ISBN: 978-628-95207-7-4. ISSN: 2414-6390. Digital Object Identifier: https://dx.doi.org/10.18687/LEIRD2023.1.1.540

Development of a facial recognition algorithm and epps detection with python for the construction industry

Ryan Abraham León León[1], Katia Fernanda Alcántara Rufino[1];
[1]Universidad Privada del Norte, Facultad de Ingeniería Industrial, Perú
Email: ryan.leon@upn.edu.pe
[1]Universidad Privada del Norte, Facultad de Ingeniería Industrial, Perú
Email: N00179978@upn.pe

Abstract

This project is based on providing an alternative solution to work accidents that occur in the construction sector, through the design of a facial recognition algorithm and EPPS detection with Python for the construction industry, for the present project it was considered control and prevention of unsafe acts during working hours, with the aim of contributing to the preservation of the health and safety of workers in areas that represent high risks of occupational accidents, where the use of personal protective equipment is of vital importance; Therefore, the objective is to design a facial recognition and EPPS detection algorithm with Python, develop a prototype that can identify and in turn indicate when personnel are using EPPS (helmet and glasses) properly, for which the use of EPPS will be necessary. of the PyCharm software that will help to process images and likewise carry out different tests of the system until validating the optimal operation. tests were performed using sequences of 300, 600 and 900 frames. Reaching with this last number of frames the optimal efficiency with 95.4%, these techniques used to detect the correct use of the epps are presented through logarithms given in real time which will be captured through a camera, according to the code of programming to be developed with the data obtained and the reference points in relation to the appearance of the personnel and the use of their helmets and glasses according to the vision of the programmed system.

Keywords: Python, algorithm, PyCharm, Facial recognition, EPPs, Security.

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

Desarrollo de un algoritmo de reconocimiento facial y detección de epps con python para la industria de construcción

Ryan Abraham León León[1], Katia Fernanda Alcántara Rufino[1];
[1]Universidad Privada del Norte, Facultad de Ingeniería Industrial, Perú
Email: ryan.leon@upn.edu.pe
[1]Universidad Privada del Norte, Facultad de Ingeniería Industrial, Perú
Email: N00179978@upn.edu.pe

Resumen

El proyecto está basado en dar una alternativa de solución a accidentes laborales que ocurren en el sector construcción, mediante el diseño de un algoritmo de reconocimiento facial y detección de EPPS con Python para la industria de construcción, para el presente proyecto se consideró control y prevención de actos inseguros en las jornadas laborales, con el afán de contribuir en la conservación de la salud y seguridad de trabajadores en áreas que representen riesgos altos de accidentes laborales, donde el uso de equipos de protección personal es de vital importancia; por lo que el objetivo es diseñar un algoritmo de reconocimiento facial y detección de EPPS con Python, desarrollar un prototipo que pueda identificar y a su vez indicar cuando el personal está usando adecuadamente los EPPS (cascos y lentes), para lo cual será necesario el uso del software PyCharm que ayudarán para el procesamiento de imágenes y así mismo ir realizando distintas pruebas del sistema hasta validar el óptimo funcionamiento. se realizaron pruebas utilizando secuencias de 300, 600 y 900 fotogramas. Alcanzando con esta última cantidad de fotogramas la eficiencia optima con un 95.4 %, Se presentan estas técnicas empleadas para detectar el correcto uso de los epps a través de logaritmos dados en tiempo real las cuales serán captadas a través de una cámara, según el código de programación a desarrollar con los datos obtenidos y los puntos de referencia con relación al aspecto del personal y el uso de sus cascos y lentes según la visión del sistema programado.

Palabras claves: Python, algoritmo, PyCharm, Reconocimiento facial, EPPs, Seguridad.

I. INTRODUCCIÓN

El presente proyecto incorpora una alternativa de control y prevención de actos inseguros en las jornadas laborales, con el afán de contribuir en la conservación de la salud y seguridad de trabajadores en áreas que representen riesgos altos de accidentes laborales, donde el uso de equipos de protección personal es de vital importancia. Por otro lado, el ser humano siempre ha tratado de facilitar la manera en la que vive, la cual busca emplear la mínima cantidad de energía en la realización de un trabajo [1] es por ello que los avances tecnológicos de la inteligencia artificial, especialmente en visión por computadora, han incrementado su importancia en muchas áreas y se ha considerado un campo de investigación interesante en los últimos años [2] buscando imitar el proceso natural de diferenciación de personas con la ayuda de algoritmos complejos [3]. El reconocimiento facial es una técnica biométrica, una versión mucho más potente que la tecnología que usa el celular o la computadora para identificar personas y validar sus identidades [4]. Para ampliar mayor esta técnica, estos métodos realizan la identificación a través de características fisiológicas o sociales del ser humano como el rostro, venas, huellas dactilares, retina, etc., o en rasgos de comportamiento como la pulsación de teclas. Se podría decir entonces que la biométrica está diseñada para que tú no tengas que hacer nada. Simplemente el dispositivo te reconoce.

Por otro lado, en el proceso de reconocimiento se utilizan algoritmos para el procesamiento de imágenes, en especial se ha dado por presentar ciertas ventajas sobre otras biométricas: una de ellas es que puede ser completamente no intrusivo, en otras palabras, dicha tecnológica se puede implementar sin la presencia de un operario, ya que las imágenes son obtenidas desde una cámara a cierta distancia. Asimismo, las imágenes pueden ser adquiridas por medios de dispositivos de no tan alto costo, a diferencia de otros sistemas de reconocimiento, los cuales poseen tanto una manufactura como un mantenimiento constante de gran valor [5]

En ese contexto, la detección a través de imágenes y videos puede ayudar en el monitoreo de la seguridad, el control de calidad y la gestión de la productividad en los sitios de construcción [6]. En este sentido la detección de EPPS utilizando técnicas de Visión Computacional resulta ser una medida efectiva y comprobada, visualizada en tiempo real de trabajo y óptima para un monitoreo constante no supervisado [7]. En la última década, la detección de equipos de protección ha sido utilizado como filtro inicial en procesos de seguimiento visual de trabajadores en sitios de construcción. Por lo que se propone un reconocimiento biométrico facial mediante el sistema Python el cual nos permite la identificación de personas, en el acceso a sitios privados y todos los lugares que necesiten seguridad [8]. En lo que respecta a seguridad laboral, el uso de Equipo de Protección Personal (EPP) para proteger la salud y seguridad del personal de la obra es obligatorio en cada proyecto de construcción. Sin embargo, existen problemas para monitorear la existencia y el uso apropiados de EPP cuando los trabajadores se presentan

en el trabajo [9]. Por lo anterior, existe la necesidad de un sistema que aplique principios modernos de recopilación y control de datos que cumplan al menos con las normas de seguridad y salud en el trabajo, como el reconocimiento biométrico fácil el cual nos permitirá monitorear el uso correcto de los EPPS en los trabajadores.

Por otro lado, esta investigación presenta una explicación de las bases teóricas como el reconocimiento biométrico facial el cual hoy en día es un avance tecnológico muy importante en comparación a décadas pasadas, pues este se ha convertido en una gran mejora en el campo de la seguridad [10] razón por la cual podría afirmarse que los sistemas de seguridad basados en biometría son un medio eficaz y eficiente [11].

Así mismo cabe mencionar que Python es un lenguaje de programación de alto nivel, interpretado, de una sintaxis sencilla, propiedades que lo hacen muy adecuado para el proceso de enseñanza-aprendizaje de lenguaje de programación [12]. Python es considerado un lenguaje de programación de alto nivel, siendo atractivo en el campo del Desarrollo Rápido de Aplicaciones. Además, Python soporta características de computación moderna, como grandes datos, aprendizaje de máquina y desarrollo de aplicaciones web con poca programación. [13]

El proyecto "Python como primer lenguaje de programación" [14] tiene como principal objetivo realizar un software que permita detectar el rostro de los colaboradores que lleven puesto los Epps para uso en el sector de construcción en las distintas instancias del proceso productivo que así lo requieran. Para ello, se incurrió en la elaboración del módulo para ejecución de software, aplicación de las librerías, determinar la cantidad de imágenes óptimas para la ejecución del sistema. Concluyendo que el programa Python es una de las mejores opciones disponibles para enseñar a programar a estudiantes de las áreas de ingeniería, computación y medios digitales. Para un primer curso de programación, este lenguaje ofrece ventajas importantes sobre otros lenguajes.

Así mismo, la tesis titulada "Diseño de una red neuronal convolucional para el reconocimiento facial" [15] busca diseñar una red neuronal convolucional para el reconocimiento facial de igual forma generar una base de datos mediante el uso de Python para la etapa de aprendizaje de la red neuronal mediante los distintos tipos de sistemas de control, como la Inteligencia Artificial, Machine Learning, Deep Learning, Redes Neuronales Biológicas, etc. Donde se procesará la programación del sistema Python.

La tesis titulada "Creación de una aplicación de reconocimiento emocional en Python para Neuromarketing y La investigación titulada "Análisis de un sistema de reconocimiento facial a partir de una base de datos realizado mediante Python" [16], destacó que el proceso de identificación de una cara se divide en dos subprocesos sucesivos. Respecto al primer subproceso que es la detección facial, el método usado se enfoca en el algoritmo de Viola-Jones basado en Haar Cascades. Con el segundo subproceso, el reconocimiento facial, los métodos utilizados en este proyecto son Fisherfaces, Eigenfaces y LBPH (Local Binary Pattern Histogram).

A su vez, la tesis de título "Clasificación de imágenes usando redes neuronales convolucionales en Python" [17] expone el análisis de redes neuronales convolucionales aptas para aprender sin supervisión a partir de datos sin estructurar y el valor que tienen en el análisis de imágenes logran entrenar la red para alcanzar un alto porcentaje de precisión, sensibilidad y especificidad, se procesará una investigación de los lenguajes de programación más usados en el campo de la Inteligencia Artificial, bases de datos para explotar la información y el funcionamiento propio de las Redes Neuronales Convolucionales.

El presente proyecto está justificado por la latente necesidad de contar con herramientas y tecnologías que puedan ayudar a supervisar y rastrear el uso del EPP, identificar deficiencias en su uso y fomentar una cultura de seguridad en el lugar de trabajo. Se pretende que la implementación de un software de detección de equipo de protección personal pueda ayudar a abordar estos desafíos y mejorar la seguridad en el sector de la construcción.

Por lo mencionado, el objetivo principal del presente proyecto es diseñar un algoritmo de reconocimiento facial y detección de EPPS con Python para la industria de construcción. Para lo cual se establecieron los siguientes objetivos específicos: utilizar las librerías open CV, Tensor Flow, numpy, os para la vision artificial de Python, determinar la cantidad óptima de fotogramas necesarios para la base de datos del sistema y hallar el porcentaje de eficiencia del sistema de detección de equipo de protección personal.

II. Materiales y métodos

Nuestro proyecto trabaja con una computadora que tiene un procesador Intel (R) Core (TM) i7-6560U CPU 2.20GHz, 2208 Mhz de un RAM de 16 GB. Además, se hicieron uso de diferentes librerías que se detallan en la Tabla 1.

TABLA I
REGISTRO DE SOFTWARE Y GALERÍAS

Nombre	Tipo/Marca	Versión/ Modelo
Tensor Flow	Librería	Python
Numpy	Librería	Python
Open cv	Librería	Python
Os	Librería	Python
Python File	Librería	3.9
Pycharm	Software	2022

En dos de los módulos; dentro de la fase "detección"; se implementó en el modelo LBPH (local Binary Patterns), este modelo interviene de manera directa en el porcentaje de efectividad y eficacia al momento de que el operador muestre su rostro y este sea detectado, para luego realizar comparaciones por píxeles [18]. Ello guiado de una base de datos o cierto conjunto de imágenes identificables sustrae

información para realizar un redireccionamiento de estas y poder identificar de que id hace referencia.

El modelo LBPH etiqueta a cada píxel, tomando como punto de guía la distribución de los píxeles vecinos. Este procedimiento continúa con una máscara que de manera iterativa selecciona un píxel central todo el tiempo, además, detecta los parámetros de sus píxeles vecinos.[19] Este píxel central, de manera más detallada, tiene como función compararse con respecto a los demás pixeles, asignando un 1 si es mayor al resto de comparados o 0 si es menor

Matemáticamente este modelo se da a conocer por medio de la ecuación a continuación 1

$$LPB\ (Xc, Yc) = \sum_{P=0}^{P-1} S(i_p - i_c)\, 2^p \quad (1)$$

```
CODIGO:
reconocimiento = cv2.face. LBPHFaceRecognizer_create()
```

Fig.1 Codificación LBPHF

Visualizando varios valores, el (Xc , Yc) se le conoce como el punto central o píxel central cuya intensidad es mayor. El símbolo S representa la siguiente función matemática, la cual tiene como objetivo obtener imágenes con mayor precisión y nitidez. El modelo matemático se expresa mediante la ecuación 2. La cual trabaja en conjunto con la ecuación 1 para parametrizar los criterios de asignación del código binario a cada píxel y facilitar así el reconocimiento facial. De esta forma, en la figura 1 se declara el lector del modelo y en la figura 2 se hace un llamado a este para que pueda ser leído

$$S(x) = \begin{cases} 1 \text{ si} \geq 0 \\ 0 \text{ otro valor} \end{cases} \quad (2)$$

```
CODIGO:
reconocimiento.read('ModeloEPPs.xml')
```

Fig.2 Declaración de modelo de reconocimiento

Realizando dicha función, obtener un número binario, el cual se transforma en decimal para formar parte de la descripción de manera previa, pasa a un histograma de las etiquetas donde se concentran todos los pixeles de acuerdo con la ecuación 3, para luego ser implementados como descriptores de la imagen.

$$Hi = \sum xy\ I((x, y) = i), i = 0, \ldots, n-1$$

$$I(x) \begin{cases} 1, si\ x\ es\ verdadero \\ 0, otro\ valor \end{cases} \quad (3)$$

```
CODIGO:
for i in range(len(caras)):
    x1,y1,ancho, alto = caras[i]['box']
    x2,y2 = x1 + ancho, y1 + alto
    cara_reg = copia2[y1:y2, x1:x2]
    cara_rec = cv2.resize(cara_reg,(150,200), interpolation
= cv2.INTER_CUBIC)
    resultado = reconocimiento.predict(cara_rec)
```

Fig.3 Codificación de parámetros de reconocimiento

Este histograma visualizado es llamado Local Binary Pattern Histogram más conocido como LBPH. La figura 3 muestra la codificación de los parámetros de detección o el medio por el que se ubicara cada pixel a ser evaluado al momento de la detección.

Con respecto al tipo de software, se implementó el Pycharm 2022, ya que identifica y lee varios tipos de códigos, lo cual lo hace una esencial herramienta intuitiva y a su vez sencilla de utilizar, además, es de libre acceso para los estudiantes [20]. Este software posee un lenguaje de programación simple con un amplio aprendizaje para que lo use, dándose uso más frecuente para la lectura.

El lenguaje Python se implementó en una de sus versiones más actuales (Python 3.9) siendo esta la más recomendable para el uso de las librerías seleccionadas.

Los módulos principales fueron opencv que posee una subcarpeta opencv-contrib-python, librerías que nos permitirán ser el punto de contacto entre su interfaz de cámara y el programa de detección [21].

Otra librería para mencionar sería la Tensor Flow, la cual se encarga del reconocimiento facial. Por último, tenemos a Numpy con función de instaurar el enlace de conexión entre las bases de datos por medio de la formación de vectores de reconocimiento.

Metodología de programación de módulos:

El software PyCharm simplifica el proceso de instalación del código al permitirle hacerlo directamente dentro del propio programa. Esto se puede hacer accediendo al programa, que aparece como se muestra en la siguiente imagen. Desde el panel de inicio del proyecto, tiene la capacidad de seleccionar el tipo de código con el que le gustaría trabajar, en este caso, Python. Además, el programa utiliza automáticamente de forma predeterminada el intérprete de código Python 3.9 y ofrece la opción de elegir el directorio en el que almacenar su archivo. En última instancia, la interfaz fácil de usar de PyCharm agiliza en gran medida el proceso de instalación del código, haciéndolo accesible para aquellos con conocimientos técnicos limitados.

La librería Tensor Flow nos proporciona una herramienta algorítmica llamada LBPH que nos ha permitido ajustar las características del rostro mediante la captura de múltiples fotografías, las cuales almacenarán en distintas carpetas para conformar una base de datos controlada de identificación [22]. De esta manera, se pretende identificar a una persona a partir de sus rasgos faciales, como la distancia y posición entre los ojos, la nariz, las orejas y la boca.

Se buscó enriquecer la base de datos para hacer el sistema más eficiente y adaptable. Por esta razón, se evaluó el funcionamiento del algoritmo con 300,600 y 900 fotogramas por cada persona, en diversas posiciones del rostro y con diferentes expresiones faciales, tanto como con EPPs como sin ellos.

Una vez que se ha recolectado toda la información necesaria, se procede a entrenar el modelo de inteligencia artificial con el fin de que pueda recibir una retroalimentación de la nueva información ingresada y estar en óptimas condiciones para llevar a cabo el reconocimiento facial. Para llevar a cabo este proceso de entrenamiento, se utilizarán las librerías Os, que permite el acceso a las carpetas de la base de datos y al detector facial; Open CV, que requiere que la escala de colores esté en grises para poder integrarse correctamente; y Numpy, que se encarga de clasificar los fotogramas de acuerdo con la identificación correspondiente [23]

En la programación del tercer módulo, haremos uso de la biblioteca Open CV para la configuración de video y cámara. Además, estableceremos un vínculo entre el archivo de comentarios y el modelo y se importará a la biblioteca Tensor Flow para el reconocimiento facial. A su vez esto nos funcionará como integrador de todos los módulos tanto el de obtención de fotogramas como del entrenador, facilitando la detección de los EPPS a tiempo real.

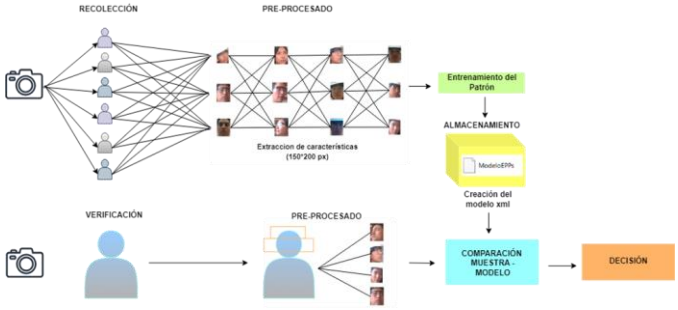

Fig.4 Arquitectura del algoritmo

La Figura 4 muestra la arquitectura del funcionamiento del algoritmo y la presencia de los 3 módulos de alimentación del algoritmo, dando paso en primer lugar a la alimentación de este, en la recolección de los fotogramas necesarios, para posteriormente ser procesados y almacenados en diferentes carpetas. Luego de recolectar esta cantidad de fotogramas el modelo pasa a ser retroalimentado, agrupando y etiquetando estos los píxeles de acuerdo con patrones que se convertirán en un archivo .xml que le brindará a nuestro algoritmo toda la información que necesita. Una vez tenga toda la información ya está listo para poder verificar el reconocimiento mediante el tercer módulo en donde se llama al lector del archivo .xml y al de reconocimiento facial y a tiempo real hacen una predicción del uso correcto o incorrecto de los EPPs.

Al activar la cámara, el algoritmo capta la imagen de la persona a través de la toma de imagen a tiempo real dando por resultado final la detección de la presencia o ausencia de los EPPS. De esta forma se refleja en pantalla con qué EPPs está contando el operario y si es que no cuenta con ninguno. Si el operario no cuenta con EPPs se rodeará su rostro con color rojo, por el contrario, si se encuentra portando los EPPs el borde tendrá un color designado tal como se muestra en la figura 5

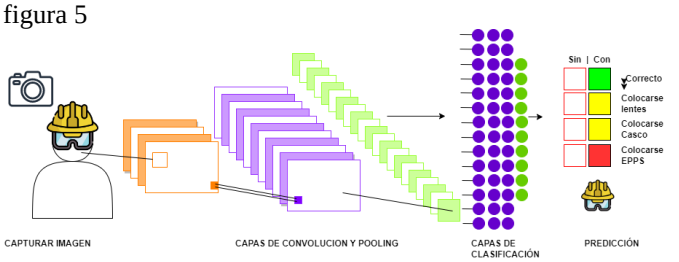

Fig.5 Diagrama de la red neuronal.

La figura 5 representa una red neuronal que muestra todo el procesamiento de fotogramas, desde la recopilación de datos hasta la predicción de características individuales basadas en comentarios
anteriores. Como primer paso, hay un momento para ingresar información en el módulo, 900 capturas de fotogramas de la persona bajo análisis se muestran y almacenan consecutivamente en carpetas que identifican y etiquetan esta información y bancos de procesamiento de imágenes posteriores donde se almacenan en capas de convolución Ir a. Esto es gracias a la herramienta LBPH, con la ayuda de una serie de puntos centrados en píxeles que separan los píxeles ubicándolos por sus coordenadas. Estos píxeles ya etiquetados son redirigidos a diferentes capas modificadoras que dan criterios de predicción según la imagen presentada y características similares encontradas en la base de datos que informa. Se hicieron uso de dos capas de convolución y una intermedia de Pooling. Las dos primeras capas son de convolución, la cual tiene como función analizar la gran cantidad de imágenes proporcionadas en el recolector de imágenes y detectar la presencia de un conjunto de features o de características comunes. Funcionan como filtros, los cuales permiten extraer de la imagen píxeles específicos como formas geométricas específicas. Se hizo uso de una primera capa de convolución la cual amplía la muestra con menos cantidad de pixeles para luego mediante la capa de pooling , se usó con el objeto de reducir aún más el tamaño de los pixeles recogidos, este método es usado para reducir el número de parámetros que tiene que aprender, creando una nueva matriz de salida donde cada uno de sus componentes corresponde a los máximos de cada región detectada o la que aparezca con más frecuencia, después estos se encuentran en una segunda capa de convolución. Gracias a este proceso de segregación podemos entrar en la clasificación, correspondiente a un modelo de percepción multicapas, con el objeto de que a cada muestra se le atribuyen etiquetas según la clase a las que pertenezcan, pudiendo así arrojar una detección a tiempo real.

III. RESULTADOS

Se llevaron a cabo una serie de 80 pruebas utilizando a 7 personas diferentes, para lograr determinar la frecuencia de errores y aciertos del sistema, tanto en presencia como en ausencia de EPPs. Si para la validación realizáramos un

análisis estadístico de ítems, bastaría con tener un tamaño muestral que fluctúa entre 50 y 100 sujetos para una primera aplicación, pensando que, en una segunda, con el instrumento mejorado, deberíamos tener entre 5 y 10 sujetos por ítem, con un mínimo de 300, ya que de esta forma se pueden tener mayores garantías respecto a la validez del instrumento [24]. De esta forma al tener un total de 4 items, se tomarán 320 pruebas por individuo en cada una de las cantidades en las que se alimentó el algoritmo con el objetivo de encontrar el número óptimo de fotogramas para la base de datos. Para lograrlo, se realizaron pruebas utilizando secuencias de 300, 600 y 900 fotogramas

TABLA II

TABLA DE RECOLECCIÓN DE DATOS

#FOTOGRAMAS	PESO ARCHIVO MB	#PERSONAS	CON EPPS		SIN EPPS		SOLO CON LENTES		SOLO CON CASCO	
			CORRECTO	FALLOS	CORRECTO	FALLOS	CORRECTO	FALLOS	CORRECTO	FALLOS
300	10.34	1	60	20	58	22	45	35	45	35
		2	58	22	62	18	48	32	52	28
		3	62	18	54	26	56	24	36	44
		4	48	32	41	39	32	48	44	36
		5	52	28	48	32	41	39	72	8
		6	54	26	42	38	58	22	55	25
		7	57	23	32	48	32	48	48	32
TOTAL		28	391	169	337	223	312	248	352	208
600	20.62	1	61	19	62	18	52	28	62	18
		2	65	15	48	32	62	18	69	11
		3	71	9	59	21	72	8	67	13
		4	45	35	54	26	71	9	66	14
		5	52	28	48	32	65	15	65	15
		6	70	10	65	15	62	18	66	14
		7	75	5	69	11	70	10	60	20
TOTAL		28	439	121	405	155	454	106	455	105
900	31.02	1	75	5	80	0	76	4	80	0
		2	74	6	76	4	79	1	79	1
		3	73	7	75	5	80	0	75	5
		4	75	5	74	6	74	6	75	5
		5	78	2	71	9	77	3	77	3
		6	74	6	76	4	80	0	77	3
		7	80	0	77	3	75	5	74	6
TOTAL		28	529	31	529	31	541	19	537	23

En la tabla II se presentan los errores detectados al emplear EPPs y al no utilizarlos, junto con la información sobre el tamaño en megabytes de la base de datos.

TABLA III

CÁLCULO DE EFICIENCIA

INDICADORES				
300 FOTOGRAMAS	CON EPPS	SIN EPPS	SOLO CON LENTES	SOLO CON CASCO
TOTAL DE PRUEBAS	560	560	560	560
EFICIENCIA	69.8%	60.2%	56%	63%
ERROR	30.2%	39.8%	44%	37%

EFICIENCIA TOTAL	62.1%
ERROR TOTAL	37.9%

600 FOTOGRAMAS	CON EPPS	SIN EPPS	SOLO CON LENTES	SOLO CON CASCO
TOTAL DE PRUEBAS	560	560	560	560
EFICIENCIA	78.4%	72.3%	81%	81%
ERROR	21.6%	27.7%	19%	19%

EFICIENCIA TOTAL	78.3%
ERROR TOTAL	21.7%

900 FOTOGRAMAS	CON EPPS	SIN EPPS	SOLO CON LENTES	SOLO CON CASCO
TOTAL DE PRUEBAS	560	560	560	560
EFICIENCIA	94.5%	94.5%	97%	96%
ERROR	5.5%	5.5%	3%	4%

EFICIENCIA TOTAL	95.4%
ERROR TOTAL	4.6%

Asimismo, como se muestra en la tabla 3 se realizaron cálculos para determinar el porcentaje de error y la eficacia en función de la cantidad de fotogramas por persona en dicha base de datos, se pudieron evaluar en 4 modalidades de arrojo de resultados, estos fueron: al usar todos los EPPS en este caso lentes y casco, al no usar ninguno de los EPPs y al usar solo lentes y al usar solo caso.

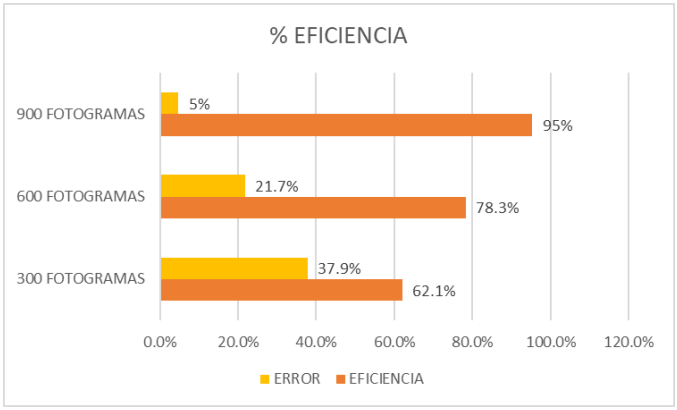

Fig.6 Gráfico de Indicadores

Según la información presentada en la figura 6, se observa que a medida que aumenta la cantidad de fotogramas, el sistema muestra un mayor porcentaje de eficacia. Sin embargo, es importante tener en cuenta que, al incrementar la cantidad de fotogramas, también se incrementa el uso de memoria, lo que podría ocasionar demoras. A pesar de este factor, se ha decidido utilizar 900 fotogramas por usuario (persona) debido a la naturaleza del proyecto y al nivel de eficiencia requerido en este contexto específico. De esta manera, se logra alcanzar un porcentaje de eficiencia del 95.4%.

IV. DISCUSIÓN

"Desarrollo de un sistema de detección de rostro con o sin mascarilla mediante el uso de software" presenta un software con visión artificial usando el lenguaje Python en Visual Studio Code el cual es capaz de detectar si una persona lleva mascarilla o no, este proyecto se hizo con el fin de poder identificar a las personas que estén sin mascarillas o mal puestas ya sea en el trabajo o entrada de un local. Se realizó una serie de 40 pruebas por cada integrante del equipo, 20 con mascarilla y 20 mal o sin mascarilla, se obtuvieron como resultados una cantidad de veces que el sistema ha dado un resultado correcto e incorrecto. tiene un porcentaje de validación por encima del 80% con la mascarilla puesta y de un 86% con la mascarilla mal puesta o sin usar, además de que su porcentaje total de validación para ambos casos es del 83.13% [19]. En consecuencia, para la detección de EPPs, en nuestro se obtuvo un porcentaje de validación fue del 95.4% esto debido a la retroalimentación sobre una base de datos más amplia y de mayor reconocimiento.

"Reconocimiento facial basado en eigenfaces, lbhp y fisherfaces en la beagleboard-xm" realizaron una implementación de un sistema de detección de rostro aplicando procesamiento de imágenes, aplicando características basados en eigenfaces, Histogramas de patrones locales binarios LBPH y discriminantes Fisher faciales, sobre el dispositivo embebido BeagleBoard-xM se pudo determinar que el algoritmo de clasificación empleado para reconocimiento facial que presenta los mejores resultados

entre los tres propuestos es el método de Eigenfaces [18], ya que es el que presenta menor porcentaje de error de clasificación y menor tiempo de ejecución con un 93% de eficiencia. Para el presente proyecto de detección de EPPs se usó LBPH herramienta la cual también hacen uso con un 95.4% de eficiencia, esto debido a la gran cantidad de información y datos que tiene que procesar, lo cual también se refleja en los tiempos de respuesta.

V.CONCLUSIONES

Como una medida de control para prevenir actos inseguros durante horas laborales, se diseñó un software de reconocimiento facial y asi detectar las EPPS: lentes y casco en los trabajadores de la industria de construcción con el programa Python. Asimismo, ayuda al monitoreo de seguridad, aumentar el control de calidad y la gestión de la productividad en dichas áreas de producción.

Python es considerado un lenguaje de programación de alto nivel, siendo atractivo en el campo del Desarrollo Rápido de Aplicaciones. Además, soporta características de computación moderna, como grandes datos, aprendizaje de máquina y desarrollo de aplicaciones web con poca programación.

Se implementaron 3 módulos de manera complementaria para la efectividad del logaritmo. En primera instancia tenemos un recolector de base de datos, donde se tomaron 900 fotogramas/persona., el siguiente modulo es para el entrenamiento del modelo artificial y así obtener una buena retroalimentación. Finalmente, el tercero es el módulo de usuario para el reconocimiento facial.

Se utilizó una muestra de 6 personas aleatorias para una serie de 80 pruebas, la cual dio como resultado que a mayor cantidad de fotogramas mayor eficiencia posee el sistema. Por lo tanto, disminuye el porcentaje de error en los EPPS. No obstante, dicho beneficio almacena mucho espacio de memoria en el operador.

Por último, se logró hallar el porcentaje de eficiencia del sistema de detección de equipos de protección personal en un 95.4% lo que hace más intuitivo.

REFERENCIAS

[1] M. Garduño. “Reconocimiento por biometría facial para aplicaciones en ciudades inteligentes”,2018. Recuperado de http://ri.uaemex.mx/bitstream/handle/20.500.11799/68484/Reconocimiento+por+Biometria+Facial+para+Aplicaciones+en+Ciudades+Inteligentes.pdf;jsessionid=E792A76F71253E2CA7B5B00A1BF18ED8?sequence=1

[2] H. Du, H. Shi, D. Zeng y T. Mei. “The elements of end-to-end deep face recognition: A survey of recent advances”,2020. Recovered from https://arxiv.org/abs/2009 .13290.

[3] R. Barreto y D. Lizarraga. “Modelo de Sistema de Reconocimiento Facial para el Control de la Trata de Personas”, 2019. Recuperado de https://repositorio.utp.edu.pe/bitstream/handle/20.500.12867/2063/Robert%20Barreto_David%20LizarragaTesis_Titulo%20Profesional_2019.pdf?sequence=1&isAllowed=y

[4] Herrera y Peña. “Reconocimiento facial con base en imágenes”, 2022. Recuperado de: https://revista.redipe.org/index.php/1/article/view/267/264

[5] H. Arguello. “Recognition Systems Based on the Facial Imagen. Revista Avances en Sistemas e Informática”, 7-13, 2011 Recuperado de: https://www.researchgate.net/publication/267296150_Sistemas_de_reconocimiento_basados_en_la_imagen_facial_Recognition_systems_based_on_the_facial_image

[6] A.Xuehui, Z. Li, L. Zuguang,W. Chengzhi, L. Pengfei, y L. Zhiwei “Dataset and benchmark for detecting moving objects in construction sites. Automation in Construction”,2021. Recuperado de https://doi.org/10.1016/j.autcon.2020.103482

[7] M. Massiris, J.A. Fernández, J. Bajo y C. Delrieux. “Sistema automatizado para monitorear el uso de equipos de protección personal en la industria de la construcción” , 2022. Revista Iberoamericana de Automática E Informática Industrial. Extraido de https://doi.org/10.4995/riai.2020.13243

[8] Cadena Moreano JA, Montaluisa Pulloquinga RH, Flores Lagla GA, Chancúsig Chisag JC, Guaypatín Pico OA. Reconocimiento facial con base en imágenes. 29 de mayo de 2017. Disponible en: https://revista.redipe.org/index.php/1/article/view/267

[9] Kelm, A., Laußat, L., Meins-Becker, A., Platz, D., Khazaee, M. J., Costin, A. M., Helmus, M., & Teizer, J. (2013). Mobile passive Radio Frequency Identification (RFID) portal for automated and rapid control of Personal Protective Equipment (PPE) on construction sites. Automation in Construction, 36, 38–52. https://doi.org/10.1016/j.autcon.2013.08.009

[10] D.Osorio. “Sistema biométrico de reconocimiento facial basado en inteligencia artificial “. Santiago garcia Mongua. Universidad autónoma de occidente facultad de ingeniería departamento de automática y electrónica programa ingeniería mecatrónica, 2022. Extraído de https://red.uao.edu.co/bitstream/handle/10614/13887/T10210_Sistema%20biom%c3%a9trico%20de%20reconocimiento%20facial%20basado%20en%20inteligencia%20artificial.pdf?sequence=4&isAllowed=y

[11] J. Osorio, F. Aguirre y J. Escobar. “Sistemas De Seguridad Basados En Biometría. Scientia et Technica, XVII(46), 98–102, 2010. Recuperado de https://www.redalyc.org/articulo.oa?id=84920977016

[12] U. Xitij Shukla y J. D. J. Parmar “Python – A comprehensive yet free programming language for statisticians, Journal of Statistics and Management Systems”, 2016. Extraido de http://www.tandfonline.com/doi/pdf/10.1080/09720510.2015.1103446

[13] Chang, “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”,2018. Extraido de https://revistas.udistrital.edu.co/index.php/tia/article/view/18081/18464

[14] Ortiz. “Python como primer lenguaje de programación”, 2010. Resumen recuperado de: https://arielortiz.info/publicaciones/primer_lenguaje_30_jun_2010.pdf

[15] L. Rodríguez, R. Becerra, D. Surita. “DISEÑO DE UNA RED NEURONAL CONVOLUCIONAL PARA EL RECONOCIMIENTO FACIAL”, 2022. Recuperado de https://repositorio.unp.edu.pe/bitstream/handle/20.500.12676/4177/IMEC-ROD-BEC-SUR-2022.pdf?sequence=1&isAllowed=y

[16] Costa. “Análisis de un sistema de reconocimiento facial a partir de una base de datos realizado mediante Python”, 2020. Resumen recuperado de: https://repositorio.ucv.edu.pe/

[17] Á Artola. “Clasificación de imágenes usando redes neuronales convolucionales en Python”. Universidad de Sevilla, Sevilla, 2019. Recuperado de: https://hdl.handle.net/11441/89506

[18] C. Franco, C. Ospina, E. Cuevas, y D. Capacho, «RECONOCIMIENTO FACIAL BASADO EN EIGENFACES, LBHP Y FISHERFACES EN LA BEAGLEBOARD-xM», Rev. Colomb. Tecnol. Av. RCTA, vol. 2, may 2017, doi: 10.24054/16927257.v26.n26.2015.2387. 21_UTS_Artículo Formato RCTA.doc (unipamplona.edu.co)

[19] Parth Singh, Understanding Face Recognition using LBPH algorithm», Analytics Vidhya, 12 de julio de 2021. https://www.analyticsvidhya.com/blog/2021/07/understanding-face-recognition-using-lbph-algorithm/

[20] PyCharm: el IDE de Python para desarrolladores profesionales, por JetBrains. https://www.jetbrains.com/es-es/pycharm/

[21] OpenCV - Open Computer Vision Library. https://opencv.org/ (accedido 29 de mayo de 2023).

[22] Face detection guide | Tensor Flow | Google for Developers». https://developers.google.com/Tensor Flow/solutions/vision/face_detector .
[23] NumPy, «ESTUDIO DE CASO: ESTIMACIÓN DE POSE 3D DE DEEPLABCUT», 2019. https://numpy.org/case-studies/deeplabcut-dnn/
[24] ROCO, A., HERNANDEZ, M y SILVA GONZALEZ, O.¿Cual es el tamaño muestral adecuado para validar un cuestionario?. Nutr. Hosp. [online]. 2021, vol.38, n.4, pp.877-878. 20-Sep-2021. ISSN 1699-5198.Â https://dx.doi.org/10.20960/nh.03633