ISBN: 978-628-95207-4-3. ISSN: 2414-6390. Digital Object Identifier: 10.18687/LACCEI2023.1.1.552

Artificial verification system on the use of safety elements

Leon Leon, Ryan[1]; Esparza Salazar, Kattia[2]; Medina-Coloma, Adriana[3]; Rojas Bacilio, Jennifer[4]

[1-4]Universidad Privada del Norte, Perú, ryan.leon@upn.edu.pe, N00167278@upn.pe, N00107105@upn.pe, N00185508@upn.pe,

Abstrac– This article shows the development process of a computer vision application for stationary computers through the Python 3.10.8 software, which verifies and detects if the person uses a surgical mask or cap (hair net or cap) or none of the surgical implements. safety, with the aim of identifying and supervising its correct use in work in the food industry. To start with the development of this program, libraries such as opencv, matplotlib, mediapipe, numpy, imutils and os were installed and used. Thus, these helped to detect faces and 3 programs were created: a program to obtain images (data) by means of video capture of images through the laptop camera, a neural network training program with the collected images and a program that detects and identifies the use or not of these implements. As results, after several tests, a 93.33% validation was obtained.

Keywords— Python; artificial vision; safety; bookstores

.

.

Digital Object Identifier: (only for full papers, inserted by LACCEI).
ISSN, ISBN: (to be inserted by LACCEI).
DO NOT REMOVE

Sistema de verificación artificial sobre el uso de elementos de inocuidad

Leon Leon, Ryan[1]; Esparza Salazar, Kattia[2]; Medina-Coloma, Adriana[3]; Rojas Bacilio, Jennifer[4]

[1-4]Universidad Privada del Norte, Perú, ryan.leon@upn.edu.pe, N00167278@upn.pe, N00107105@upn.pe, N00185508@upn.pe,

***Resumen*– En el presente artículo se muestra el proceso de desarrollo de una aplicación de visión artificial para computadoras fijas, por medio del software Python 3.10.8, que verifica y detecta si la persona utiliza mascarilla quirúrguica o gorra (red de cabello o cofia) o ninguno de los implementos de inocuidad, con el objetivo de identificar y supervisar su correcto uso en trabajos de industrias alimenticias. Para empezar con el desarrollo de este programa, se instalaron y usaron librerías como opencv, matplotlib, mediapipe, numpy, imutils y os. Así, estos ayudaron a la detección de rostros y se crearon 3 programas, un programa para la obtención de imágenes (data) por medio de la videocaptura de imágenes a través de la cámara de la laptop, un programa de entrenamiento de red neuronal con las imágenes recopiladas y un programa que detecta e identifica el uso o no de estos implementos. Como resultados, después de varias pruebas se obtuvo un 93.33% de validación.**

***Palabras clave*– Python; vision artificial; inocuidad; librerías.**

I. Introducción

La situación económica actual en el Perú y en la mayoría de los países de Sudamérica donde la informalidad es común, exige controles más estrictos en los estándares de seguridad e inocuidad para las empresas que se dedican al sector alimentario [1]. No solo en las pequeñas empresas sino también en las grandes se deben fomentar las medidas correspondientes, ya que incluso en panificadoras conocidas se han presentado incidentes de falta de indumentaria e higiene en las instalaciones, lo que hace que afecte directamente a la salud pública [2]. Tal es el caso, de las panaderías de San Miguel - Lima las cuales fueron clausuradas debido a que el personal no contaba con los equipos de protección establecidos para la manipulación de alimentos [3]. Asimismo, en Tacna se realizó un operativo en dos locales donde los que elaboraban el pan no contaban con la indumentaria y carnet de sanidad por lo que se realizó una clausura preventiva de 5 días [4].

En relación a ello, este trabajo implementa una solución económica haciendo uso de un software gratuito que verifique el empleo adecuado de los artículos de inocuidad personal dentro de una industria alimenticia. Es necesario tener en cuenta que en todas las industrias existe la necesidad de mejorar la seguridad y la productividad de los trabajadores, gestionando un plan de seguridad y salud ocupacional [5]. El 44% de las panaderías Mype encuestadas en Lima Metropolitana utilizan gorras de cocina, el 23.3% utiliza mandiles. Asimismo, el 22% emplea sus guantes blancos al laborar. Mientras que el uso de mascarilla bucal conforma uno de los elementos menos utilizados con un 8.8% [6]. Por otra parte, estos porcentajes sólo se presentan en ocasiones ya que hay momentos en los que no usan su indumentaria establecida debido a que el dueño, que se encarga generalmente de supervisar el uso de estos, pierde el cuidado [7]. Trabajar en condiciones seguras y saludables tiene el potencial de mejorar la eficiencia de los trabajadores y, por lo tanto, reducir los costos económicos de los accidentes con respecto a la inocuidad, en beneficio de las empresas [8]. Muchas organizaciones implementan la norma OHSAS 18001, el cual permite promover las buenas prácticas en la prevención de riesgos laborales, asegurando un sistema estructurado de gestión de la seguridad y salud en el trabajo [9]. Esta investigación desarrolló el software mediante Python, el cual, es un lenguaje de programación utilizado en muchos campos debido a su alto nivel en estructura de datos, multiparadigma, interpretado, dinámico y sintaxis sencilla [10]. Además, se usó OpenCv que es una librería de código abierto para el procesamiento de imágenes y visión por computadora con diferentes sistemas operativos [11]. Así también, se empleó el algoritmo LBPH por su alta eficiencia en el reconocimiento facial gracias a su simplicidad computacional de analizar imágenes en tiempo real y su poder discriminativo [12].

Esto se evidencia en un trabajo de investigación que tuvo como objetivo principal implementar un sistema de reconocimiento facial utilizando el algoritmo PCA con C++ y otro sistema con Python 2.7 con la biblioteca OpenCV, este último demostró que tiene un mayor rendimiento que el algoritmo original [13]. Asimismo, en otro proyecto basado en el análisis de un sistema enfocado solo en el reconocimiento facial a partir de una base de datos realizado en Python 3.6 junto a la visión por computador OpenCV, obteniendo como resultado un 95.11% de ratio de éxito [14].

Del mismo modo, otro estudio aplicó el Deep learning en Python para el reconocimiento facial inclusive cuando se utilizan mascarillas y/o lentes, teniendo un dataset de 2400 fotografías de los participantes con lo que lograron un entrenamiento de la red neuronal en Google colab de 0.2 y un porcentaje de acierto obtenido de la aplicación en la segunda fase de 71% [15]. Por otra parte, en un artículo de investigación se evidenció el desarrollo de una aplicación web

Digital Object Identifier: (only for full papers, inserted by LACCEI).

ISSN, **ISBN:** (to be inserted by LACCEI).

para rastrear el uso de mascarillas en entornos públicos utilizaron el marco Flask de Python con el algoritmo Haar Cascade para clasificar rostros con o sin mascarilla, lo que permitió el almacenamiento de bases de datos, teniendo un alcance de precisión del 63% [16]. Además, un estudio orientado a la creación de un sistema de visión por computador que registre indicadores del uso correcto de EPPs como cascos y chalecos de seguridad para la industria de la construcción, logró reducir la necesidad de monitoreo y supervisión manual, a través de una arquitectura de red neuronal que opera sobre imágenes digitales utilizando CV y técnicas de aprendizaje automático [17]. Así también, en otro trabajo de investigación se creó un método de reconocimiento facial accesible para la gestión de asistencia de alumnos de una institución educativa mediante el uso del algoritmo LBPH, el cual se reconoció como uno de los más óptimos encontrados dentro del Open CV debido a que su factor de confianza es de 2-5 y su bajo nivel de interferencia de ruido [18]. Por otro lado, en el estudio de investigación construyeron una red neuronal convolucional entrenada con el software Cascade Trainer GUI en Python utilizando las bibliotecas cv2, Numpy e Imutils para detectar mascarillas en varios escenarios, teniendo una precisión del 92 % con mascarilla y 100% sin mascarilla [19].

La presente investigación propone crear un sistema de detección del uso de gorras de cocina y mascarilla quirúrgica bucal mediante el empleo de inteligencia artificial. Que nos permitirá verificar el empleo correcto de estos elementos de higiene y seguridad, con el objetivo de prevenir y reducir el número de ocurrencias de incidentes relacionados a la ausencia de estos en el desarrollo laboral de una industria alimentaria, dando inicio al desarrollo posterior de un sistema de control completo de los elementos de seguridad y salubridad necesarios para labores de diferentes tipos dentro de esta industria.

II. MATERIALES Y MÉTODOS

A. *Materiales.*

Para la realización de este proyecto se necesitarán los siguientes materiales:

TABLA I
Materiales

Material	Descripción	Cantidad
Laptop	Procesador: Intel(R) Core (TM) i5-7200U CPU @ 2.50GHz 2.71 GHz RAM instalada: 8.00 GB Tipo de sistema: Sistema operativo de 64 bits, procesador basado en x64 Calidad de la foto: 0.9 MP 16:9 (1280x720) Calidad del video: 720p 16:9 30fps Reducción de parpadeo: 50 Hz	1
Software Python 3.10.8	Es una distribución de los lenguajes de programación Python y R para computación científica (ciencia de datos, aplicaciones de Machine Learning, procesamiento de datos a gran escala, análisis predictivo, etc.). Tiene como ventaja simplificar la gestión e implementación de paquetes.	1
Recursos humanos: Grupo de personas que contribuyen con el desarrollo del software.	Planificador Encargados de planificar de manera funcional el programa.	2
	Desarrollador Encargados de ejecutar mediante Python el programa.	2
	Sujetos de prueba Personas que contribuyeron con el entrenamiento mediante la captura de imágenes.	5

B. *Procedimiento.*

Los pasos que tiene este proyecto son esenciales para su buen funcionamiento, los podemos ver en la Fig. 1 a continuación.

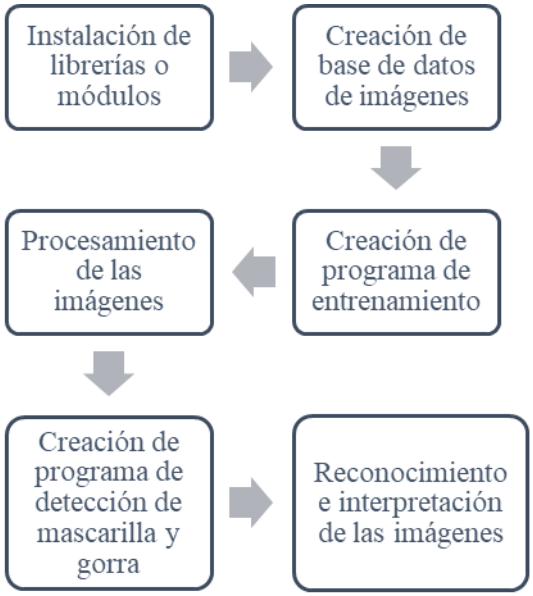

Fig. 1 Procedimiento del proyecto.

C. *Métodos.*

La metodología de trabajo de la presente investigación para el análisis del reconocimiento de elementos de inocuidad en una panadería, se realizó mediante una serie de pasos de la metodología ágil de desarrollo de software denominado SCRUM. Esta metodología se constituye de buenas prácticas, trabajo en equipo y el objetivo de tener resultados [20]. Por medio del equipo de trabajo se realizaron las actividades en 4 etapas. En la primera se definió la necesidad del desarrollo de este software. En la segunda fase, se establecieron los elementos de inocuidad a detectar por el programa. En la

tercera etapa, los desarrolladores ejecutaron la planeación del programa mediante 3 pasos (obtención de data, creación del programa y realización de pruebas) y por último en la cuarta etapa se analizaron los resultados obtenidos. En la Fig. 2 se explica el ciclo de desarrollo del programa mediante la metodología SCRUM [21]:

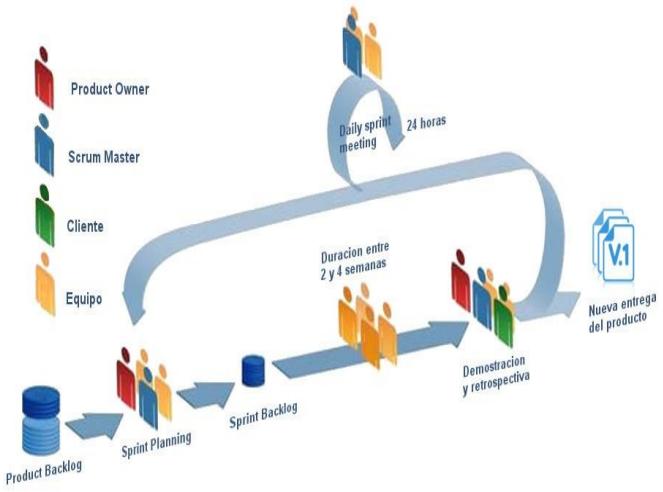

Fig. 2 Metodología SCRUM.

Finalmente se evalúan y discuten los resultados obtenidos a través del Software Python 3.10.8, el cual es un lenguaje de programación multiparadigma de código abierto, que se caracteriza por tener una sintaxis sencilla y de libre acceso. Así también, permite la integración de varias librerías para la implementación de sistemas de inteligencia artificial.

Instalación de librerías:

Para importar las librerías, se instalaron desde cmd, donde se usó el comando "pip install (nombre de la librería)" para instalar una por una. Las librerías descargadas fueron: NumPy, que es para crear vectores y matrices grandes multidimensionales, junto con una gran colección de funciones matemáticas de alto nivel para operar con ellas [22]; Matplotlib, para realizar gráficas que nos permitan mostrar imágenes [23]; TensorFlow para lograr aplicar el Deep Learning que tiene como objetivo entrenar y aportar inteligencia artificial al computador [24].

Creación de base de datos de imágenes:

Se adjuntó una base de datos de 2450 imágenes usando mascarilla, 2387 usando el gorro y 3239 sin ninguno de los implementos como podemos ver en las figuras Fig. 3, Fig. 4, Fig. 5 y Fig. 6 divididas en 3 carpetas diferentes. Estas imágenes fueron tomadas por Open CV mediante el código de colores BGR, por lo cual se tuvo que convertir a RGB para que puedan ser trabajadas por Matplotlib. Además, se tomó en cuenta que la calidad de imagen cumpla con factores como: la iluminación, resolución y orientación para facilitar la interpretación por parte de la máquina.

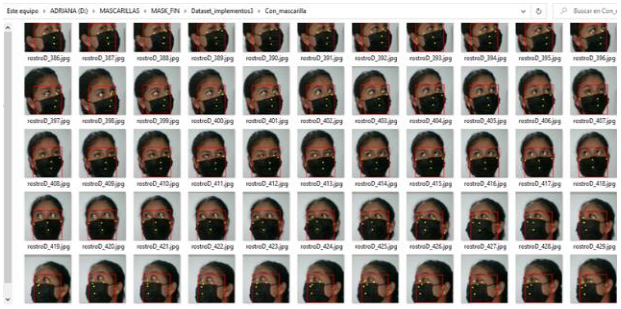

Fig. 3 Data imágenes con mascarilla.

Fig. 4 Data imágenes con gorro.

Fig. 5 Data imágenes con mascarilla y gorro (implementos).

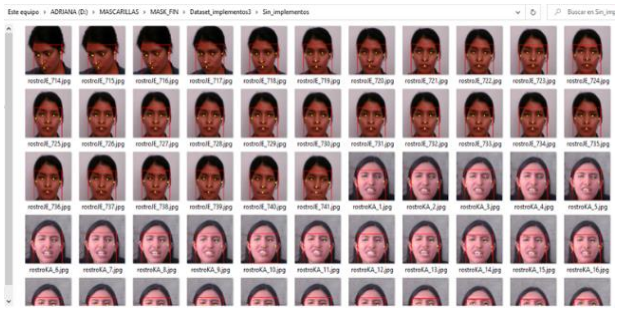

Fig. 6 Data imágenes sin mascarilla ni gorro (implementos).

Para la conversión de las imágenes a color RGB a una escala de grises se utilizó el método del promedio que consiste en calcular el promedio de los componentes RGB según la ecuación (1) de conversión de imágenes a color [25]:

$$Gris(i,j)=\{1/3\ (R(i,j)+G(i,j)+B(i,j))\} \qquad (1)$$

El computador capta las imágenes pixel a pixel, estos están formados por colores (RGB), que se clasifican en:

- R: Cantidad de rojo que tiene 1 pixel
- G: Cantidad de verde que tiene 1 pixel
- B: Cantidad de azul que tiene 1 pixel

El valor de intensidad de la imagen de color a gris puede ir de 0 a 255, donde el 0 representa el negro absoluto y el 255 el blanco absoluto [26].

Programa de entrenamiento:

En la figura Fig.7 podemos ver los pasos de cómo se entrenó el programa.

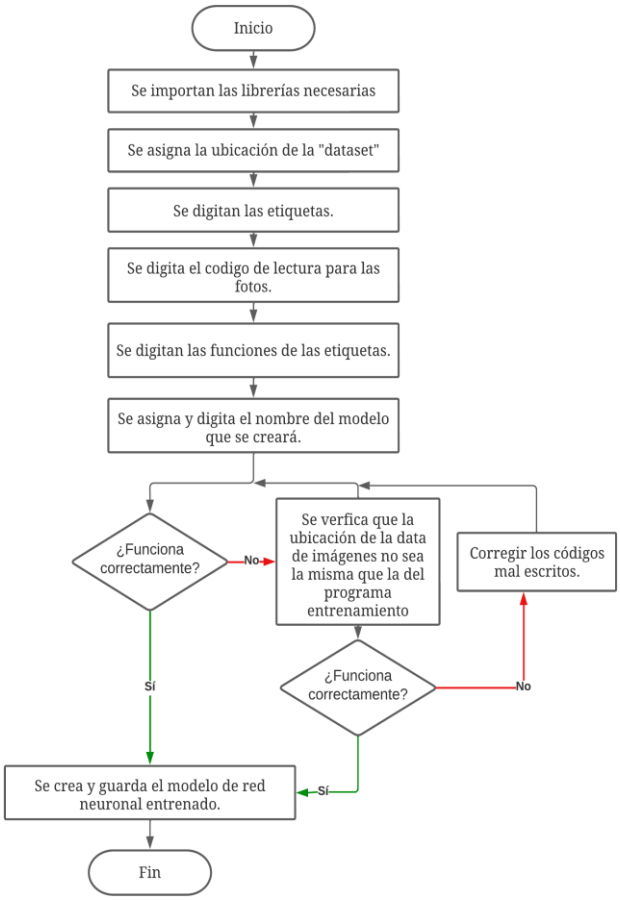

Fig. 7 Flujograma del programa de entrenamiento de imágenes de implementos de inocuidad.

Teniendo al dataset como entrada, se emplearon las redes neuronales convolucionales con el fin de obtener una variación de datos que permita que las imágenes tengan distintas formas, es decir, capten nuevas características para que pueda mejorar la interpretación de estas [27].

Para la adecuación y realce de las características de las imágenes fueron procesadas por medio de filtros dentro de los cuales tenemos el filtro Sobel, para detectar los diferentes bordes de las imágenes, basándose en el concepto de la gradiente. El gradiente de la función f en el punto de coordenadas (x, y) se define como el vector (2) [28]:

$$\nabla f = [(\partial f/\partial x)/(\partial f/\partial y)] = [(Gx/Gy)] \quad (2)$$

Tanto la magnitud como la dirección del gradiente son importantes para detectar los bordes. La magnitud (3) está dada por:

$$|\nabla f| = [G_x\text{^}2+G_y\text{^}2]^{(1/2)} \quad (3)$$

y su dirección (4) por:

$$\alpha(x,y) = \tan\text{^}(-1)(G_y/G_x) \quad (4)$$

Por otra parte, se implementó la galería media pipe para reconocer los rostros de la persona basándose en los puntos clave característicos como nariz, centro de la boca y orejas [29]. Además, se trabajó en condicionales Index, donde 0 hace referencia a las imágenes sin uso de mascarilla y gorra, 1 cuando se usa mascarilla y 2 cuando de hace uso de la gorra. Esto sirvió para que al momento que el programa se ejecute pueda etiquetar si la persona ubicada frente a la cámara se encuentra portando los implementos establecidos, observamos el procesamiento de las imágenes en las figuras Fig. 8, Fig. 9, Fig. 10 y también podemos ver parte del código de entrenamiento en la figura Fig. 11.

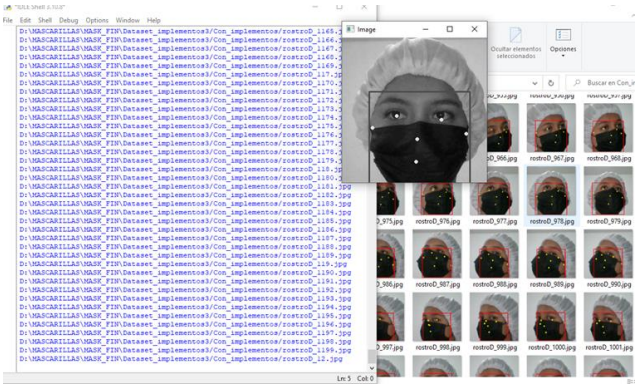

Fig. 8 Modelo procesando imágenes con mascarilla y gorro.

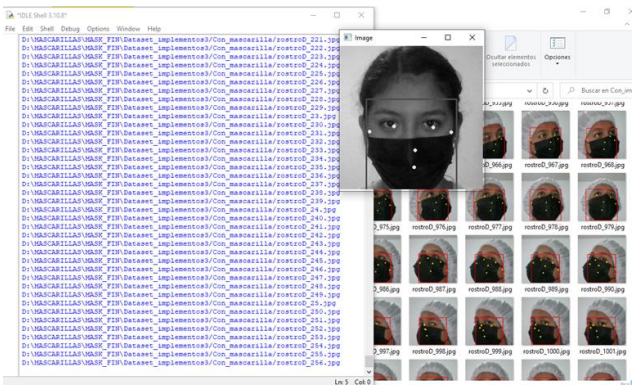

Fig. 9 Modelo procesando imágenes con mascarillas.

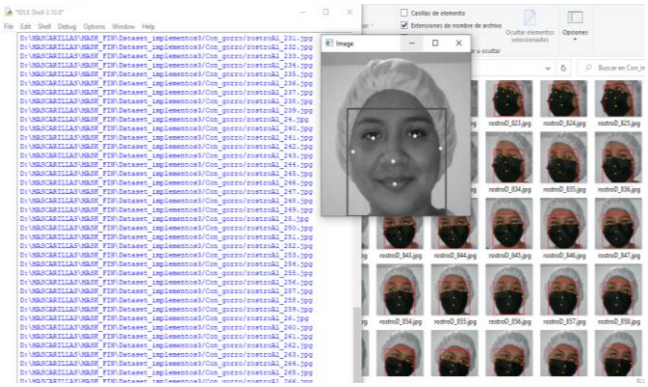

Fig. 10 Modelo procesando imágenes con gorro.

```python
import cv2
import os
import numpy as np
dataPath = "D:\MASCARILLAS\MASK_FIN\Dataset_implementos3"
dir_list = os.listdir(dataPath)
print("Lista archivos:", dir_list)
labels = []
facesData = []
label = 0
for name_dir in dir_list:
    dir_path = dataPath + "/" + name_dir

    for file_name in os.listdir(dir_path):
        image_path = dir_path + "/" + file_name
        print(image_path)
        image = cv2.imread(image_path, 0)
        cv2.imshow("Image", image)
        cv2.waitKey(10)
        facesData.append(image)
        labels.append(label)
    label += 1
print("Etiqueta 0: ", np.count_nonzero(np.array(labels) == 0))
print("Etiqueta 1: ", np.count_nonzero(np.array(labels) == 1))
print("Etiqueta 2: ", np.count_nonzero(np.array(labels) == 2))
print("Etiqueta 3: ", np.count_nonzero(np.array(labels) == 3))
# LBPH FaceRecognizer
face_mask = cv2.face.LBPHFaceRecognizer_create()
# Entrenamiento
print("Entrenando...")
face_mask.train(facesData, np.array(labels))
# Almacenar modelo
face_mask.write("MyG6.xml")
print("Modelo almacenado")

```

Fig. 11 Código del programa de entrenamiento.

Ejecución de programa de detección:

Se obtienen imágenes en tiempo real, se compara con el modelo entrenado y muestra los resultados en pantalla. Estos se ven reflejados en un recuadro que indica si la persona porta o no la mascarilla o gorra. El cuadro de color azul identifica si la persona lleva mascarilla, el cuadro de color rojo si está utilizando gorra, el cuadro de color verde si está empleando ambos implementos (gorra y mascarilla) y el de color celeste si no está utilizando ninguno de los implementos de inocuidad, podemos revisar el flujograma de la figura Fig. 12 para tener mayor detalle de la detección y el la Fig. 13 se presenta parte del código del programa de detección.

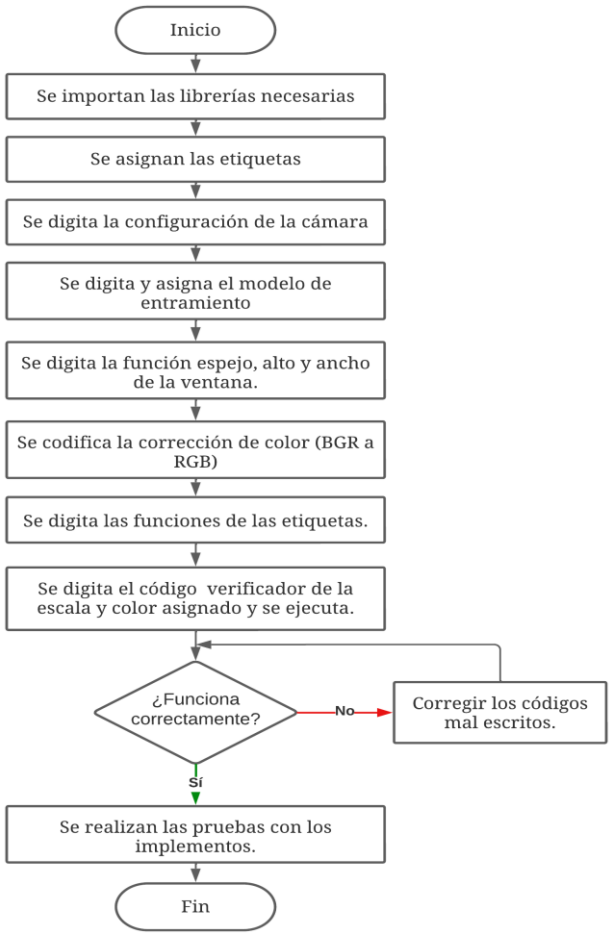

Fig. 12 Flujograma del programa de detección de implementos de inocuidad.

```python
import cv2
import os
import mediapipe as mp
mp_face_detection = mp.solutions.face_detection
LABELS = ["Con_gorro", "Con_implementos", "Con_mascarilla", "Sin_implementos"]
# Leer el modelo

face_mask = cv2.face.LBPHFaceRecognizer_create()
face_mask.read("MyG7.xml")
cap = cv2.VideoCapture(0, cv2.CAP_DSHOW)
```

Fig. 13 Primeras líneas de código del programa detector.

D. Arquitectura.

En las siguientes figuras Fig. 14 y la Fig. 15 tenemos las arquitecturas de todo el desarrollo del programa de detección de los implementos por medio del procesamiento de las imágenes a través de las redes neuronales convolucionales, para poder sacar las características de los diferentes rostros e identificar si tiene o no los implementos requeridos. Podemos ver la figura Fig. 14 la cual nos muestra la primera parte de la arquitectura para leer el modelo.

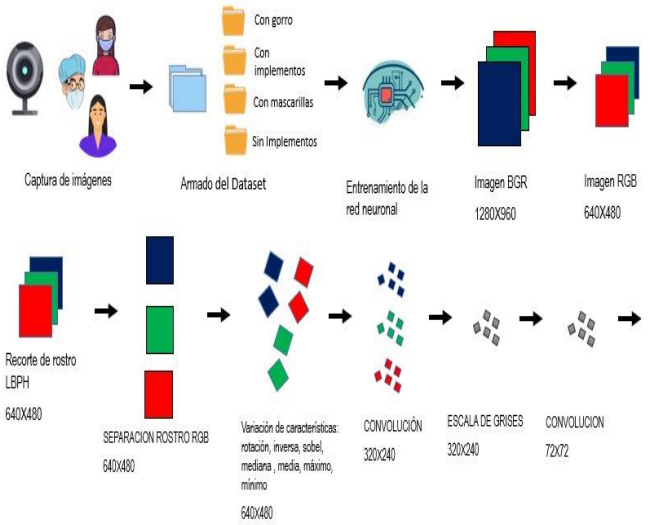

Fig. 14 Arquitectura del proyecto (primera parte).

Fig. 15 Arquitectura del proyecto (segunda parte).

III. RESULTADOS

Se desarrollaron 70 pruebas como podemos ver en las figuras Fig. 16, Fig. 17, Fig. 18 y Fig. 19 a cada sujeto en las que se alternó que 35 sean realizadas con mascarilla y/o gorra y 35 sin mascarilla y/o gorra, en esta se obtuvieron los resultados que se pueden apreciar en la Tabla II, III y IV donde se observa la cantidad de veces que el sistema ha dado un resultado correcto e incorrecto, sumado a esto se obtiene su porcentaje de validación.

"Sin_implementos"

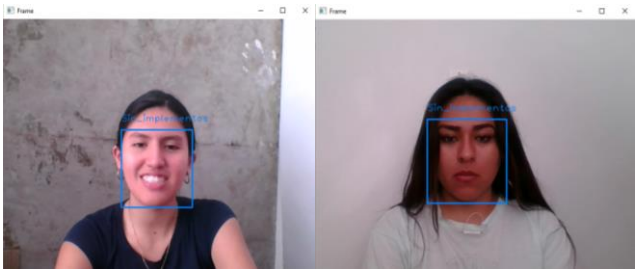

Fig. 16 Resultados de las pruebas en los sujetos sin implementos.

"Con_mascarilla"

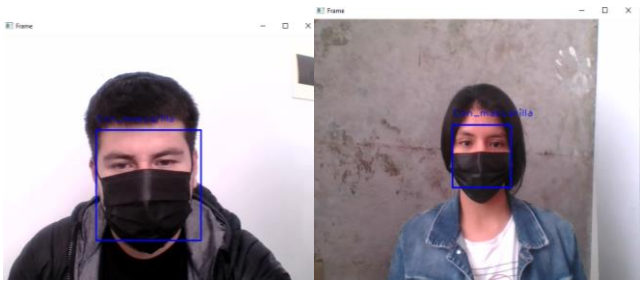

Fig. 17 Resultados de las pruebas en los sujetos con mascarilla.

TABLA II
Resultados de prueba de validación de sujetos con y sin mascarilla.

Sujetos	Con mascarilla (correcto)	Con mascarilla (incorrecto)	Sin mascarilla (correcto)	Sin mascarilla (incorrecto)
1	35	0	30	5
2	33	2	33	2
3	32	3	32	3
4	34	1	31	4
5	32	3	35	0
Total	166	9	161	14
%	94.85%	5.15%	92%	8%

"Con_gorro"

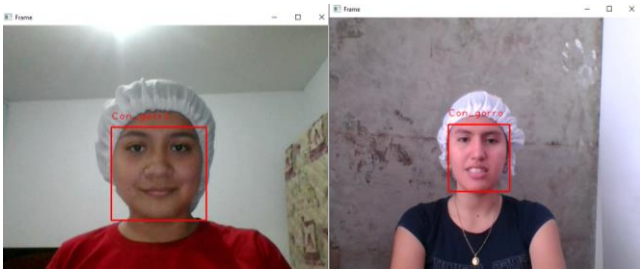

Fig. 18 Resultados de las pruebas en los sujetos con gorro.

"Con_implementos"

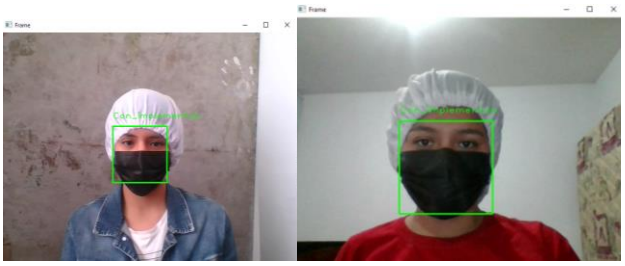

Fig. 19 Resultados de las pruebas en los sujetos.

TABLA III

Resultados de prueba de validación de sujetos con y sin gorro.

Sujetos	Con gorro (correcto)	Con gorro (incorrecto)	Sin gorro (correcto)	Sin gorro (incorrecto)
1	33	2	32	3
2	33	2	33	2
3	31	4	31	4
4	32	3	34	1
5	34	1	32	3
Total	163	12	162	13
%	93.14%	6.86%	92.57%	7.43%

TABLA IV
Resultados de prueba de validación de sujetos con y sin implementos (mascarilla y gorro).

Sujetos	Con implementos (correcto)	Con implementos (incorrecto)	Sin implementos (correcto)	Sin implementos (incorrecto)
1	35	0	31	4
2	34	1	33	2
3	33	2	34	1
4	33	2	32	3
5	32	3	31	4
Total	167	8	161	14
%	95.43%	4.57%	92%	8%

IV. Discusión

En el artículo [17] concluyen que un sistema de visión por computador genera indicadores automatizados de uso adecuado de equipos de protección personal de gran importancia en la industria de la construcción, concretamente el uso de casco de seguridad y chaleco de alta visibilidad. Su sistema se construye sobre una arquitectura de redes neuronales que trabaja sobre imágenes digitales reduciendo de esta forma la necesidad de monitoreo y supervisión. Este resultado confirma que el uso de un sistema Python de detección de EPPs es adecuado para las empresas, ya que se emplea la visión artificial, logrando un mejor control de los sistemas de monitoreo, además de ser más eficiente que un sistema manual y presencial.

Aunado a esto, en nuestra investigación se aprecia el proceso de desarrollo de un programa de visión artificial por medio del software Phyton 3.10.8, logrando que verifique y detecte si la persona utiliza mascarilla quirúrgica y/o gorra.

Así también, en el artículo [15] llegaron a la conclusión que una máquina puede aprender de manera autónoma utilizando redes neuronales para el reconocimiento facial inclusive cuando se utiliza mascarillas y/o lentes, alcanzando un porcentaje de precisión de 71% de acierto en la realización de las pruebas. Ese resultado discrepa de este proyecto cuyo sistema de detección mediante la visión artificial tiene un promedio del 93.33% de acierto en sus diferentes pruebas.

Del mismo modo, discrepa del resultado obtenido por la investigación [16] quienes en su trabajo para monitorear el uso de mascarillas de protección utilizando el framework Flask, en el lenguaje de Python evidenciaron en sus pruebas un 63% de precisión.

Rescatamos que los resultados de esta investigación presentan una similitud en tanto a los proyectos ya que se realizó este sistema de detección de gorras de cocina y mascarillas quirúrgicas a través del uso del Software Python mediante la visión artificial, además de beneficioso para las industrias alimenticias, ya que se tendrá un alto margen de control, evitando que se sigan produciendo estos errores por parte de los trabajadores y se preserve la inocuidad alimenticia.

V. Conclusiones

Se pudo crear un programa de visión artificial para la detección e identificación de mascarillas quirúrgicas y gorras mediante inteligencia artificial.

Sujeto a lo visto en la tabla 2, el porcentaje de validación de la detección del software con mascarilla es del 94.85% y sin mascarilla es del 92%. Dando un total de validación del 93.43%.

Por otro lado, tenemos una mayoría de acierto en el reconocimiento total de mascarilla con 166 veces correctas y 9 incorrectas. Con respecto al reconocimiento sin la mascarilla, de manera correcta fueron 161 veces y 14 veces de manera incorrecta.

Sujeto a la validación del software en la tabla 3, el porcentaje obtenido de la detección con gorros asciende a 93.14% y el porcentaje sin gorros a 92.57%. Dando una validación total del 92.86%.

También se puede observar que se obtienen un reconocimiento total de los sujetos con gorros de 163 veces correctas y 12 veces incorrectas. Además, la cantidad de detección de sujetos sin gorros de manera correcta es de 162 y sin gorros de manera incorrecta es de 13.

Sujeto a lo visto en la tabla 4, el porcentaje de validación de la detección del software con mascarilla y gorro es del 95.43% y sin ambos implementos es del 92%. Dando un total de validación del 93.71%.

Por otro lado, tenemos una mayoría de acierto en el reconocimiento total de mascarilla y gorro de 167 veces correctas y 8 incorrectas. Con respecto al reconocimiento sin ambos implementos, de manera correcta fueron 161 veces y 14 veces de manera incorrecta.

Se concluye que en general, el programa de visión artificial para detectar los elementos de inocuidad tiene una validación total del 93.33%

Referencias

[1] Molano Velandia, J. H., & Arévalo Pinilla, N. (2013). De la salud ocupacional a la gestión de la seguridad y salud en el trabajo: más que

semántica, una transformación del sistema general de riesgos laborales. Innovar, 23(48), 21-32. Recuperado de http://www.scielo.org.co/scielo.php?pid=S0121-50512013000200003&script=sci_abstract&tlng=es

[2] Marin Mendez, M., Rodríguez Julian, A. R., Minier Pouyou, L., Zayas Tamayo, E., & Soler Santana, R. (2020). Caracterización de agentes bacterianos aislados en brotes de enfermedades transmitidas por alimentos. Medisan, 24(2), 235-251. Recuperado de https://www.medigraphic.com/cgi-bin/new/resumen.cgi?IDARTICULO=96107

[3] Municipalidad de San Miguel (8 de junio de 2020). Municipalidad de San Miguel clausuró panaderías que no cumplían con ordenanza municipal para prevenir el covid-19. https://www.munisanmiguel.gob.pe/municipalidad-de-san-miguel-clausuro-panaderias-que-no-cumplian-con-ordenanza-municipal-para-prevenir-el-covid-19/

[4] Trome (11 de marzo de 2021). Tacna: Intervienen dos panaderías que funcionaban en condiciones antihigiénicas. https://trome.com/actualidad/nacional/tacna-intervienen-dos-panaderias-que-funcionaban-en-condiciones-antihigienicas-nnpp-noticia/

[5] Oña, G. E. C., Peralta, V. P., Herrera, J. C., Lozada, D. F., & Carrasco, D. I. (2019). Determinación de Riesgos Existentes en la Industria Panificadora y la Manera de Evitarlos y Controlarlos. European Scientific Journal, 15, 13. http://dx.doi.org/10.19044/esj.2019.v15n13p19

[6] Quispe Chauca, M. J. Propuesta de un proceso de gestión de seguridad y salud ocupacional para una agrupación de panaderías Mype de Lima Metropolitana. Recuperado de https://repositorioacademico.upc.edu.pe/handle/10757/622371

[7] De Paula Denipotti, Maria Emilia, & Do Carmo Cruz Robazzi, Maria Lúcia. (2011). RISCOS OCUPACIONAIS IDENTIFICADOS NOS AMBIENTES DE PANIFICAÇÃO BRASILEIROS. Ciencia y enfermería, 17(1), 117-127. https://dx.doi.org/10.4067/S0717-95532011000100012

[8] Antão, P., Calderón, M., Puig, M., Michail, A., Wooldridge, C., & Darbra, RM (2016). Identificación de indicadores de salud, seguridad y salud en el trabajo (SST) y desempeño ambiental en las áreas portuarias. Ciencias de la seguridad, 85, 266-275. https://doi.org/10.1016/j.ssci.2015.12.031

[9] Fernández-Muñiz, B., Montes-Peón, JM, & Vázquez-Ordás, CJ (2012). Clima de seguridad en organizaciones certificadas OHSAS 18001: Antecedentes y consecuencias del comportamiento de seguridad. Análisis y prevención de accidentes, 45, 745-758. https://doi.org/10.1016/j.aap.2011.10.002

[10] Sanner, M. F. (1999). Python: a programming language for software integration and development. J Mol Graph Model, vol. 17, no 1, p. 57-61.

[11] Figueroa, D. y Roa, E. (2016). Sistema de visión artificial para la identificación del estado de madurez de frutas (granadilla). Revista Redes de Ingeniería. 7(1), 84-92. Doi: 10.14483/udistrital.jour.redes.2016.1.a08

[12] Campos, N. y Verdeguer, D. (2022). Diseño e implementación de un sistema de identificación de personas para la seguridad de los accesos a condominios, basado en el algoritmo de reconocimiento facial LBPH Faces. Recuperado de http://dx.doi.org/10.18687/LACCEI2021.1.1.213

[13] Sandoval, A., López, E., Martínez, C., & Rivas, L. C. (2015). Sistema de Autenticación Facial mediante la Implementación del algoritmo PCA modificado en Sistemas embebidos con arquitectura ARM. La Mecatrónica en México, 4, 53-64. Recuperado de http://www.mecamex.net/revistas/LMEM/

[14] Costa Mari, D. (2020). Análisis de un sistema de reconocimiento facial a partir de una base de datos realizado mediante Python. Recuperado de https://upcommons.upc.edu/handle/2117/331277?show=full

[15] Mucha, J. E., Ortega, A. R., Salazar, L. M. R., Montero, M. M. A., Pahuacho, R. C. U., & Moreno, A. E. G. (2021). Aplicación del deep learning para el reconocimiento facial con la presencia de oclusiones en el contexto de la pandemia covid 2021. Revista ECIPerú Volumen, 18(1). DOI: https://doi.org/10.33017/RevECIPeru2021.0002/

[16] Pereira Júnior, A., Donadon Homem, T. P., & Oliveira Teixeira, F. (2021). Aplicación de inteligencia artificial para monitorear el uso de mascarillas de protección. Revista Científica General José María Córdova, 19(33), 205-222. https://doi.org/10.21830/19006586.725

[17] Massiris, M., Fernández, J. Á., Bajo, J. M., & Delrieux, C. A. (2021). Sistema automatizado para monitorear el uso de equipos de protección personal en la industria de la construcción. DOI:10.4995/riai.2020.13243

[18] Sudhir, B., Ananya, M., Shruti, B., & Sakshi, K. (2020). Smart Attendance System using OPENCV based on Facial Recognition, *INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT*) Volume 09, Issue 03 (March 2020). Doi: 10.17577/IJERTV9IS030122

[19] Chuquimarca, L., Pinzón, S. & Rosales, A. (2021). Detección de Mascarilla para COVID-19 a través de Aprendizaje Profundo usando OpenCV y Cascade Trainer GUI. Revista Científica y Tecnológica UPSE Vol. 8, Nº 1Junio2021, 68-73(julio-diciembre2021) Recuperado de https://incyt.upse.edu.ec/ciencia/revistas/index.php/rctu/article/view/572/496

[20] Ramírez, M. R., Soto, M. D. C. S., Moreno, H. B. R., Rojas, E. M., Millán, N. D. C. O., & Cisneros, R. F. R. (2019). Metodología SCRUM y desarrollo de Repositorio Digital. *Revista Ibérica De Sistemas e Tecnologias De Informação*, (E17), 1062-1072.

[21] Dominguez, P. (2018). El ciclo de producción en SCRUM. Welcomedevelopers_ Recuperado de https://welcomedevelopers.es/agile/el-ciclo-de-produccion-en-scrum/

[22] Ardila, J. C. H., & Peña, C. C. R. (2022). Implementación de un prototipo para la detección de rostros utilizando el lenguaje de programación Python: Prototype for face detection using the Python programming language. Tecnología Investigación y Academia, 10(1), 200-210. https://revistas.udistrital.edu.co/index.php/tia/article/view/18081/18464

[23] Moreno Fernández, S. (2020). Herramienta de Reconocimiento de Imágenes en Python. (Trabajo Fin de Grado Inédito). Universidad de Sevilla, Sevilla. Recuperado de https://idus.us.es/handle/11441/102090

[24] Zamorano, J. (2019). Comparación y análisis de métodos de clasificación con las bibliotecas scikit-learn y TensorFlow en Python. Recuperado de https://riuma.uma.es/xmlui/handle/10630/19037

[25] Constante P. & Gordón A. (2015). Diseño e implementación de un sistema de visión artificial para clasificación de al menos tres tipos de frutas. Recuperado de https://bibdigital.epn.edu.ec/handle/15000/11368

[26] Solano, G. (2019). RGB en OpenCV – Python [Mensaje de un blog] *Omes*. Recuperado de https://omes-va.com/rgb/

[27] Artola Moreno, Á. (2019). Clasificación de imágenes usando redes neuronales convolucionales en Python. (Trabajo Fin de Grado Inédito). Universidad de Sevilla, Sevilla. Recuperado de https://idus.us.es/handle/11441/89506

[28] Angulo F., Montes N., Osorio G. & Prieto F. (2001). La visión artificial aplicada al proceso de producción de café. Recuperado de https://www.academia.edu/1344027/La_visi%C3%B3n_artificial_aplicada_al_proceso_de_producci%C3%B3n_del_caf%C3%A9

[29] Solano, G. (2021). ¿Detección de rostros con MEDIAPIPE? Python – MediaPipe – OpenCV [Mensaje de un blog] *Omes*. Recuperado de https://omes-va.com/deteccion-de-rostros-mediapipe-python/