12.

c) Procesador Intel Core i7: El tiempo que se demoró en entrenar un video de una seña con nuestra red entrenada fue de 12 minutos, para ver el proceso de entrenamiento ir al anexo n° 13.

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

3.5.5.2.2. Pruebas del entrenamiento de video. Al entrenar un video, este genera el video de salida y un archivo con extensión .csv, en este archivo se obtiene los puntos clave para el reconocimiento y procesamiento, para ver esto con mayor detalle revisar la tabla n° 4 a continuación:

Tabla 4.

Entrenamiento de videos de 5 segundos a 10 segundos como máximo (Procesador el Core i5)

LETRAS DEL ABECEDARIO VIDEO	TIEMPO DE PROCESAMIENTO DEL VIDEO	ANEXOS
Letra A	24 minutos	Ver anexo n° 38
Letra B	37 minutos	Ver anexo n° 39
Letra C	30 minutos	Ver anexo n° 40
Letra D	32 minutos	Ver anexo n° 41
Letra E	35 minutos	Ver anexo n° 42
Letra F	37 minutos	Ver anexo n° 43
Letra G	32 minutos	Ver anexo n° 44
Letra H	23 minutos	Ver anexo n° 45
Letra I	32 minutos	Ver anexo n° 46
Letra K	43 minutos	Ver anexo n° 47
Letra L	33 minutos	Ver anexo n° 48
Letra M	28 minutos	Ver anexo n° 49
Letra N	44 minutos	Ver anexo n° 50
Letra O	43 minutos	Ver anexo n° 51
Letra P	36 minutos	Ver anexo n° 52
Letra Q	26 minutos	Ver anexo n° 53
Letra R	45 minutos	Ver anexo n° 54
Letra S	35 minutos	Ver anexo n° 55
Letra T	28 minutos	Ver anexo n° 56
Letra U	30 minutos	Ver anexo n° 57
Letra V	28 minutos	Ver anexo n° 58
Letra W	48 minutos	Ver anexo n° 59

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

Letra X	23 minutos	Ver anexo n° 60
Letra Y	46 minutos	Ver anexo n° 61

Fuente: Propia

3.6. Despliegue, entrega y Retroalimentación

3.6.1. Pruebas en streaming utilizando Webcam

Para finalizar nuestro algoritmo, una vez la red ha sido entrenada se puede observar en las siguientes figuras 3.25, 3.26, 3.27 y 3.28 el resultado, tanto con imágenes solas como con videos de 5 segundos como mínimo y 10 segundos como máximo, gesticulando las posiciones de cada letra del abecedario estático, trabajamos con la red neuronal estándar y los SVM (Support Vector Machine), pasamos a hacer las pruebas del procesamiento en tiempo real.

Se inicializa la webcam, se define el tiempo que estará capturando los frames la cámara web, en este caso definimos 8 segundos, este tiempo puede ser configurable como se observa en la figura 3.29.

En la figura 3.30 se ve como empieza a almacenar los frames en la carpeta de entrada “test_data”, y como los frames procesados empiezan a aparecer en la carpeta de salida “test_data_salida” que se observa en la figura 3.31 y figura 3.32, esto se hace en un tiempo corto de aproximadamente 5 minutos, generando finalmente el video de salida mostrando la seña que corresponde y satisfaciendo así al usuario final como se observa en la figura 3.33, con esto el prototipo concluye y cumple su función definida en la metodología y descrita según requerimientos.

Cabe recalcar que el tiempo estimado de entrenamiento usando video es muy diferente al tiempo que el algoritmo corre y procesa con las letras ya entrenadas en tiempo real, evitar generar confusión con respecto a este punto.

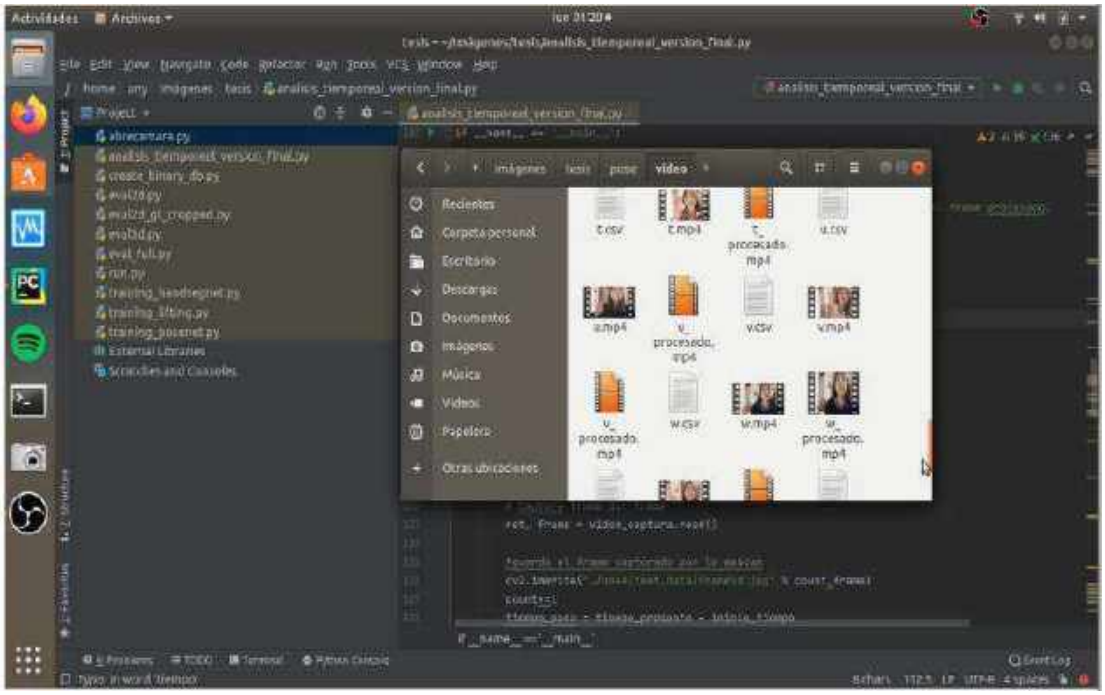
Además de considerar que el computador con el cual se estuvo trabajando consta de un procesador Core i5 de segunda generación (2.50 GHz), con una tarjeta de video GF108M GeForce GT 620M NVIDIA y sistema operativo Ubuntu 18.04 como sistema único y principal.

Generando un video de salida final el cual muestra el resultado procesado, para mayor detalle revisar el enlace del video en el anexo n° 62.d, donde se muestra la diferencia de entrenamiento y como se ejecuta en tiempo real el código, se muestra la prueba del código final de la detección de la seña peruana estática.

DERECHOS DE AUTOR RESERVADOS

Figura 3. 25.

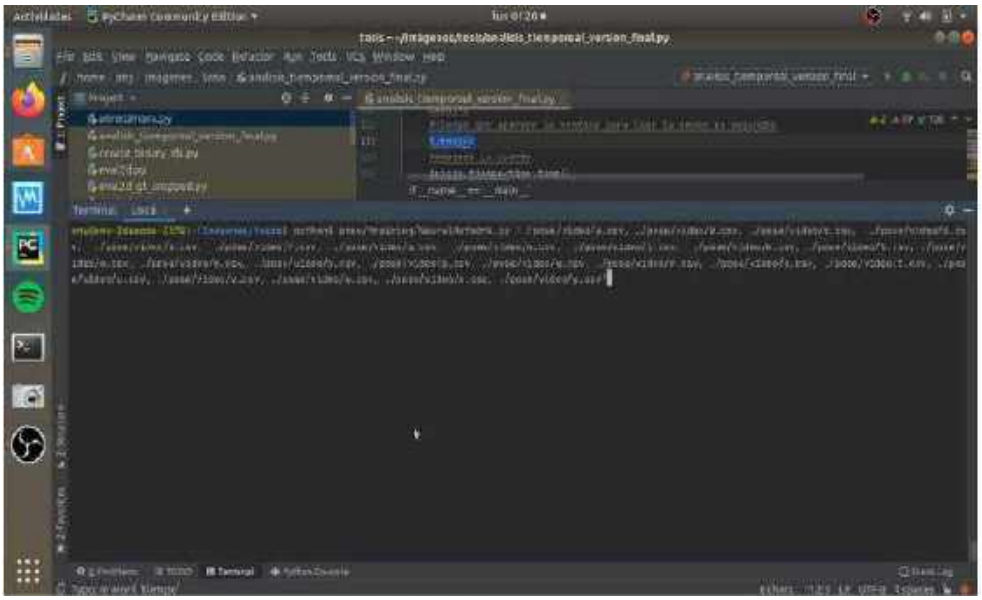
Obtención de los .csv como dataset para la fase II



Nota. A partir del entrenamiento individual de un video por letra de abecedario. Fuente: Propia

Figura 3. 26.

Procesamiento de los .csv



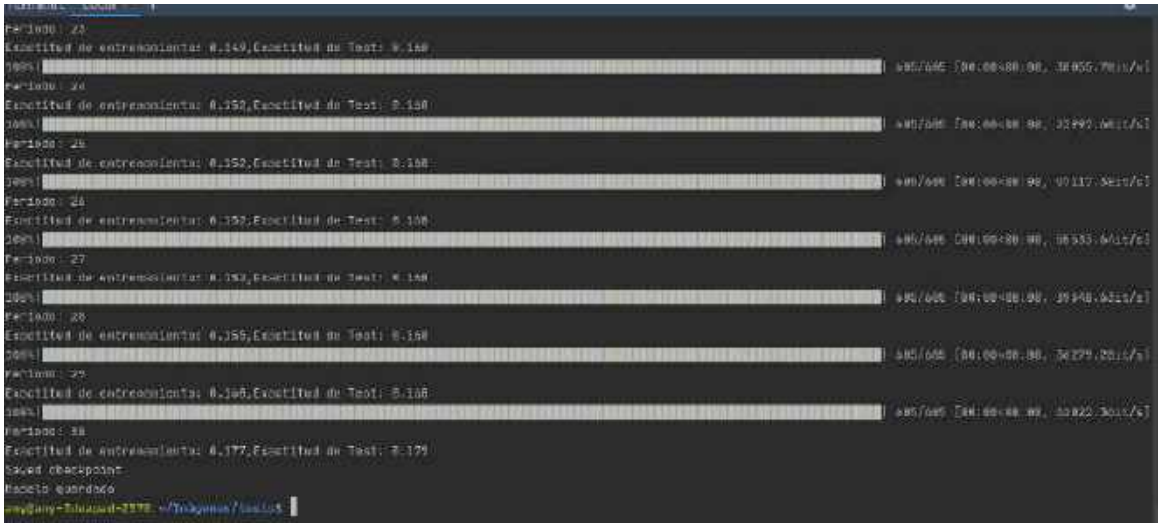
Nota. Nos permiten tener los puntos claves de cada seña del abecedario. Fuente: Propia

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

Figura 3. 27.

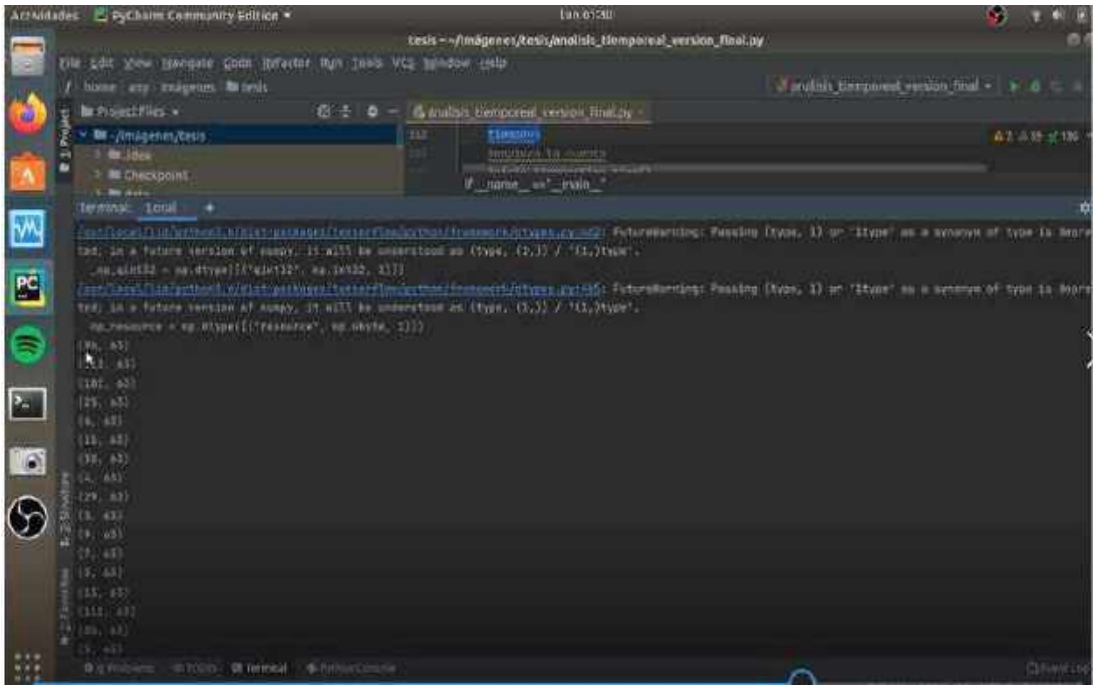
Procesamiento de los .csv



Nota. Nos permiten tener los puntos claves de cada seña del abecedario, se ve el accuracy por letra. Fuente: Propia

Figura 3. 28.

Procesamiento de los .csv



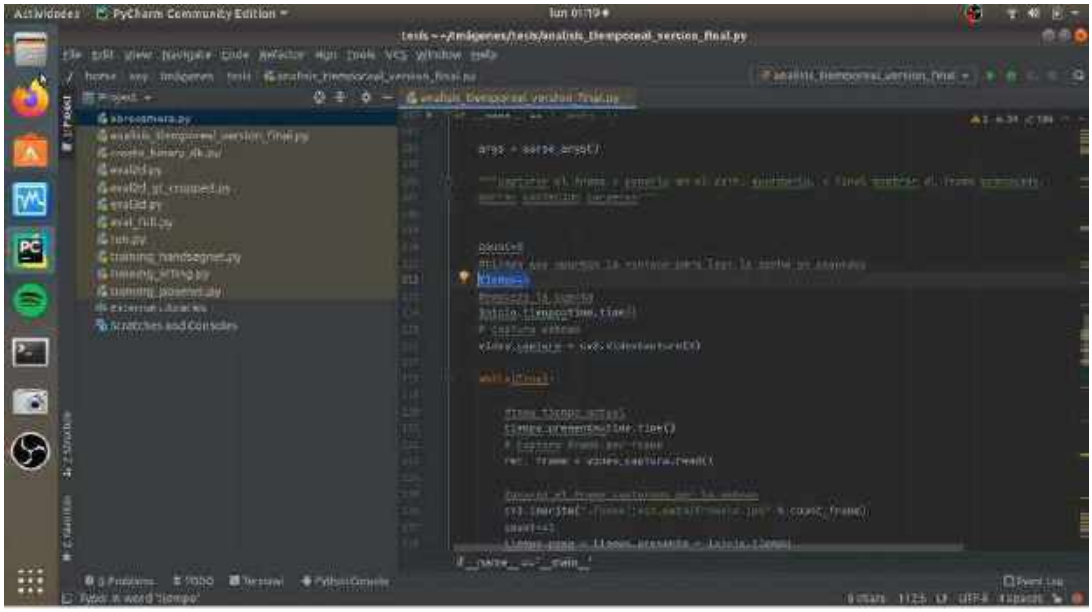
Nota. Nos permiten tener los puntos claves de cada seña del abecedario y el tipo de matriz por letra. Fuente: Propia

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

Figura 3. 29.

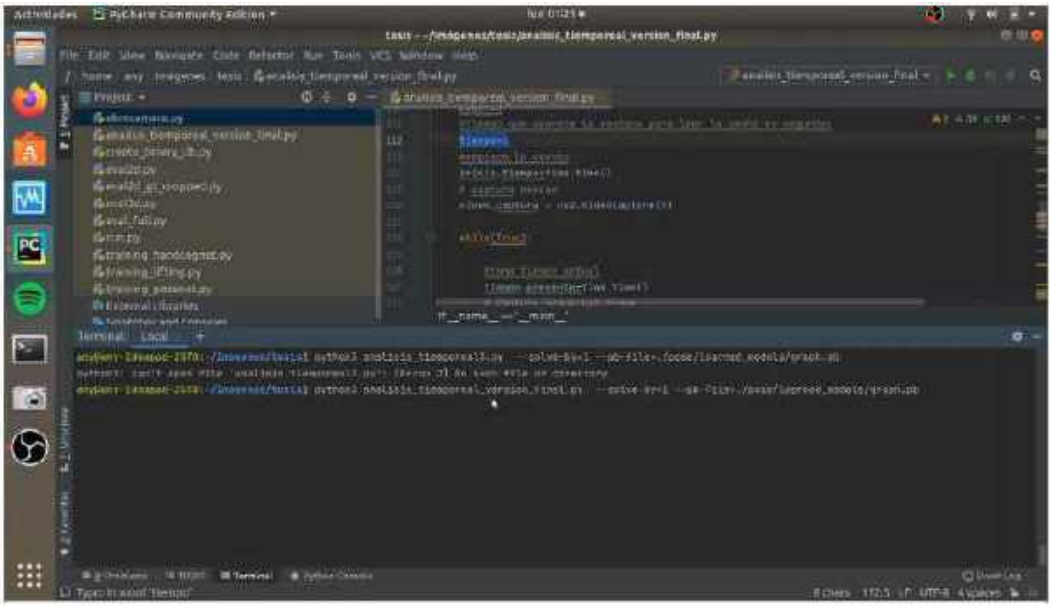
Código para determinar número de tiempo para capturar los frames



Fuente: Propia

Figura 3. 30.

Pruebas en tiempo real del reconocimiento de las letras del abecedario



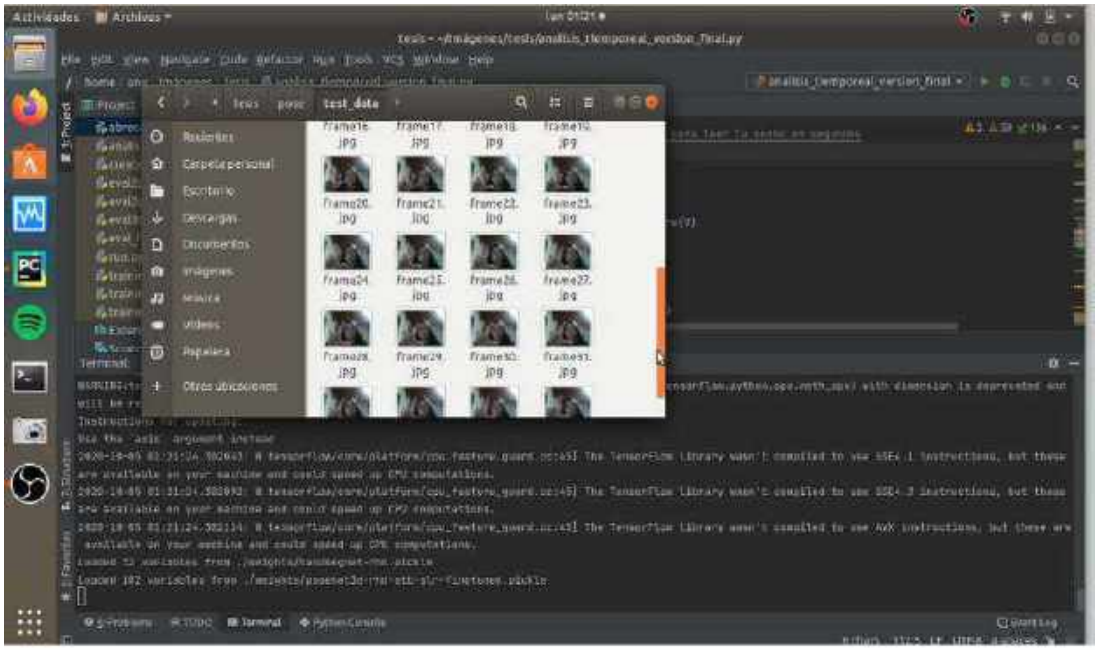
Fuente: Propia

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

Figura 3. 31.

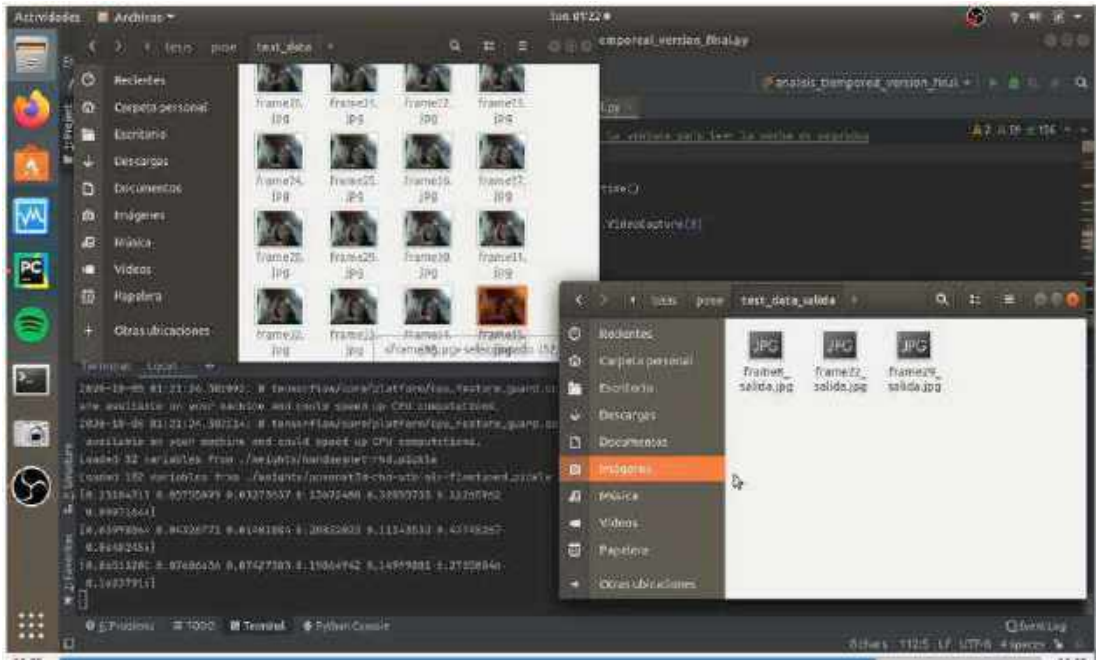
Obtención de los frames de la captura del video de 6 segundos



Fuente: Propia

Figura 3. 32.

Procesamiento de los frames para la detección de la seña



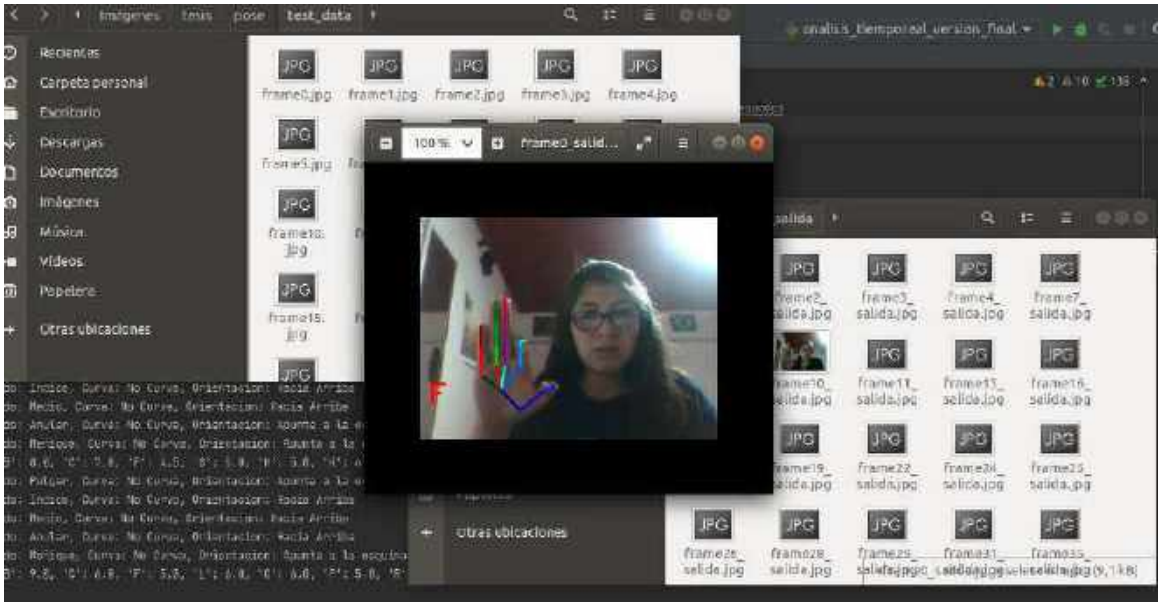
Fuente: Propia

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

Figura 3. 33.

Resultado del procesamiento del video



Nota.Como se puede observar hace el procesamiento de cada frame para determinar que letra del abecedario corresponde. *Fuente: Propia*

3.6.2. Entrega al director

El día 15 de octubre del 2020 se tuvo una reunión vía Zoom con el director Helmer del Pozo Cruz para poder acordar los términos de implementación y la fecha que se haría la entrega del prototipo, en dicha reunión se estimó la implementación del prototipo en uno de los equipos principales del laboratorio del CEBE, la hora que se llegaría al colegio y los protocolos de seguridad y salud que se mantendrían durante el desarrollo del trabajo debido a la pandemia del Covid-19.

Con respecto a la implementación del prototipo, se llevó a cabo el día: sábado 17 de octubre del 2020, el equipo donde se realizó la instalación tiene las siguientes características:

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

Tabla 5.

Características de la computadora para la instalación del prototipo

Procesador	Memoria RAM	GPU	Sistema Operativo
Intel Core i5-7400 3.00GHz	8GB DDR4	Intel HD Graphics 630	Windows 10 Home

Fuente: Propia

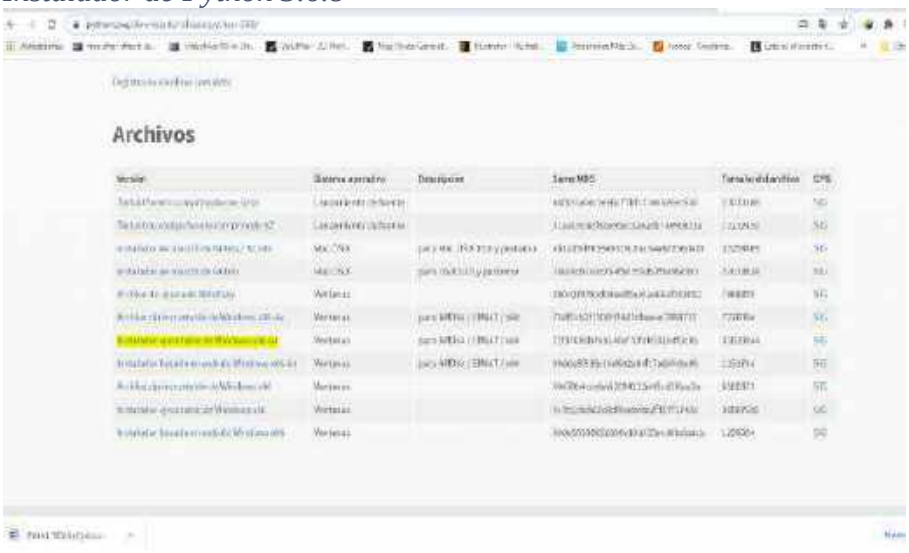
Se realizaron los siguientes pasos para la instalación del prototipo:

1. Instalación de Python en Windows

   Paso descargar la versión 3.6.8 de Python, acceder al link que se encuentra en la página oficial de Python:
   https://www.python.org/downloads/release/python-368/ .
   Descargar el instalador ejecutable para Windows (plataforma 64 bits).

Figura 3. 34.

Instalador de Python 3.6.8



Fuente: Propia

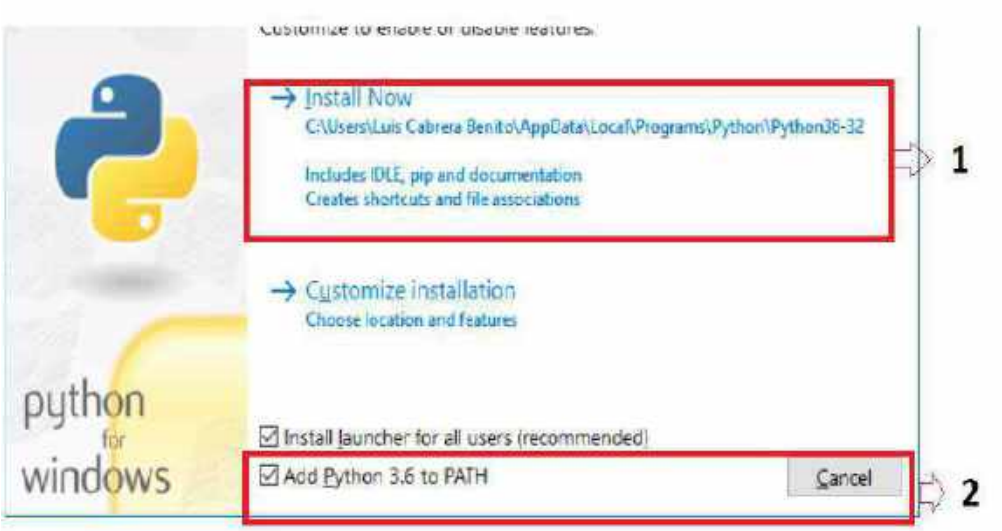
DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

2. Al realizar la instalación de Python marcar las casillas “Add Python 3.6.8 to PATH”, esto permitirá reconocerlo en la consola de comandos de Windows.

Figura 3. 35.

Instalación de Python 3.6.8 en Windows

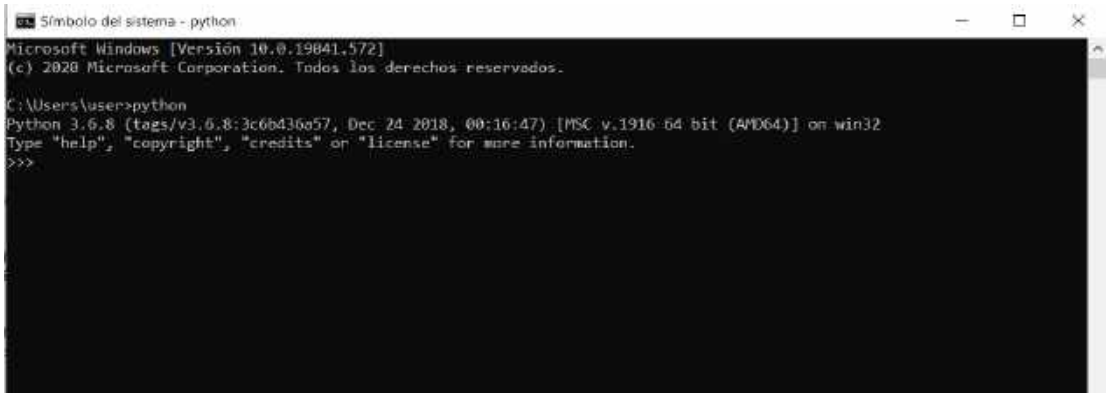


Fuente: Propia

3. Para comprobar la instalación de Python y la versión requerida, entramos en el bash de Windows, digitamos Python

Figura 3. 36.

Verificación de la version de Python instalado



Fuente: Propia

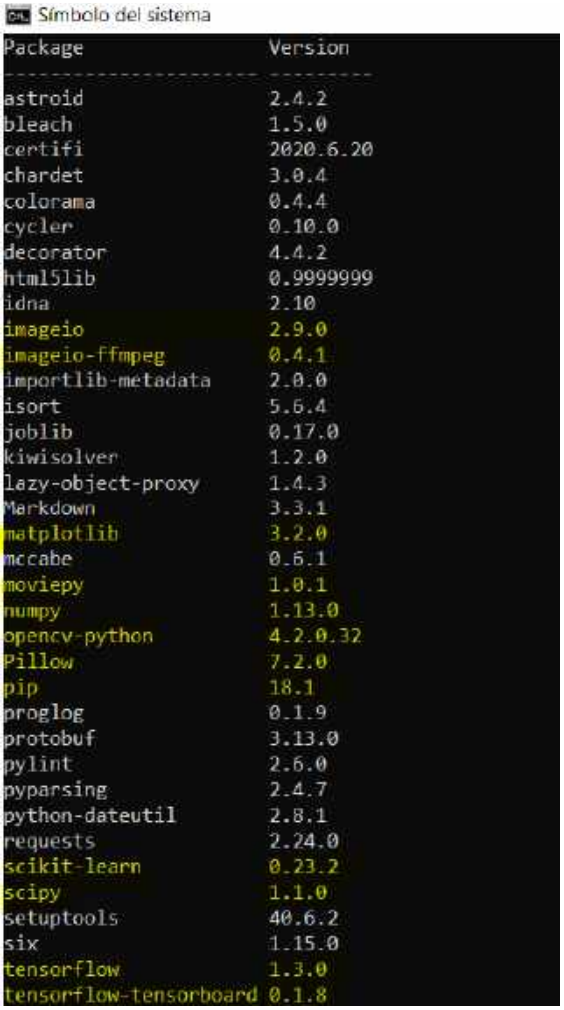
DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

4. Instalar las siguientes bibliotecas con el comando pip, ejemplo: "**pip install numpy==1.13.0"**

Figura 3. 37.

Bibliotecas requeridas para el funcionamiento del prototipo



Fuente: Propia

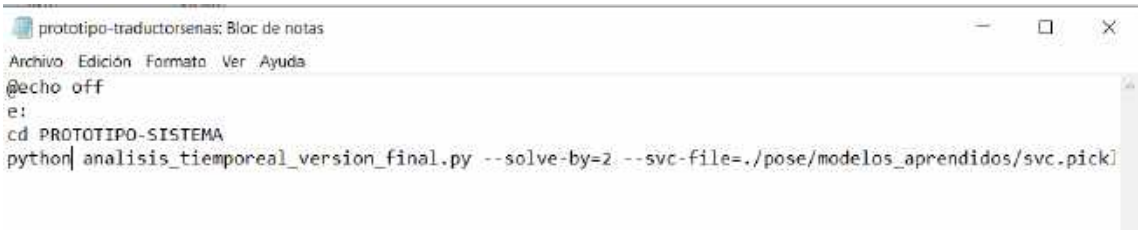
5. Como último paso creamos el archivo con extensión .bat para que el programa sea ejecutable desde escritorio, el archivo debe contener las siguientes líneas de instrucciones, una vez la carpeta del prototipo se haya copiado en la ruta 'E' o la ruta correspondiente según sea.

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

Figura 3. 38.

Línea de comandos para ejecutar prototipo con SVM



Fuente: Propia

6. Guardamos cambios. Se creará con el siguiente icono, el cual podrá ser configurado para que se ejecute automáticamente al inicial la computadora o para que pueda ser ejecutado automáticamente al darle un clic.

   El acta de entrega del prototipo se detalla en el anexo n° 63, para ver mayores detalles al respecto.

Figura 3. 39.

Icono de prototipo

Fuente: Propia

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

4 Resultados

4.1. Comprobación de la prospectiva

Al momento de ser entregado el trabajo de investigación al Lic. Helmer, aceptando el prototipo de manera formal, detallado en el anexo n°63, esperamos que la institución CEBE Don Jose de San Martin puedan darle un uso eficiente y posteriormente tener mejoras del prototipo.

Consideramos solo las letras estáticas, mas no las dinámicas que son las letras “J”,”Ñ” y ”Z ”, obteniendo resultados favorables.

A pesar de que el prototipo es multiplataforma, se optó por hacer las pruebas debidas en el sistema operativo de Ubuntu 18.04, en un equipo core i5 obteniendo resultados aceptables, por otro lado en el sistema operativo Windows en un core i7 los resultados fueron favorables con respecto al tiempo de procesamiento, los resultados para correr una misma letra usando SVM en una configuración de 5 segundos de captura de frames con la webcam en ambos equipos se puede visualizar en el cuadro n° 6 a continuación.

Tabla 6.

Resultados de clasificar una letra en diferentes equipos de cómputo

Letra	Procesador	Memoria RAM	Modelo	GPU	Sistema Operativo	Tiempo de procesamiento
C	Core i7 – 7700HQ	12 GB	ASUS GL553VE	GTX – 1050 TI / 4 GB	Windows 10 Home	4:48:22 minutos
C	Core i5 – 2450M	6 GB	Lenovo z570	Geforce GT 640M	Ubuntu 18.04	7:50:25 minutos

Fuente: Propia

Posteriormente se realizó una prueba de falsos positivos de 50 frames por letra, teniendo un ponderado de este para sacar un valor final ya sea falso positivo, falso negativo y finalmente obtener el valor de incertidumbre por cada letra mostrado a continuación en la matriz de confusión:

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

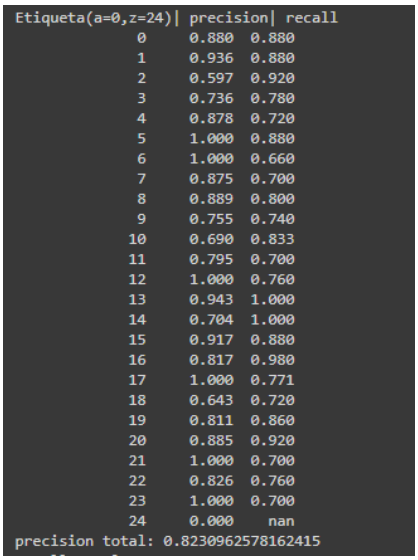
4.1.1. Matriz de Confusión

Para hallar la matriz de confusión se usó los datos capturados en un Excel, trabajados en Colab de la siguiente manera, usando las bibliotecas de Pandas, Matplotlib, Seaborn y Numpy se obtuvo el overall accuracy considerando una precisión de multiclase que utiliza la siguiente formula:

$$AC = \frac{Mii}{\sum_j Mji}$$

Figura 4. 1.

Accuracy por letra y accuracy overall



Etiqueta(a=0,z=24)	precision	recall
0	0.880	0.880
1	0.936	0.880
2	0.597	0.920
3	0.736	0.780
4	0.878	0.720
5	1.000	0.880
6	1.000	0.660
7	0.875	0.700
8	0.889	0.800
9	0.755	0.740
10	0.690	0.833
11	0.795	0.700
12	1.000	0.760
13	0.943	1.000
14	0.704	1.000
15	0.917	0.880
16	0.817	0.980
17	1.000	0.771
18	0.643	0.720
19	0.811	0.860
20	0.885	0.920
21	1.000	0.700
22	0.826	0.760
23	1.000	0.700
24	0.000	nan

precision total: 0.8230962578162415

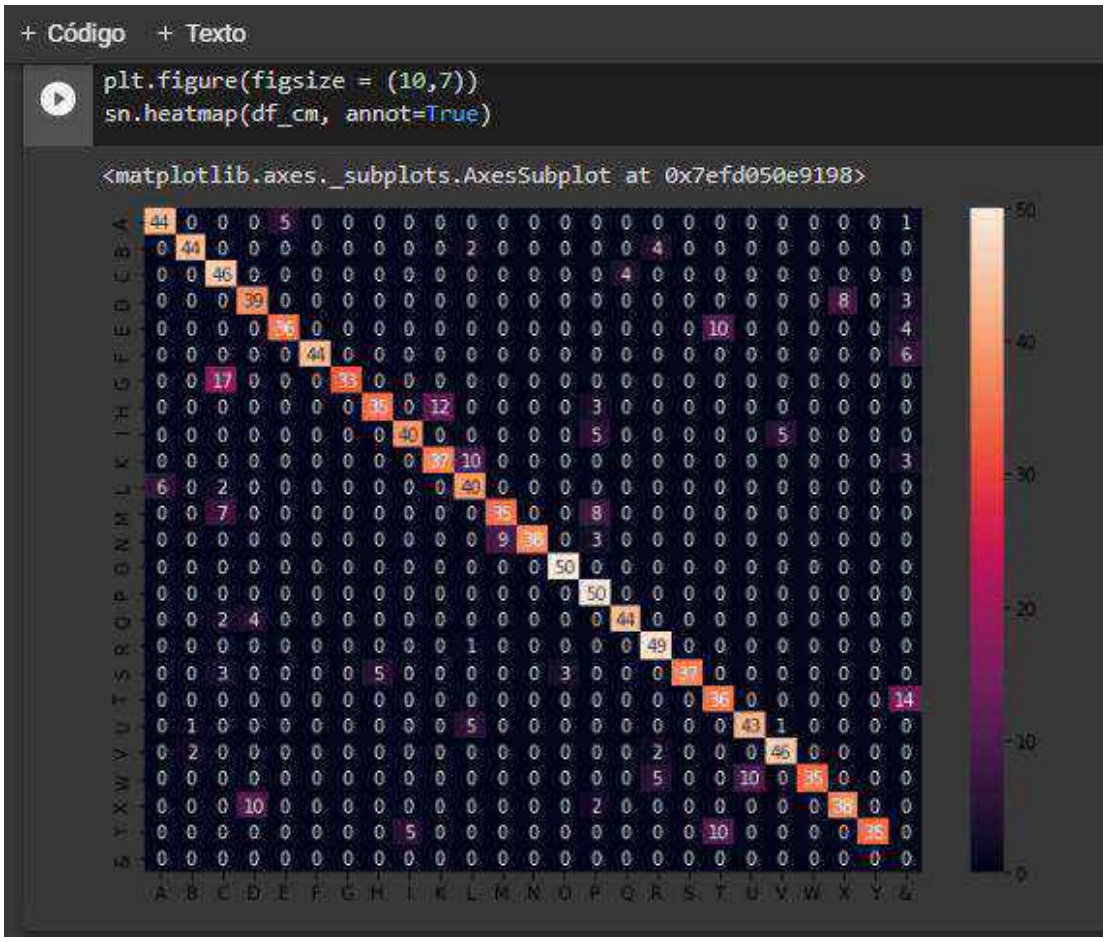
Fuente: Propia

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

Figura 4. 2.

Matriz de confusión de resultados



Nota. Se obtuvo que el Accuracy Overall, o precisión general es del: 82.3 %. Fuente: Propia

Se obtuvo un accuraccy general del 82.3 % con respecto a la clasificación de las LSP en el prototipo.

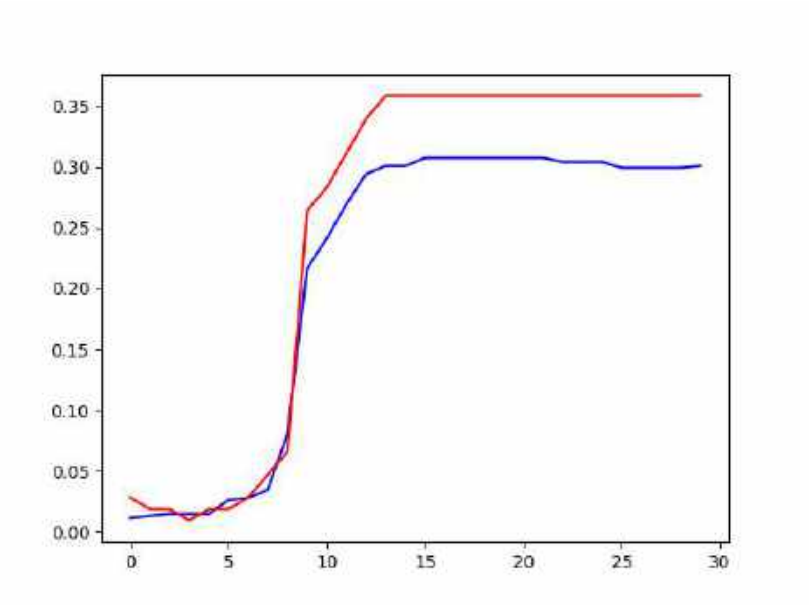
Con respecto a la precisión en la Red Neuronal de la fase II, se obtiene los siguientes cuadros de accuracy de entrenamiento versus test, siendo la línea roja de test y azul de entrenamiento, se observa en la figura 4.3 un cuadro con overfitting y ya resuelto el problema en la figura 4.4.

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

Figura 4. 3.

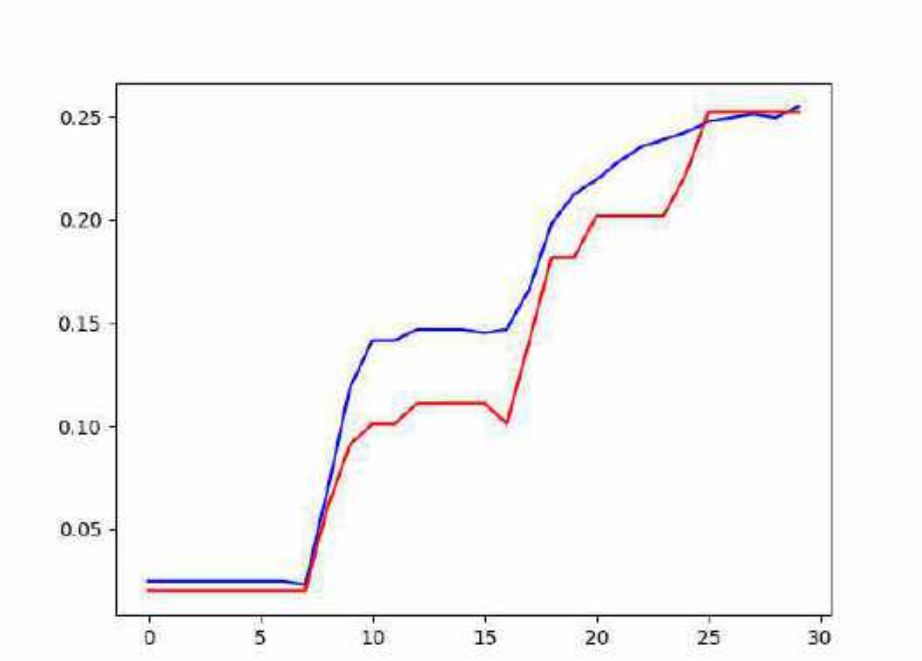
Cuadro de Test versus Entrenamiento con overfitting



Fuente: Propia

Figura 4. 4.

Cuadro de Test versus entrenamiento, resuelto el ajuste de overfitting



Fuente: Propia

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

4.2. Cumplimiento de objetivos

4.2.1. Desarrollar un prototipo de traductor de lengua de señas peruanas básicas estáticas utilizando machine learning.

En este trabajo de investigación se desarrolló un prototipo de traductor de lenguaje de señas peruanas básicas estáticas que identifica cada letra del alfabeto de lenguaje de señas peruano (LSP) en un tiempo aceptable.

El prototipo funcional desarrollado utilizó redes neuronales convolucionales con estructuras algorítmicas de entrada y salida justificando un modelo supervisado de CNN, una red neuronal estándar y la máquina de vectores de soporte para la fase II, por ende, podemos afirmar que se utiliza la tecnología de machine Learning por lo que cumple con nuestro objetivo general.

4.2.2. Obtener los requerimientos principales para la construcción del prototipo

Se obtuvo los requerimientos principales para la construcción del prototipo, teniendo reuniones con el director del CEBE Don José de San Martin y con las docentes encargadas de las aulas de los niños con discapacidad auditiva (D.A), se realizaron encuestas que se puede evidenciar en los anexos nº 1, 2, 3, 4, 5, 6, 7,8 y 9.

4.2.3. Realizar el diseño rápido del prototipo

Se realizó el diseño rápido del prototipo realizando un mockup de como seria la interfaz final para el usuario, como se puede ver en la figura n° 3.1, que esta descrita en el capítulo 3.3 Modelado.

4.2.4. Construcción del prototipo inicial.

Se realizó la construcción del prototipo inicial, primero detectando las curvaturas y direcciones de cada dedo, además de los rizos para poder entrenar las imágenes estáticas de las letras del abecedario del lenguaje de señas peruano (LSP), seguidamente se realizó entrenamientos sucesivos con videos de aproximadamente 10 segundos aproximadamente para obtener los puntos clave para que detecte la letra del alfabeto peruano, esto se detalla en capítulo 3, ítem 3.4.1.

4.2.5. Evaluar el prototipo en función al usuario.

Se evaluó el prototipo en función al usuario, se determinó en que tiempo se hará la captura de la webcam para que los frames sean procesados con la red neuronal y máquinas de vectores de soporte (Support vector machine SVM), y así obtener el video de salida de la letra obtenida que se mostró al usuario, esto se detalla en el capítulo 3, ítem 3.4.2

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

4.2.6. Refinar el prototipo

Se refino el prototipo para reducir errores en el reconocimiento de las letras del abecedario, modificando el algoritmo del archivo DeterminacionPositcion.py, y del método principal, para que se obtengan mejores resultados y puedan ser de mayor precisión, se re entreno muchas veces la red, y se usó diferentes videos, además de correcciones en el algoritmo de las poses y del algoritmo principal de procesamiento, esto se detalla en el capítulo 3, ítem 3.4.3. Tomar en consideración el error de incertidumbre con el que cuentan todos los sistemas de Machine Learning, ninguno es 100% certero, más se trató de llegar a un umbral aceptable detallado en el capítulo IV, item 4.1.1 de resultados.

4.2.7. Implementar el prototipo en el CEBE

Para la implementación se hizo la entrega mediante un acta con el director Lic. Helmer del Pozo Cruz la entrega del prototipo se hace en un USB de 32 GB además de la implementación del mismo detallado en capítulo III, ítem 3.6.2.

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

4.3. Contribuciones

El reconocimiento dinámico de gestos con la mano entre otras modalidades de gestos es un paso importante hacia el reconocimiento de la lengua de signos, así como cualquier aplicativo de interacción humano-ordenador; La investigación permite incluso poder procesar letras estáticas de manera secuencial, dado que en el futuro pueda implementar palabras.

Al realizar el prototipo de traductor de lenguaje de señas peruanas básicas, se contribuye en la traducción del alfabeto de señas estáticas, esto posibilita que las personas involucradas con los niños con discapacidad auditiva (D.A) del CEBE Don José de San Martin puedan entender al menos alguna urgencia o algo importante que los niños quieran expresar.

Se contribuye también a futuras investigaciones que puedan ofrecer una mayor precisión y un catálogo con mayor cantidad de señas, asimismo se ofrece el prototipo como un sistema de libre acceso para investigación y educación.

Ya que el prototipo usa bibliotecas open source y puede correr en un sistema operativo libre reduce costos de implementación y de pago de licencias a futuro; Además de que no requiere ninguna cámara especial de reconocimiento de profundidad como es la Kinect u otras, esto contribuye al uso de software libre y el ahorro económico del colegio en cuanto a la implantación futura de este tipo de sistemas.

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

4.4. Recursos y Presupuesto

Tabla 7.

Tabla de costos de elaboración del prototipo

RUBRO	CANTIDAD	PRECIO UNITARIO	COSTO TOTAL
ACTIVO FIJO			
Equipos			
LAPTOP LENOVO Z570	1	S/ 2,800.00	S/ 2,800.00
LAPTOP ASUS ROG STRIX GL553VE	1	S/ 4,700.00	S/ 4,700.00
Total Equipo			**S/ 7,500.00**
Herramientas			
Python	1	S/ 0.00	S/ 0.00
OpenCV	1	S/ 0.00	S/ 0.00
Ubuntu	1	S/ 0.00	S/ 0.00
TensorFlow	1	S/ 0.00	S/ 0.00
Keras	1	S/ 0.00	S/ 0.00
Total de Herramientas			**S/ 0.00**

Mano de Obra	CANTIDAD	MESES/MEDIO TIEMPO	PRECIO MES	PRECIO TOTAL
Programadoras **Full Stack**	1	4	S/ 2,500.00	S/ 10,000.00
Administrador de Data	1	3	S/ 2,000.00	S/ 6,000.00
Analista de requerimientos	1	2	S/ 2,500.00	S/ 5,000.00
Diseñador de Prototipo	1	1	S/ 1,800.00	S/ 1,800.00
Total mano de Obra				**S/ 22,800.00**

Costos Indirectos	CANTIDAD	MESES	PRECIO MES	PRECIO TOTAL
Internet	1	18	S/ 120.00	S/ 2,160.00
Servicio de Luz	1	18	30	S/ 540.00
Papel bond A4	1paquete	-	20	S/ 20.00
Total de Costos Indirectos				**S/ 2,720.00**

RUBROS	
Equipos	S/ 7,500.00
Herramientas	S/ 0.00
Mano de Obra	S/ 22,800.00
Costeos Indirectos	S/ 2,720.00
Total	**S/ 33,020.00**

Fuente: Propia

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

GLOSARIO

ACAT: Software de código abierto desarrollado por Steven Hawkings, sus siglas significan: Kit de herramientas de asistencia sensible al contexto (Assistive Context-Aware Toolkit), se usa para permitir a las personas con discapacidades motrices poder comunicarse.

ANNs: Artificial Neural Networks.

API: Es un conjunto de funciones y procedimientos que permite la conexión de entre un aplicativo y otro, además de poder enviar información a donde se requiere y recibirla de la misma manera.

Batch: El tamaño del lote que define la cantidad de muestras que se propagarán a través de la red. Por ejemplo, supongamos que tiene 1050 muestras de entrenamiento y desea configurar un batch_size igual a 100. El algoritmo toma las primeras 100 muestras (del 1 al 100) del conjunto de datos de entrenamiento y entrena la red.

BGR: Es el formato que usan ciertas bibliotecas invirtiendo los colores RGB.

Blender: Es un programa multiplataforma que permite desarrollar modelados y animaciones 3D.

CEBE: Centro de educación básica especial.

Clase: En este contexto llamamos clase a cada pose definida, aquel dato que se quiere clasificar, ubicar en diferentes “clases”.

CMYK: Es una composición de colores que se basa en los colores de impresión que son Cyan, Magenta, Yellow y Key (Negro).

CNN: Red Neuronal Convolucional.

Colab: Es un proyecto de Google para la colaboración de proyectos de programación en Python y DataScience.

Convolución: La convolución definida matemáticamente es una operación que permite obtener la salida de una suma de funciones o ante una entrada conocida.

CPP: Redes Neuronales Competitivas.

D.A: Discapacidad auditiva.

Data Science: En español llamado ciencia de datos, una ciencia que se encarga del estudio de los datos y en computación se encarga de extraer cantidades enormes de información para poder procesarla o darle uso en la informática.

Dataset: Conjunto de datos digitales que se pueden manipular y usar.

Epoca: Una época es un término utilizado en el aprendizaje automático e indica el número de pasadas de todo el conjunto de datos de entrenamiento que ha completado el algoritmo de

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

aprendizaje automático. Los conjuntos de datos generalmente se agrupan en lotes (especialmente cuando la cantidad de datos es muy grande). Algunas personas usan el término iteración libremente y se refieren a pasar un lote por el modelo como una iteración.

FF: Redes Neuronales Forward.

Formato JPG: Formato de compresión que se usa generalmente para imágenes, no guarda transparencias.

Formato mp4: Formato de compresión que se usa generalmente para videos.

Formato PNG: Formato de compresión que se usa generalmente para imágenes, guarda transparencias.

Frame: Es la imagen instantánea que se toma de un video.

Función de activación: la Función de Activación de un nodo define la salida de un nodo dada una entrada o un conjunto de entradas. Se podría decir que un circuito estándar de computador se comporta como una red digital de funciones de activación al activarse como "ON" u "OFF".

Gradient Boosting: Es una técnica de aprendizaje automático, se usa en el análisis de regresión y data science.

Hiperplano: Un hiperplano viene definido como la recta que separa los datos de diferentes clases, en el algoritmo de clasificación de SVM.

HSI: (Hue, Saturartion, Intensity)- (Matiz, Saturacion, Intensidad), define un modelo de color en términos de sus componentes constituyentes.

HSV: El modelo HSV, también llamado HSB, define un modelo de color en términos de sus componentes, del inglés Hue, Saturation, Value - Matiz, Saturación, Valor.

I.A: Inteligencia artificial.

Imagen 2D: Se refiere a una imagen plana sin profundidad, una imagen común y corriente como una fotografía, basada solamente en los ejes X, Y.

INEI: Instituto nacional de estadística e informática.

Interfaz: Es la vista de un sistema de información que se ofrece al usuario con la cual puede interactuar.

JetBrains: Es una compañía de desarrollo de software.

Kinect: Es un dispositivo electrónico tipo cámara que se conecta a través de periféricos externos a un PC o una videoconsola, usado normalmente en experiencia de videojuegos.

K-means: Es un tipo de algoritmo de clasificación que se basa en el vecino más cercano.

Licencia BSD: Es la licencia de software otorgada principalmente para los sistemas BSD (Berkeley Software Distribution), un tipo del sistema operativo Unix-like.

LSP: Lenguaje de señas peruano.

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

Mapa de activación: La salida de cada operación de convolución produce una salida 2D llamada mapa de activación.

Matlab: Es un programa que ofrece un entorno para desarrollar computo numérico.

Matriz de confusión: Es una herramienta que se usa normalmente para el campo de la I.A, permite la visualización del desempeño de un algoritmo que se emplea en aprendizaje supervisado.

Mixamo: Es un programa que permite utilizar personajes 3D y darles un ringing o animación personalizado.

MockUp: Es un modelado que se realiza de un sistema de información o programa parecido a un boceto, se podría decir que es el diseño final de interfaces no funcional.

Motor OCR: Es un tipo de programa que reconoce texto a partir de imágenes.

Open Source: En este caso hablamos de un software que puede ser modificado por las personas y en algunos casos compartido y de dominio público. Muchas veces sin costo alguno.

PaaS: Es un entorno de desarrollo e implementación completo en la nube, con recursos que permiten entregar todo, desde aplicaciones sencillas basadas en la nube hasta aplicaciones empresariales sofisticadas habilitadas para la nube.

Pixel: La unidad mínima de una imagen digital que representa un color.

Plano 3D: Se refiere a un sistema de coordenadas que incluya los ejes X, Y, Z.

Procesador: Componente electrónico que se encarga del procesamiento principal del motherboard u otra placa, unidad central del procesamiento.

RELU: Una de las muchas funciones de activación usadas en Machine Learning.

RGB: Es una composición de colores que usa un computador en función al rojo, verde y azul.

SDLC: Ciclo de vida del desarrollo de sistemas de información.

Seaborn: Biblioteca que facilita las interfaces gráficas y de ploteo en Python.

Seña: Es un movimiento o pose con significado.

Sistema Operativo: Es el sistema principal y funcional de un computador en el cual pueden correr programas, como por ejemplo Windows o Linux Ubuntu, Linux Debian, Linux Cent Os, entre otros.

Streamming: Es una tecnología que permite ver o escuchar, sin descargar datos previamente algún video o audio, en este caso la captura de la webcam se consideraría una transmisión de frames en streaming, ya que muestra las imagines en tiempo real sin necesidad de descargarlas previamente.

Tarjeta gráfica: Es una tarjeta que usualmente usa un computador para llevar a cabo los procesos de procesamiento de imagen y video.

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

VSM: Máquinas de vectores de soporte (Support vector machine), es um conjunto de algoritmos computacionales de aprendizaje supervisado que se usa en la clasificación de información o imágenes.

YCbCr: Es una familia de espacio de color usada en sistemas de vídeo y fotografía digital El modelo YIQ define un espacio de color, usado por el estándar de televisión NTSC (National Television System Committee).

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

CONCLUSIONES

La investigación del prototipo de traductor que se realizó utilizando machine learning, ayuda a traducir las señas básicas estáticas del lenguaje de señas peruano (LSP), permite que las personas involucradas con niños que tienen discapacidad auditiva (D.A) puedan comunicarse con las personas que no conocen el lenguaje de señas peruano (LSP) dentro de su entorno; Además que al usar machine Learning automatiza los procesos de clasificación de la seña y agiliza procesos.

Mediante reuniones, encuestas con el director y docentes encargadas de niños con discapacidad auditiva (D.A) del CEBE Don José de San Martin, se logró obtener requerimientos principales para la construcción del prototipo, concluyendo que como mínimo se debe reconocer las señas del alfabeto peruano.

Por medio de la realización de un diseño rápido del prototipo se pudo visualizar como seria la interfaz para el usuario final que son las personas involucradas con los niños con discapacidad auditiva (D.A).

Con la construcción del prototipo inicial, nos permite hacer el reconocimiento de imágenes estáticas, se trabajó con una imagen estática de cada letra del alfabeto peruano, donde se identificó la curvatura y direcciones, orientaciones y rizos para cada dedo que corresponde a una pose del alfabeto peruano.

La evaluación del prototipo en función al usuario determinó que tiempo tendría que hacerse las capturas de la webcam para que el reconocimiento de la seña sea factible y al ser procesado posteriormente con la red neuronal y las máquinas de vectores de soporte (Support vector machine SVM), ofrezca una salida coherente y rica en información, con el menor error de incertidumbre posible, siendo este tiempo de entre 7 – 10 segundos, tiempo que se estimó gracias a la tabla n°6.

Se refino el prototipo para reducir errores en el reconocimiento de las letras del abecedario, modificando el algoritmo del archivo DeterminacionPositcion.py, del método principal, para que se obtengan mejores resultados, con una mayor precisión por letra en el reconocimiento de las letras del alfabeto.

Con la implementación del prototipo en el CEBE detallado en el anexo n°63, se concluye que el trabajo de investigación fue satisfactorio cumpliendo además con sus objetivos y requerimientos del colegio obtenidos inicialmente.

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

RECOMENDACIONES

a. Se recomienda hacer uso del dataset de 7GB que proporciona Universidad de Friburgo, ya que este dataset es pequeño y vine con todo lo necesario para realizar nuestros entrenamientos.
b. Se debe aplicar los filtros de tipo cv2.ADAPTIVE_THRESH_GAUSSIAN_C, ya que fueron los que mejor resultado emitieron para tener una imagen, más nítida y que haga el reconocimiento del contorno de la mano sin tener detalles que confundan al algoritmo como rugosidades o imperfecciones de la palma.
c. Para el procesamiento de las imágenes entrenadas con la red neuronal de Zimmerman, se recomienda tener una tarjeta gráfica de gama media a alta que incluya CUDA, como por ejemplo una GTX 1050 TI, y un procesador de 2.80 GHZ a más velocidad como por ejemplo una Core i7 de séptima generación para adelante o una AMD Ryzen 5 para que los resultados sean mejores en términos de velocidad de procesamiento, y los tiempos de trabajo sean cortos.
d. Se recomienda utilizar una red neuronal Convolucional, pues nos permite procesar la información por capas teniendo ya resultados complejos de una capa a otra capa, permitiendo a los algoritmos de las capas ocultas procesar de manera más eficiente la clasificación de las imágenes y al final combinar información en las capas de salida de la red.
e. Se recomienda usar imágenes no tan pesadas que usen el formato .jpg y en dimensionalidades de 320x320 pixeles como máximo.
f. Se recomienda correr el prototipo en un ambiente que contenga la iluminación adecuada, mostrando la pose de forma correcta para no producir errores de punto de vista, ni ambientes con escenarios turbios o saturados de muchas cosas y colores, pues podría producir errores como se muestra en la figura 2.2 que puede confundir al algoritmo y llevar a errores de incertidumbre altos.
g. Para un reconocimiento multiclase sencillo, en las pruebas se realizaron una comparativa usando una red neuronal estándar y SVM, los mejores resultados para la fase II se obtuvieron con SVM, por lo que recomendamos el uso de este algoritmo de clasificación para tareas similares.
h. Se recomienda el uso de paradigma de prototipo tipo parches, ya que permite hacer uso de diferentes algoritmos para construir un todo funcional, es un paradigma sencillo de comprender y ayuda a organizar mejor la construcción de un prototipo tipo algorítmico.

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

i. Se recomienda el lenguaje de programación Python3, debido a las bibliotecas con las que cuenta que son las más estables, y nos ayudaron para trabajar correctamente con diferentes librerías de machine learning.
j. Se recomienda a futuras investigaciones utilizar una Plataforma como servicio (PAAS), ya que en el transcurso para realizar entrenamiento de datos es muy complejo, se necesita de buen soporte de hardware para no tener problemas cuando se realice este tipo de investigaciones.
k. Se recomienda revisar el código del repositorio de GitHub ubicado el link en el anexo nª 62.c para futuros trabajos con señas dinámicas o aplicativos de palabras

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

REFERENCIAS

Documentation | CMake. (s. f.). Recuperado 2 de agosto de 2020, de https://cmake.org/documentation/

Gavali, P., & Banu, J. S. (2019). Deep Convolutional Neural Network for Image Classification on CUDA Platform. En *Deep Learning and Parallel Computing Environment for Bioengineering Systems* (pp. 99-122). Elsevier. https://doi.org/10.1016/B978-0-12-816718-2.00013-0

Hand, D. J. (2010). Machine Learning: An Algorithmic Perspective by Stephen Marsland. *International Statistical Review*, *78*(2), 325-325. https://doi.org/10.1111/j.1751-5823.2010.00118_11.x

Hernández Sampieri, R., Fernández Collado, C., & Baptista Lucio, P. (2014). *Metodología de la investigación*.

Kendall, K., & Kendall, J. (2011). *ANÁLISIS Y DISEÑO DE SISTEMAS* (Octava Edicion). Pearson Education.

Lee, M. (s. f.). *pytesseract: Python-tesseract is a python wrapper for Google's Tesseract-OCR* (0.3.4) [Python]. Recuperado 2 de agosto de 2020, de https://github.com/madmaze/pytesseract

Matplotlib: Python plotting—Matplotlib 3.3.0 documentation. (s. f.). Recuperado 2 de agosto de 2020, de https://matplotlib.org/

Mercado Polo, D., Pedraza Caballero, L., & Martínez Gómez, E. (2015). Comparación de Redes Neuronales aplicadas a la predicción de Series de Tiempo. *Prospectiva*, *13*(2), 88. https://doi.org/10.15665/rp.v13i2.491

Pillow—Pillow (PIL Fork) 7.2.0 documentation. (s. f.). Recuperado 2 de agosto de 2020, de https://pillow.readthedocs.io/en/stable/

DERECHOS DE AUTOR RESERVADOS

Universidad Andina del Cusco
Repositorio Digital

Pressman, R. (2010). *Ingeniería del software.Un enfoque práctico* (Séptima Edición). The McGraw-Hill Companies.

PyCharm: Uno de los mejores IDE para Python. (2018, enero 31). *Escuela de Python*. https://www.escuelapython.com/pycharm-uno-de-los-mejores-ide-para-python/

Qué es Git. (s. f.). CódigoFacilito. Recuperado 30 de septiembre de 2020, de https://codigofacilito.com/articulos/que-es-git

Rosebrock, A. (2017). *Deep Learning for Computer Vision with Python* (1st ed.). PYIMAGESEARCH.

Scikit-learn: Machine learning in Python—Scikit-learn 0.23.1 documentation. (s. f.). Recuperado 2 de agosto de 2020, de https://scikit-learn.org/stable/

SciPy.org—SciPy.org. (s. f.). Recuperado 2 de agosto de 2020, de https://www.scipy.org/

Sharma, A. R., Beaula, R., Marikkannu, P., Sungheetha, A., & Sahana, C. (2016). Comparative study of distinctive image classification techniques. *2016 10th International Conference on Intelligent Systems and Control (ISCO)*, 1-8. https://doi.org/10.1109/ISCO.2016.7727002

Xu, B., Wang, N., Chen, T., & Li, M. (2015). Empirical Evaluation of Rectified Activations in Convolutional Network. *arXiv:1505.00853 [cs, stat]*. http://arxiv.org/abs/1505.00853

Zimmermann, C., & Brox, T. (2017). Learning to Estimate 3D Hand Pose from Single RGB Images. *arXiv:1705.01389 [cs]*. http://arxiv.org/abs/1705.01389

DERECHOS DE AUTOR RESERVADOS