EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

Instituto Politécnico Nacional
"La Técnica al Servicio de la Patria"

Research in Computing Science

Vol. 151 No. 9
September 2022

Research in Computing Science

Research in Computing Science, Año 21, Volumen 151, No. 9, septiembre de 2022, es una publicación mensual, editada por el Instituto Politécnico Nacional, a través del Centro de Investigación en Computación. Av. Juan de Dios Bátiz S/N, Esq. Av. Miguel Othon de Mendizábal, Col. Nueva Industrial Vallejo, C.P. 07738, Ciudad de México, Tel. 57 29 60 00, ext. 56571. https://www.rcs.cic.ipn.mx. Editor responsable: Dr. Grigori Sidorov. Reserva de Derechos al Uso Exclusivo del Título No. 04-2019-082310242100-203. ISSN: en trámite, ambos otorgados por el Instituto Politécnico Nacional de Derecho de Autor. Responsable de la última actualización de este número: el Centro de Investigación en Computación, Dr. Grigori Sidorov, Av. Juan de Dios Bátiz S/N, Esq. Av. Miguel Othon de Mendizábal, Col. Nueva Industrial Vallejo, C.P. 07738. Fecha de última modificación 01 de septiembre de 2022.

Research in Computing Science, year 21, Volume 151, No. 9, September 2022, is published monthly by the Center for Computing Research of IPN.

Volume 151(9)

Advances in Computing Science and Applications

Hiram Calvo-Castro (ed.)

Instituto Politécnico Nacional, Centro de Investigación en Computación
México 2022

ISSN: in process

Formerly ISSNs: 1870-4069, 1665-9899

Instituto Politécnico Nacional (IPN)
Centro de Investigación en Computación (CIC)
Av. Juan de Dios Bátiz s/n esq. M. Othón de Mendizábal
Unidad Profesional "Adolfo López Mateos", Zacatenco
07738, México D.F., México

http://www.rcs.cic.ipn.mx
http://www.ipn.mx
http://www.cic.ipn.mx

Indexed in LATINDEX, DBLP and Periodica

Electronic edition

ISSN 1870-4069

Table of Contents

Page

ISSN 1870-4069

Clasificador de gestos motrices utilizando vectores de atributos distancia DTW sobre series de tiempo en representación SAX

Luis E. Valle[1], Rodolfo Romero[2], Jesús Y. Montiel[3]

[1] Instituto Politécnico Nacional,
Escuela Superior de Cómputo,
México

[2] Instituto Politécnico Nacional,
Escuela Superior de Cómputo,
Laboratorio de Posgrado en Sistemas Computacionales Móviles,
México

[3] Instituto Politécnico Nacional,
Centro de Investigación en Computación,
Laboratorio de Robótica y Mecatrónica,
México

lvalle212@gmail.com, rromeroh@ipn.mx, yalja@ipn.mx

Resumen. Este trabajo presenta una revisión del análisis de series de tiempo aplicado a la clasificación y reconocimiento de patrones. El artículo indaga en 3 métodos y compara su desempeño para la tarea específica del reconocimiento de gestos motrices empleando series de tiempo multivariables que derivan de la medición en la aceleración en los 3 ejes espaciales sobre un dispositivo sensor acelerómetro. Como aportación, se plantea un nuevo método capaz de superar la técnica de referencia y modelos del estado del arte con un 94.06 % de precisión en el conjunto de datos *GesturePeeble*; disponible en *The UCR Time Series Archive*. El método propuesto implementa el algoritmo *Dynamic Time Warping* para la conformación de vectores con atributos de distancia entre series de tiempo representadas con *Symbolic Aggregate Approximation*, posibilitando para las fases posteriores (entrenamiento y predicción) su ingreso en el modelo elegido. El método permite la elección del clasificador a conveniencia de la aplicación, optando para los resultados reportados en este trabajo, un modelo clásico de Máquinas de Soporte Vectorial.

Palabras clave: Clasificación de series de tiempo, deformación temporal dinámica (DTW), aproximación simbólica agregada (SAX).

Luis E. Valle, Rodolfo Romero, Jesús Y. Montiel

Motor Gesture Classification Using DTW-Based Feature Vectors on Time Series with SAX Representation

Abstract. This work features a review of the time series analysis applied to pattern classification and recognition. The paper deepens in 3 methods and compares their performance for the specific task of gesture recognition using multivariate time series derived from the measurement in the 3-spatial-axis acceleration on an accelerometer sensor device. As a contribution, a new method capable of surpassing the reference technique and state-of-the-art models with 94.06 % accuracy in the GesturePebble dataset is introduced; dataset available in the UCR Time Series Archive. The proposed method implements the Dynamic Time Warping algorithm for the composing of distance feature vectors between time series in Symbolic Aggregate Approximation representation, providing for the subsequent phases (training and prediction) their input in the chosen model. This method allows the classifier to be elected at the application's convenience, designating for the reported results in this work, a classical Support Vector Machine model.

Keywords: Time series classification, dynamic time warping, symbolic aggregate approximation.

1. Introducción

El reconocimiento de gestos es materia de interés en un amplia variedad de áreas del conocimiento, este trabajo se desarrolla en el marco específico de una propuesta que tiene como objetivo la creación una herramienta embebida que facilite la comunicación verbal para pacientes que recientemente han sufrido una lesión por traumatismo craneoencefálico o un accidente cerebrovascular que ha dañado principalmente el lóbulo izquierdo del cerebro.

El tipo de lesiones que derivan de accidentes de esta naturaleza generalmente ocasionan 3 condiciones médicas típicas: Las afasias[1], apraxias[2] y disartrias[3]. Los síntomas de estas condiciones se manifiestan como alteraciones del lenguaje, pérdida de la capacidad para la realización de gestos diestros aun cuando se mantiene el deseo y capacidad física de hacerlos, así como trastornos en la ejecución motora del habla.

La herramienta propuesta que permitirá al paciente conformar palabras replicando gestos de un código motriz, parte de la propuesta, atraviesa por una etapa de conformación de texto y finalmente de producción oral. Lo que compete a este artículo, en específico es la investigación de las técnicas y métodos disponibles para el desarrollo del algoritmo que se encargará de la clasificación de los gestos, como series dependientes del tiempo, a clases alfabéticas. Considerando el contexto al que aporta este trabajo, se toma especial consideración en la elección del conjunto de datos prueba y el desarrollo

experimental, esto con la motivación de replicar lo más fielmente las condiciones en las que funcionará la propuesta de herramienta.

Existen 2 decisiones especialmente importantes que se consideran para la elección del modelo de reconocimiento. La primera de estas es la naturaleza del código motriz, el cual en este caso será dependiente únicamente de las mediciones de un sensor acelerómetro. Esta decisión aporta grandes posibilidades al método de reconocimiento, como por ejemplo la posibilidad de intercambiarse de un prototipo de módulo sensor a dispositivos de uso común, tal como pueden ser relojes inteligentes y otros tipos de dispositivos *wearable*.

La segunda condición es la limitante en hardware disponible en un sistema embebido. Originalmente esta propuesta considera un entorno de RaspberryPi 4 para el algoritmo de reconocimiento, dispositivo en el que no es conveniente utilizar técnicas de aprendizaje profundo o algoritmos costosos en tiempo de ejecución computacional (exclusivamente en la etapa de predicción) motivación por el cual este trabajo busca alternativas que ofrezcen una alta precisión de predicción sin comprometer requerimientos de *hardware* no disponibles en un sistema en chip u ordenador en una placa.

2. Trabajos relacionados

Las series de tiempo, inherentemente descritas por una noción de ordenamiento en la que el tiempo es la unidad más típica, les mantiene constantemente presentes en casi cualquier tarea que involucra un proceso cognitivo humano[4], sucediendo a su vez en muchos otros fenómenos recurrentes en la naturaleza y cantidad de ámbitos de interés humano como negocios, economía, ingeniería, medio ambiente, medicina y otras áreas de investigación científica que recolectan datos en forma de secuencias dependientes del tiempo[5].

La tarea de clasificación de series temporales o *Time Series Classification*(TSC), consiste en el entrenamiento de un clasificador en un conjunto de entradas de entrenamiento en unos casos específicos donde cada uno contiene un conjunto ordenado de atributos en valores reales y una etiqueta de clase[6]. Se trata de un tema ampliamente estudiado y de gran interés en un extenso rango de áreas como *data mining*, estadística, procesamiento de señales, ciencias ambientales, biología computacional, procesamiento de imágenes, quimiometría, etc[6]. Representa tal importancia e influencia que cualquier problema de clasificación que utiliza datos registrados tomando en consideración una noción de ordenamiento puede ser transformado a un problema de TSC [4].

Con los años, investigadores en el tema han invertido un gran esfuerzo en estudios para el desarrollo de modelos que resuelvan el problema planteado por TSC y cada vez con una mayor precisión. Con la creciente disponibilidad de datos temporales[4] y archivos como *UCR Time Series Archive*[7], se ha propiciado el crecimiento en el número de algoritmos propuestos para TSC, implementando técnicas que utilizan modelos de aprendizaje máquina y diferentes medidas de distancia, técnicas para la transformaciones del espacio de los datos[6], técnicas de *ensembling*(entrenamiento de un conjunto de modelos menores que

integrando sus salidas resultan en un mejor modelo)[8], hasta nuevas alternativas de modelos de aprendizaje profundo en tendencia para muchas áreas de la IA, con una influencia relativamente nueva pero creciente en lo referente al problema de TSC[4].

Retomando el marco de este trabajo con la propuesta de herramienta de apoyo embebida como facilitadora de la comunicación verbal para personas con afecciones del habla, existen trabajos como [9,10,11,12], que atienden un problema similar pero dirigen su enfoque en la traducción de gestos pertenecientes a un lenguaje de señas, estas aplicaciones plantean una condición primordial, el previo conocimiento y dominio del lenguaje señado. Con gran diferencia la situación que enfrentan los pacientes que recientemente han sufrido de alguna lesión encefálica es distinta, no cumplen con la condición del dominio o conocimiento del lenguaje de señas, lo que es más, la mayoría contrario a eso y comprometido por el control pleno o parcial del lenguaje oral, desconocen un lenguaje complejo como el Lenguaje de Señas Mexicano(LSM), en el caso específico de México, sucediendo de la misma manera para gran porcentaje de la población en la sociedad que no convive en su entorno con una persona con afectaciones en el habla.

Aunado al complejo proceso de adquisición de un lenguaje señado, que equivale al aprendizaje de un nuevo idioma, se tienen los síntomas como la incapacidad para realizar gestos diestros y finos, como el movimiento controlado y preciso de los dedos para algunos gestos del lenguaje. La decisión de utilizar el acelerómetro permite entonces mantener en un mínimo el *hardware* sensor, puede portarse sin ser invasivo y es mucho más preciso para describir gestos amplios(movimiento de brazos y manos) más fácilmente ejecutables por pacientes con las afecciones descritas en[1,2,3].

Enfocándose principalmente en el objetivo que atiende este trabajo se distingue un artículo que atiende en profundidad, y de manera particular, la identificación de gestos motrices utilizando un sensor acelerómetro para el muestreo de las fuerzas en los 3 ejes durante la ejecución del patrón con una mano[13]. En el trabajo, Mezari y Maglogiannis, utilizan un *smartwatch* Pebble como elemento sensor, exponiendo el objetivo del trabajo como una examinación del uso de dispositivos básicos como *smartphones* y *wearables* para el reconocimiento confiable de gestos simples y naturales, proponiendo además una metodología para mejorar el desempeño y precisión. De la metodología elaborada por Mezari y Maglogiannis en su artículo[13] se toma parte de inspiración para desarrollar la propia del trabajo, siendo a la vez su método uno de los 3 evaluados, así como su principal componente en el método híbrido aportado por este artículo.

3. Descripción del conjunto de datos

Manteniendo en mente las condiciones del proyecto, se eligió acorde un conjunto de datos para el entrenamiento y prueba de los métodos experimentados. El conjunto de datos elegido *GesturePebble*, se deriva

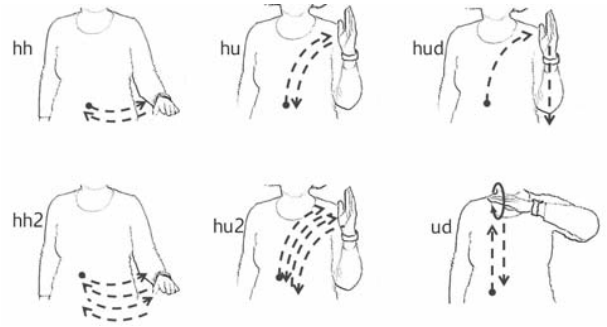

Fig. 1. Elección de los 6 gestos sencillos etiquetados y realizados por los participantes para la generación de las secuencias temporales disponibles en el *dataset GesturePebble*.

del trabajo *"Gestures Recognition using Symbolic Aggregate Aproximation and Dynamic Time Warping in Motion Data"*[13], el conjunto de datos se encuentra disponible para su uso público en el repositorio de series temporales *UCR Archive* [7].

El conjunto de datos resulta del estudio de dispositivos cotidianos como celulares y *wearables* para el reconocimiento de gestos, por lo que la recolección de las muestras se realizan con mediciones del acelerómetro del reloj inteligente Pebble en los 3 ejes espaciales mientras el dispositivo es utilizado en la muñeca por los participantes.

Incluye instancias de series temporales con longitud variable con un máximo de 456 mediciones por patrón. Los archivos provistos por UTC[7] colocan la etiqueta de clase como primer elemento de cada secuencia y aquellas series de tiempo de longitud menor han sido acompletadas con *NaNs*.

El manejo de los elementos *NaNs* se realiza en el código de la implementación al eliminarseles, esta acción otorga como resultado una serie de tiempo con la longitud de mediciones reales tomadas, más tarde esto no representa un inconveniente para los métodos de *Symbolic Aggregate Approximation*(SAX) y *Dynamic Time Warping*(DTW), ambos algoritmos manejan nativamente secuencias de longitudes variables y a su vez SAX procesa cada serie de forma individual.

El *dataset* involucra a 4 participantes, cada uno instruido para realizar 6 gestos(Figura 1) que repitieron durante un par de sesiones separadas entre días para conseguir un total de 8 grabaciones y completando el *dataset* a un total de 304 gestos.

De las series temporales captadas, se proveén las mediciones del sensor acelerómetro en el eje Z únicamente, facilitando a su vez un par de conjuntos de datos con características específicas cada uno: *GesturePebbleZ1* y *GesturePebbleZ2*.

El conjunto de datos *GesturPebbleZ1*, proveé un archivo con instancias para el entrenamiento del modelo clasificador, componiendose los patrones desarrollados

por los participantes durante la primera sesión. Mientras que el archivo de prueba, es la colección de los patrones realizados en la segunda sesión.

Para el conjunto *GesturePebbleZ2* el conjunto de entrenamiento consiste en los datos de un par de sujetos, mientras el conjunto de prueba se conforma del par de sujetos restante. Presumiblemente esta segunda colección está destinada a ser más compleja en dificultad que el primero, impidiendo que el modelo sea entrenado con patrones de los sujetos que más tarde serán evaluados. La dificultad de este *dataset* se argumenta basándose en el hecho de que cada participante posee una postura y mecánica de movimiento distinta a las de los demás.

4. Definición de algoritmos y técnicas utilizadas

4.1. *Dynamic Time Warping*

Dynamic Time Warping o DTW, es una técnica bien conocida para encontrar una alineación óptima entre 2 secuencias dependientes del tiempo(series de tiempo) dadas y bajo ciertas restricciones[14]. Este algoritmo es sumamente útil para medir la similaridad entre dos secuencias temporales que no se alinean exactamente en el tiempo, velocidad o extensión[14]. Originalmente este algoritmo había sido utilizado para comparar patrones del habla en reconocimiento de habla automático, siendo aplicado en otros campos de forma exitosa para lidiar con deformaciones en el tiempo y diferentes velocidades.

El objetivo de DTW es la comparación de 2 secuencias dependientes del tiempo X y Y, siendo series discretas, o más generalmente una secuencia de características muestreadas a puntos equidistantes en el tiempo.

$$\begin{aligned} X &= (x_1, x_2, ..., x_N) \;\; N \in \mathbb{N} \\ Y &= (x_1, x_2, ..., x_M) \; M \in \mathbb{N}. \end{aligned} \tag{1}$$

Con un *espacio de características* denotado por $\mathcal{F}$, entonces $x_n, y_m \in \mathcal{F}$ con $n \in [1 : N]$ y $m \in [1 : M]$. Para comparar dos características diferentes se realiza mediante una *medida de costo local*, también llamada *medida de distancia local* definida por:

$$c : \mathcal{F} \times \mathcal{F} \to \mathbb{R}_{\geq 0}. \tag{2}$$

La implementación de la *medida de distancia local* depende de la medida de distancia general definida, para el documento se implementa la distancia Euclidiana c_e, definida como: $c_e(x_i, y_j) = (x_i - y_j)^2$ donde $i \in [1 : N]$ y $j \in [i : M]$, pero otro tipo de medidas de distancia local pueden utilizarse, como por ejemplo L1 o distancia *Manhattan*.

Típicamente *c(x,y)* es pequeña(bajo costo) si x y y son similares, de otra forma esta es alta. Evaluando el costo local medido para cada par de elementos de las secuencias X y Y, se obtiene la *matriz de costo* $C \in \mathcal{R}^{N \times M}$ definida por $C(n, m) := c(x_n, y_m)$ entonces la meta es encontrar una alineación entre X y Y obteniendo el costo total mínimo.

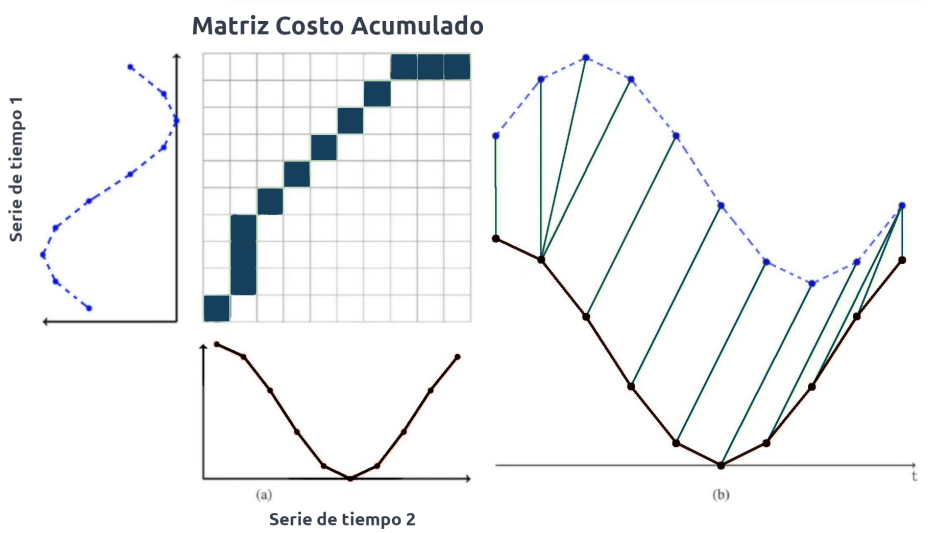

Fig. 2. Matriz de costo acumulada del cálculo de la distancia entre la Serie de tiempo 1 y Serie de tiempo 2.
a) Los recuadros rellenos en la cuadrícula conforman el camino de deformación óptimo(*optimal warping path*) p^* en la matriz de costo.
b) Resultado de la alineación óptima para la Serie de tiempo 1 en la Serie de tiempo 2 utilizando DTW.
Figura recuperada de: Thales Sehn Körting(Productor).(2017) *How DTW (Dynamic Time Warping) algorithm works*[YouTube]. https://www.youtube.com/watch?v=_K1OsqCicBY.

Formalmente un camino de deformación o *warping path*, es una secuencia $p = (p_1, ..., p_L)$ con $p_\ell = (n_\ell, m_\ell) \in [1:N] \times [1:M]$ para $\ell \in [1:L]$ satisfaciendo las siguientes 3 condiciones:

(I) *Condición límite*: $p_1 = (1,1)$ y $p_\ell = (N,M)$
(II) *Condición Monotonicidad*: $n_1 \leq n_2 \leq ... \leq n_L$ y $m_1 \leq m_2 \leq ... \leq m_L$
(III) *Condición del tamaño del paso*: $p_{\ell+1} - p_\ell \in (1,0),(0,1),(1,1)$ para $\ell \in [1 : L-1]$

Un *camino de deformación-(N,M)* $p = (p_1, ..., p_L)$ define una alineación entre las secuencias X y Y al asignar el elemento x_{n_ℓ} al elemento y_{m_ℓ}. Un ejemplo de la aplicación de las 3 condiciones durante la construcción del camino óptimo se observa en la Figura 2, donde utilizando la matriz de costo acumulado se marca una secuencia, con las casillas rellenas, de las distancias óptimas satisfaciendo siempre las condiciones(i,ii,iii).

El costo total $c_p(X,Y)$ de un camino de deformación p entre X y Y con respecto a la medida de costo local c se define como:

$$c_p(X,Y) = \sum_{\ell=1}^{L} c(x_{n_\ell}, y_{m_\ell}). \tag{3}$$

Mientras que un camino de deformación óptimo entre X y Y es un camino de deformación p^* que tiene un costo total mínimo entre todos los posibles caminos de deformación.

Finalmente, la distancia DTW entre X y Y es definido entonces como el costo total de p^*:

$$\begin{aligned} &DTW(X,Y) = c_{p^*}(X,Y), \\ &DTW(X,Y) = min\{c_p(X,Y)|p : CaminoDeformación - (N,M)\}. \end{aligned} \tag{4}$$

4.2. Banda Sakoe-Chiba

DTW es un algoritmo muy popular por sus excelentes resultados en la comparación de secuencias temporales, y sin embargo debido a la construcción de la matriz de costos, su complejidad cuadrática es un detractor significativo en tiempo y precisión de cálculo. Una variante común en DTW que aborda esta situación es la implementación de la banda Sakoe-Chiba[15].

De forma general, la implementación de restricciones a DTW aporta un par de beneficios. El primero de ellos y el más importante si se plantea el objetivo de la optimización, es la reducción de complejidad con la implementación de la banda, forzando al cálculo de un subconjunto del espacio global e inmediatamente reduciendo su complejidad de $O(L^2)$ a $O(w \times L)$ con $0 \geq w \leq L-1$, donde w define una banda.

El segundo de los beneficios refiere en cuanto a la fiabilidad de la similitud, pues previene alineaciones patológicas, evitando la desviación excesiva de la diagonal principal al camino óptimo posible.

La banda Sakoe-Chiba corre a lo largo de la diagonal principal y tiene un ancho fijo $T \in \mathbb{N}$ horizontal y verticalmente. Esta restricción implica que para un elemento x_n puede ser alineado únicamente a uno de los elementos de y_m con $m \in \left[\frac{M-T}{N-T} \times (n-T), \frac{M-T}{N-T} \times n + T\right] \cap [1:M]$. En la matriz de costo acumulado la ventana de deformación(*warping window* o WW) se observa como en la Figura 3.

4.3. Optimización del tamaño de la ventana de DTW

El algoritmo DTW es un método robusto y con una extraordinaria competitividad, además de utilidad, probando ser una medida de distancia para series de tiempo excepcionalmente fuerte, razón por la que es de gran importancia para la comunidad el conocimiento de la capacidad que logra el algoritmo cuando se optimiza correctamente el tamaño de la ventana w, la cual es crítica en la disminución del error durante la predicción[16].

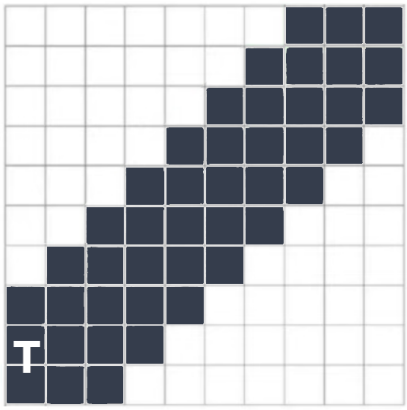

Fig. 3. Banda de Sakoe-Chiba que corre sobre la diagonal principal de la matriz y que tiene un ancho fijo T.

Argumentando que la restricción en su cantidad máxima de deformación, cuando es establecida correctamente, cierra la mayoría de los espacios de mejora obtenidos con métodos de clasificación de series de tiempo "más sofisticados". El artículo[17] propone un método nobel para el aprendizaje del parámetro óptimo en configuraciones supervisadas y no supervisadas, siendo el caso primero el que compete a este trabajo.

Existe una concepción errónea de que el valor de tamaño de ventana w puede ser dependiente del dominio de cálculo requiriendo su cálculo por única vez, cuando en realidad no existe un solo valor de w que pueda ser transferible entre diferentes contextos. El valor óptimo para la banda restrictiva de DTW dependerá de 2 factores, el número de instancias del conjunto de datos y la estructura propia de los datos, contundentemente abatiendo la posibilidad de la existencia de un valor prototípico de w para aplicaciones específicas como ritmos cardíacos o gestos[17].

Formalmente el problema se plantea como: Dado un conjunto de entrenamiento de series de tiempo etiquetadas, encontrar el valor de w(Tamaño de ventana Sakoe-Chiba) que maximiza la calidad de clasificación en un conjunto de prueba sin etiquetar.

La evaluación de la calidad de clasificación se realiza mediante la medida de la precisión.

Debido al creciente consenso que ubica al método DTW-k-NN como una línea base robusta, deciden los autores del trabajo[17] utilizar el algoritmo k-NN como el clasificador subyacente. Como se describe en secciones posteriores, este método es utilizado en todos los modelos evaluados, de forma que clasificadores como k-NN y SVM son implementados también con diferencia al artículo original[17].

El enfoque que toma este método es particularmente útil cuando el conjunto de entrenamiento es limitado, pues implementa una técnica de remuestreo con

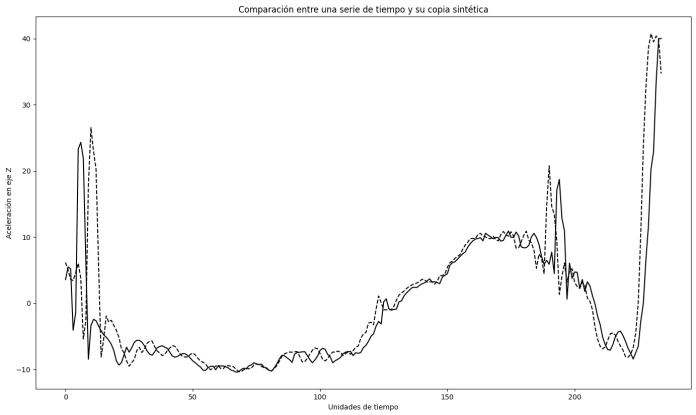

Fig. 4. Comparación de una instancia perteneciente al conjunto de datos *GesturePebble*(línea continua) y su remplazo sintético(línea punteada).

la creación de datos sintéticos que reemplazan al problema de un número de instancias limitadas.

El algoritmo consiste en hacer N copias del conjunto de entrenamiento original, reemplazando para cada copia una fracción de los datos con reemplazos sintéticos, para posteriormente realizar validación cruzada con el fin de aprender el porcentaje de error contra la curva de valores de w. Se utiliza el valor promedio de todas las iteraciones N para la predicción del mejor tamaño de ventana w.

De los componentes lógicos que conforman al método, los de mayor interés son el proceso de creación de un nuevo conjunto de datos con una porción sintética, y la función capaz de otorgar una deformación a una secuencia para otorgar un dato sintético.

El proceso de creación de un nuevo conjunto de datos con instancias sintéticas inicia por dividir aleatoriamente el conjunto de datos cumpliendo con una relación especificada de porcentaje de datos sintéticos sustitutos, mientras la otra partición permanece sin modificación. Posteriormente utilizando *K-Fold Cross Validation*, se realiza la división en conjunto de entrenamiento y prueba para medir iterativamente la calidad del clasificador.

La función capaz de sintetizar una serie de tiempo inicia con el encogimiento no lineal de la medición real mediante la eliminación aleatoria de un porcentaje definido por el usuario, de los datos que la componen. Seguido a esto, utilizando una función común a bibliotecas de procesamiento de señales, la nueva serie debe muestrearse una vez más a la misma longitud de la secuencia original, agregando antes de este proceso, un acolchado(*padding*) de la señal repitiendo en los extremos los últimos y primeros 10 valores respectivamente. Finalmente y terminado el remuestreo, se eliminan los valores de acolchado en los extremos y se obtiene una serie sintética de longitud igual a la original.(Figura 4)

Esta técnica requiere la configuración de 3 parámetros para su funcionamiento, utilizando como valores constantes para la experimentación los

valores propuestos por los mismo autores del artículo[17], 20 % de deformación para la creación de datos sintéticos, una relación de 8 a 2 para la creación del nuevo conjunto de datos con copias sintéticas como número de instancias mayoritarias, y finalmente para el número de iteraciones *N*, mientras mayor sea su valor mejor, considerándose de forma conservadora en 10.

4.4. *Piecewise Aggregate Approximation*

Piecewise Aggregate Approximation o PPA, es un algoritmo que tiene como idea básica la reducción dimensional de una serie de tiempo de entrada mediante la partición de esta en segmentos del mismo tamaño, sobre cada cual se realiza el cálculo promedio de los valores en el segmento[18]. Con una serie de tiempo $Y = Y_1, Y_2, ..., Y_n$ con tamaño $n \in \mathbb{R}$ la partición o reducción a una serie $X = X_1, X_2, ..., X_m$ donde $m \leq n$, la ecuación que describe los elementos en la serie reducida es:

$$\overline{X_i} = \frac{m}{n} \sum_{j=\frac{n}{N(i-1)+1}}^{\frac{n}{M}i} x_j. \tag{5}$$

Es importante resaltar que antes de realizar la aproximación el vector debe ser z-normalizado, y una vez haya sido estandarizado la aproximación por partes se calcula.

Se identifican 2 casos durante la separación en segmentos. El caso trivial, $m < n$ y m es múltiplo de n, se trata dividiendo el vector en ventanas de tamaños iguales y realizando el cálculo del promedio para la asignación del nuevo valor por segmento. El segundo cuando $m < n$ y m no es múltiplo de n, no existe el balance para la división exacta por lo que es requerido un redimensionado de cada ventana que permita a cada elemento de la serie de salida ser el promedio de un segmento de mismo tamaño al vector de entrada[18].

Se observa un ejemplo en la Figura 5 de la discretización de un par de series de tiempo con pocos atributos pero de extensión distinta, que después de la transformación PAA son equiparables en longitud con un número de palabras *m=9*.

4.5. *Symbolic Aggregate Approximation*

Symbolic Aggregate Approximation o SAX, es una técnica desarrollada y enfocada en la reducción dimensional de una serie numérica con una serie de tiempo, a un espacio simbólico de 'palabras'. Dada una serie dependiente del tiempo de longitud arbitraria n se realiza la transformación a una cadena de longitud w' utilizando un alfabeto $A = a_1, a_2, ..., a_3$[19].

La primer parte de la discretización es manejada por la transformación PAA, siguiendole el proceso de asignación de símbolos para cada sección, conformando como requisito para esta segunda parte de la discretización que los símbolos sean asignados equi probablemente (propiedad de una colección de eventos teniendo

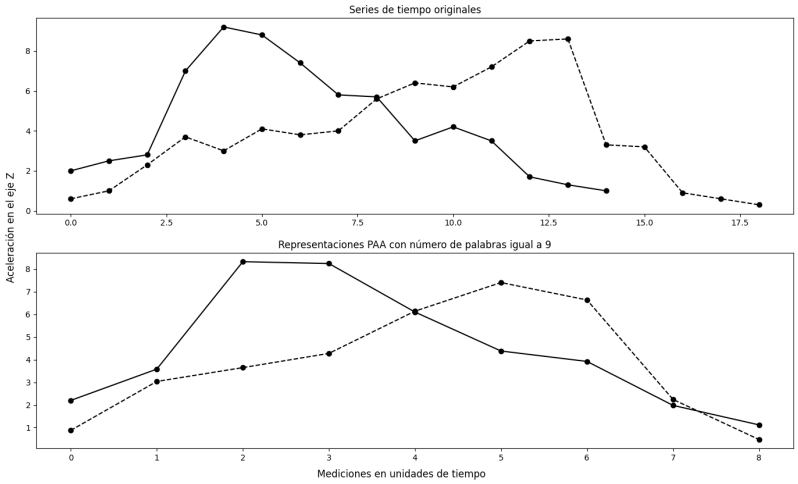

Fig. 5. El algoritmo PAA mantienen las tendencias de las secuencias.

la misma probabilidad de ocurrir). Esto se cumple fácilmente por la previa normalización de los datos de la serie en una distribución Gausiana, razón por la que las áreas bajo la curva de campana de la distribución pueden ser utilizadas para la creación de los puntos de quiebre(*breakpoints*) en los datos normalizados para la asignación de palabras.

El método algorítmico toma los siguientes procesos en orden [13] (Figura 6):

1. Estandarización o normalización-z de los datos de la serie de tiempo para media $\mu = 0$ y desviación estándar $\sigma = 1$.
2. Transformación o reducción de dimensionalidad desde n a w' mediante el algoritmo *Piecewise Aggregation Approximation*(PAA).
3. Asignación de los puntos de quiebre $\beta = \beta_1, ..., \beta_{\alpha-1}$.
4. La serie de tiempo es discretizada tomando el promedio de cada segmento para ser mapeado a un alfabeto A.

La distancia entre 2 palabras o cadenas, correspondiendo cada una a diferentes series de tiempo, se calcula como el promedio de los pares de las distancias de símbolos.

Los puntos de quiebre es una lista de números $\beta = \beta_1, ..., \beta_{\alpha-1}$ tal que el área debajo la curva gausiana de β_i a $\beta_{i+1} = \frac{1}{\alpha}$[19].

Estos puntos de quiebre pueden ser obtenidos mirando una tabla estadística y puede ser utilizada para la discretización de series de tiempo donde el coeficientes debajo el punto de quiebre más pequeño son mapeados a la letra del alfabeto en el índice 1, mientras los coeficientes mayores o igual al punto de quiebre más pequeño y menor al segundo punto de quiebre se le asigna la letra del alfabeto con el segundo índice, y así sucesivamente.

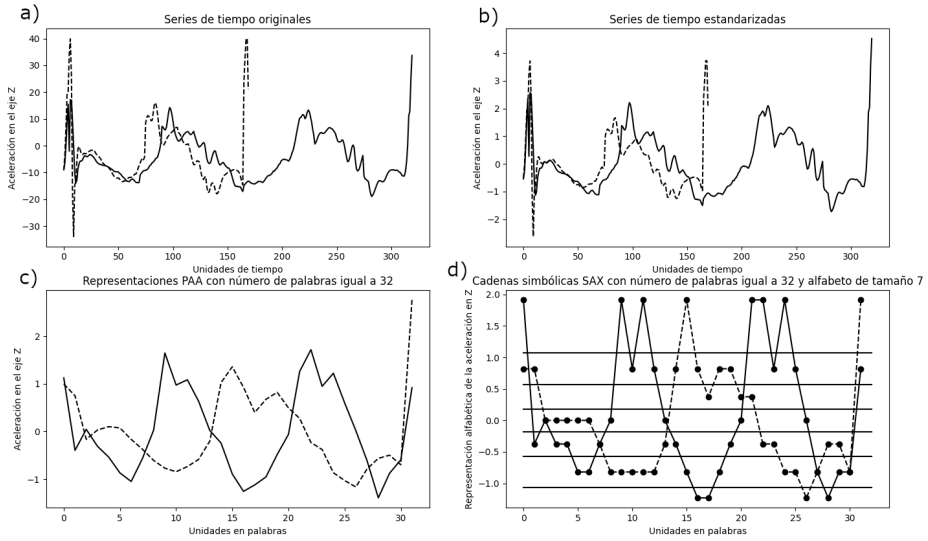

Fig. 6. Graficación de un par de series de tiempo tomadas del conjunto de datos *GesturePebble* durante el proceso completo de la discretización SAX
a) Series de tiempo de longitud variables con valores reales(originales)
b) Series de tiempo después de estandarización.
c) Series de tiempo después de la transformación PAA con un número de palabras igual a 32.
d) Series de tiempo en su representación SAX como una cadena de símbolos numéricos con alfabeto de tamaño 7.

La distancia entre 2 símbolos se define a 0 si el índice difiere a lo más por 1(por ejemplo resulta 0 entre símbolos a_i y a_{i+1}), en cualquier otro caso la distancia entre símbolos a_i y a_k, donde $k > i$, se define como $b_{k-1} - b_i$[13].

La distancia entre 2 cadenas correspondientes a diferentes series de tiempo, se calcula como el promedio de las distancias de los símbolos por parejas(por ejemplo; el promedio de la distancia entre el primer símbolo de cada serie, la distancia entre el segundo símbolo de cada serie, y así sucesivamente)[13].

4.6. *DTW Barycenter Averaging*

DTW Barycenter Averaging o DBA, es un algoritmo iterativo que utiliza DTW para la alineación de series de tiempo utilizando un promedio envolvente[20]. Introducido por Petitjean[20], la implementación del algoritmo DBA trae consigo varias ventajas importantes frente a un proceso plano o simple de promediación de un conjunto de secuencias.

El algoritmo *grosso modo*, opera de la siguiente manera:

1. Las n series por ser promediadas son etiquetadas $S_1, S_2, ..., S_n$ y tienen una longitud T.

2. Se inicia el proceso con una serie abreviada inicial *I*.
3. Se realizan los siguientes hasta que el promedio converge:
   a) Por cada serie *S*, se aplica DTW contra *I* y se salva el camino de deformación.
   b) Se utiliza el camino de deformación y se construye un nuevo promedio de *I* al dar a cada punto un nuevo valor: El promedio de cada punto de *S* conectado a este en el camino de deformación resultante de DTW.

Una buena inicialización del proceso con un candidato correcto para *I* es de extrema importancia, porque mientras el proceso DBA por sí mismo es determinístico el resultado final dependerá esencialmente de la secuencia abreviada inicial. Se identifican entonces 3 objetivos distintivos:

– Preservar la forma de las entradas.
– Preservar la magnitud de los extremos en el eje y.
– Preservar la sincronización de aquellos extremos en el eje x.

En el trabajo[20] se recomienda inicializar DBA eligiendo de forma aleatoria la serie abreviada de inicio *I*, y en un principio esta técnica preserva bien la forma, pero la sincronización de los extremos dependerá en cual serie resulta escogida, razón por la que un proceso determinístico de elección es preferible.

Un ejemplo de la aplicación de este algoritmo en el modelo propuesto se muestra en la figura 7, donde utilizando un conjunto de series de la misma clase se calcula una serie de tiempo representante de toda la clase.

5. Métodos de clasificación

5.1. DTW 1-NN

Un modelo obligatorio como punto de partida entre las vastas alternativas de técnicas que atienden el problema TSC, es a través de una combinación del algoritmo DTW y el modelo *k-NN* para la clasificación de secuencias dependientes del tiempo, mostrando sus capacidades de precisión y robustez cuando se compara con técnicas más poderosas[21].

Esta robusta técnica combina la medida de similitud entre secuencias de tiempo DTW y el algoritmo de clasificación k-NN(*k Nearest Neighbours*) en su variante con *k=1*, parámetro definido de forma empírica por infinidad de trabajos anteriores y para conformar actualmente el método base en el problema de TSC por la creciente aceptación por parte de la comunidad.

En esta variante del algoritmo k-NN, se sustituye la distancia Euclidiana como función de medida entre los datos por la medida de similitud DTW entre series de tiempo, resultando en su gran capacidad de clasificación con una buena precisión. Aún con los beneficios que otorga el modelo referencia, es importante revisar los detrimentos de la técnica frente a las limitaciones y condicionantes naturales de la problemática que se atiende. La gran debilidad de este algoritmo reside en la complejidad y tiempo de cálculo, dado principalmente por el uso de DTW como función de medida en el modelo clasificador.

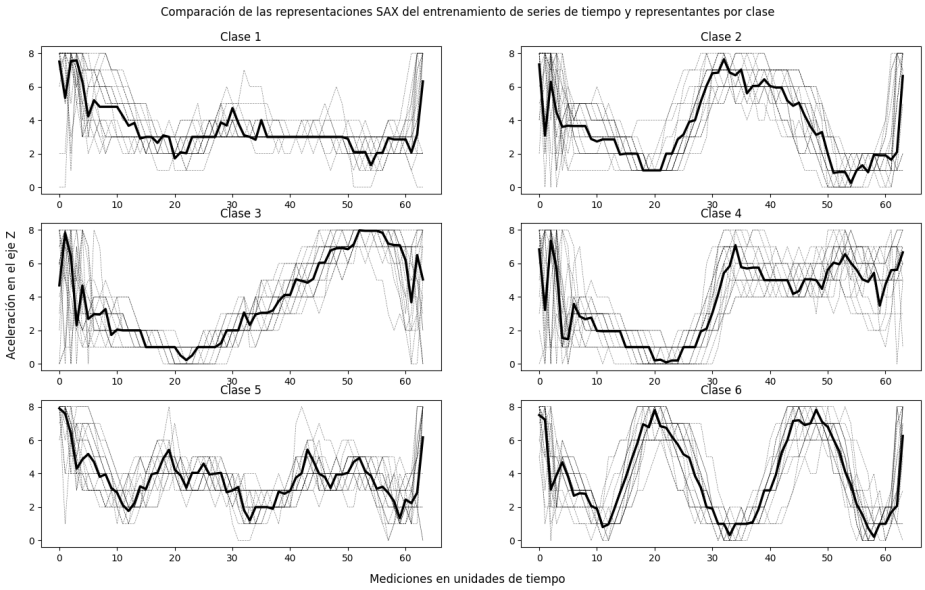

Fig. 7. Ejemplo de las series representantes de cada clase para el conjunto de datos *GesturePebble*. Cada representante(línea gruesa) fue calculada utilizando el algoritmo DBA con un promediado de 5 instancias discretizadas(línea delgadada y punteada) en representación SAX de alfabeto tamaño 9 y número de palabras 64.

La complejidad cuadrática de DTW, $O(N^2)$, cuando se evalúa la similitud de la series de tiempo que se intenta predecir con respecto al número de instancias que conforman el conjunto de datos de "entrenamiento", incrementa significativamente el número de cálculos derivado de la conformación de la matriz de costo del algoritmo. Frente a esta situación, se ha buscado la optimización del algoritmo utilizando las restricciones, en un intento de disminuir la complejidad en la conformación de la matriz de costo acumulada, inevitablemente requiriendo el aprendizaje del tamaño óptimo del parámetro tamaño de ventana w que restringe los cálculos a elementos más cercanos a la diagonal.

La versión desarrollada experimentalmente en este trabajo, implementa la técnica de optimización del tamaño de la ventana DTW[17] en la búsqueda de satisfacer las condiciones de la solución que exigen un modelo optimizado sin dejar de lado una buena precisión en la predicción.

5.2. Modelo SAX-DTW

Técnica propuesta en el artículo *"Gesture recognition using Symbolic Aggregate Approximation and Dynamic Time Warping in Motion Data"*[13], como el método más preciso de entre los 3 evaluados en aquel trabajo, supera a los demás al reportar una razón promedio de clasificación correcta igual al 99.21 %.

El método se basa en la coexistencia del algoritmo SAX y la función de medida DTW, logrando combinar las mejores características de ambos métodos; La efectividad y baja complejidad de SAX, con la insensibilidad de DTW a las fluctuaciones de velocidad durante la ejecución de un gesto[13].

Este segundo método, al igual que el de referencia, utiliza como clasificador subyacente a 1-NN, pero implementado como función de distancia la propia distancia SAX modificada para implementar la alineación óptima de DTW.

La totalidad de las secuencias temporales serán discretizadas a la representación SAX, tanto las instancias de conjunto de "entrenamiento", como los nuevos patrones ingresados para su etiquetado mediante predicción.

Importante para el método SAX aclarar los valores de los parámetros utilizados. En el trabajo desarrollan sus experimentos con un valor para el número de palabras igual a 32(Representación PAA) y un tamaño de alfabeto igual a 7.

Finalmente la función de medida adopta el concepto de la distancia entre 2 símbolos SAX con una modificación. En lugar de ocuparse la comparación por defecto donde 2 símbolos del mismo índice en diferentes cadenas se comparan(una comparación de distancia Euclidiana en el ámbito discreto simbólico de SAX), se ocupa el camino óptimo de deformación proponiendo una comparación de la distancia entre símbolos relacionados por el mismo camino óptimo de deformación DTW. En la Figura 8 se ilustra el uso del método SAX-DTW.

La implementación del algoritmo DTW en este método exige la optimización del parámetro tamaño de ventana w como restricción en forma de banda Sakoe-Chiba, aplicándose también durante los resultados de la experimentación el método de aprendizaje del tamaño óptimo de ventana[17].

5.3. Método de vectores de atributos distancia DTW

El trabajo del artículo *"Using dynamic time warping distances as features for improved time series classification"*[22] basa su metodología partiendo de la robustez de DTW como una medición de distancia para series de tiempo, y tiene como objetivo aprovechar tal fortaleza utilizando indirectamente DTW para la creación de nuevas características que pueden ser alimentadas posteriormente a un método de aprendizaje máquina estándar en vez de utilizarse con el típico 1-NN(*1 Nearest Neighbour*).

El autor Kate en su trabajo[22] describe la representación de las series de tiempo en términos de sus distancias DTW de entre cada uno de los ejemplos de entrenamiento, de tal forma que dada una serie de tiempo T y un conjunto de entrenamiento $D = [Q_1, Q_2, ..., Q_n]$ el vector de características resultante $Feature - DTW(T)$ de T construido utilizando DTW es simplemente:

$$DTW(T) = (DTW(T, Q_1), DTW(T, Q_2), ..., DTW(T, Q_n)). \quad (6)$$

Una de las grandes fortalezas que destaca el autor sobre su método, es la mejora en el desempeño gracias a la implementación de algoritmos más

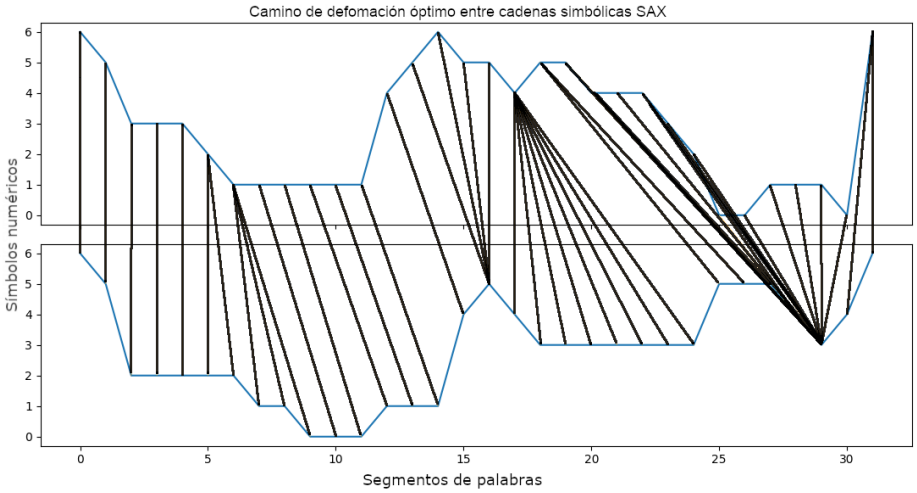

Fig. 8. Algoritmo del camino de deformación óptima calculada por DTW sobre 2 cadenas simbólicas SAX.

avanzados de aprendizaje máquina, eligiendo para la experimentación en su trabajo un modelo de Máquinas de Soporte Vectorial(SVM), indicando que de esta manera su método es capaz de aprender como la clase de una serie de tiempo se relaciona con sus distancias DTW de entre varios ejemplos de entrenamiento[22].

El método se distingue por ser además fácilmente extendible al utilizarse en combinación con otros métodos basados en características estadísticas y simbólicas añadiendolas simplemente como características adicionales. En el mismo trabajo Kate[22] desarrolla una variante de su método que concatena las características distancias DTW con características bolsa de palabras de SAX, reportándose en sus experimentos como el mejor clasificador no ensamble.

5.4. Método vectores de atributos distancia DTW de series de tiempo SAX

También nombrado Atributos SAX-DTW o *SAX-DTW Features*, es el método original que se propone en este trabajo, y que busca al igual que el modelo SAX-DTW, la coexistencia y adición de los beneficios que ofrece la disminución dimensional y discretización por parte de las representaciones SAX, con la robustez en el proceso de medición de la similitud de DTW para series con extensión y velocidades distintas. Aún cuando ambos algoritmos se implementan en un mismo modelo, el enfoque y filosofía de funcionamiento del modelo difiere de SAX-DTW, implementado una técnica desarrollada en el artículo[22] que propone representar a una serie de tiempo como un vector conformado por atributos calculados con las distancias DTW con respecto a cada ejemplo de entrenamiento[22].

Así como este método toma inspiración de las técnicas desarrolladas en ambos artículos[13,22], ninguna de las técnicas se aplica directamente como son descrita por los autores en sus respectivos trabajos, esto por que su combinación requiere realizar modificaciones para cumplir con el objetivo y resultados esperados del método nuevo.

Un cambio especialmente importante en comparación al par de métodos del que toma inspiración, es la implementación de un algoritmo *Machine Learning* mucho más robusto que el sencillo k-NN. Se ocupa en cambio un clasificador de soporte vectorial a raíz de ser considerado una de las grandes fortalezas cuando se ocupa un modelo clasificador más sofisticado con los nuevos vectores conformados por las distancias DTW como atributos[22] en comparación del sencillo k-NN.

Fase de preprocesamiento de los datos La manera más sencilla de obtener una mejora en la precisión de la clasificación para secuencias dependientes del tiempo es la implementación de un método basado en características(Representación estadística o simbólica definida para una serie de tiempo), y aprovechando este hecho se requiere la transformación de los datos en un espacio alternativo donde las características discriminatorias pueden ser más fácilmente detectadas que si se compara con un clasificador más complejo que permanece operando en el contexto temporal[6].

El manejo del total de series de tiempo, tanto las instancias que se utilizarán para entrenar el modelo clasificador como las que se alimentarán para predecir su etiqueta, es a través de su representación en cadenas simbólicas SAX, que proveé además de una transformación a un espacio alternativo de dominio simbólico, una reducción dimensional, lo que traduce en una simplificación de la complejidad. Esto implica que ninguna de las etapas siguientes pueda realizarse sin antes transformar las secuencias en cadenas simbólicas con número de palabras iguales a 64 y su discretización en el dominio de valores con un alfabeto tamaño 9.

Pensando en aprovechar todo el potencial de las similitudes calculadas por DTW, se escoge un alfabeto numérico $A \in [0, 8]$ en las representaciones SAX, permitiendo a su vez utilizar sin realizar una previa modificación a los símbolos, la distancia DTW.

Fase de entrenamiento (Figura 9) Cuando se involucra un método de aprendizaje máquina como SVC, el modelo debe pasar por una etapa de entrenamiento antes de poder ofrecer la tarea de predicción, situación que no sucede con el algoritmo k-NN de los métodos previos, pues realmente la etapa de entrenamiento y predicción sucede en un proceso integral.

La creación de los vectores de atributos en este método es esencial, no solo por la posibilidad que ofrece de aplicarse como entrada a modelos robustos de *Machine Learning*, pero además diverge del trabajo[22] en el que se inspira y se toma una interpretación alternativa que impacta positivamente en las etapa de entrenamiento y predicción, pero siendo de mayor relevancia su ventaja durante la predicción; Disminuyendo significativamente el tiempo de cálculo

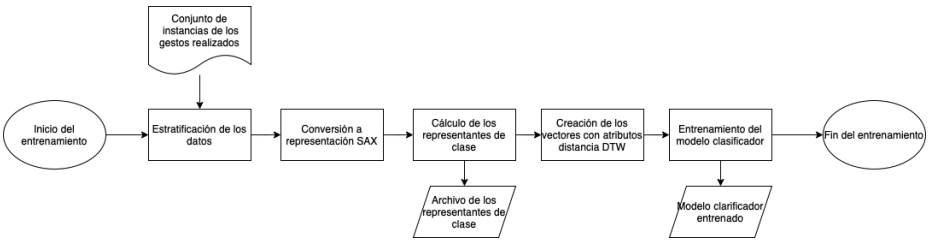

Fig. 9. Diagrama del algoritmo de entrenamiento para el modelo Atributos SAX-DTW

y complejidad requerido para la creación del vector de instancias por cada secuencia temporal.

Tomando un enfoque alternativo a la creación del vector de atributos, antes de pasar siquiera a esta etapa, se requiere crear una cadena simbólica representante de cada clase existente en el conjunto de datos, y como requisito se exige la estratificación del conjunto para mantener la equiprobabilidad de las clases durante el entrenamiento.

La fabricación del conjunto de cadenas representantes se logra mediante el promedio de todos los vectores existentes en el conjunto de datos que pertenecen a una misma clase, resultando el método más efectivo para lograr un representante fidedigno de cada etiqueta el algoritmo DBA[20].

Este conjunto de representantes resultante se convierte a partir de su cálculo en el conjunto de cadenas simbólicas más importantes del modelo. El hecho de que mediante la similitud DTW de estos representantes con los vectores destinados al entrenamiento y posteriormente los nuevos patrones a ser predecidos, exige la preservación del conjunto durante la vida útil del modelo entrenado. A pesar de tratarse de un algoritmo determinístico que sugiere el recálculo del conjunto de representantes frente a la pérdida de este, si la elección de la serie de tiempo abreviada inicial I se obtiene por un proceso aleatorio, un segundo cálculo del conjunto de representantes significa desechar el modelo entrenado para volver a crear las instancias de entrenamiento con este nuevo conjunto.

Creado el conjunto de cadenas representantes de cada clase, se calculan los vectores de atributos distancia DTW entre la instancia de entrenamiento y los representantes de clase(es importante el orden en que se ingresan ambas cadenas simbólicas, pues la operación de similitud DTW no es conmutativa), alimentando el modelo SCV para iniciar la etapa de entrenamiento.

Fase de predicción (Figura 10) Muestreado, filtrado y preprocesado un patrón de movimiento, el primer paso será representarlo mediante SAX en una cadena simbólica con el alfabeto numérico definido previamente A.

Como cadena simbólica, ahora esta secuencia puede ser usada para conformar un vector de atributos distancia DTW utilizando el mismo conjunto de representantes de clase usado para fabricar los vectores de entrenamiento.

Finalmente el vector resultante puede ser alimentado al modelo SVC previamente entrenado para la obtención de la etiqueta de clase resultante del proceso de predicción.

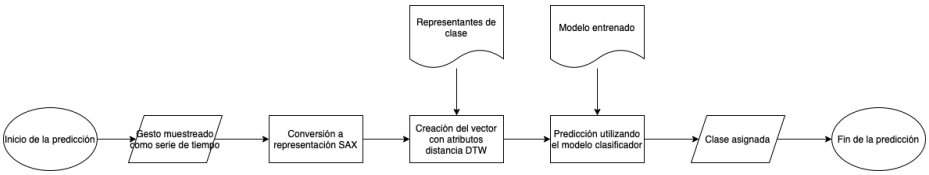

Fig. 10. Diagrama del algoritmo del proceso de predicción para el modelo Atributos SAX-DTW.

6. Resultados experimentales

El proceso experimental consiste en la replicación de los modelos descritos teóricamente en la sección previa, exceptuando el método de vectores de atributos distancia DTW, aplicando inicialmente la técnica de optimización para el tamaño de la ventana w para cada uno de los modelos y posteriormente midiendo la precisión del modelo utilizando el óptimo valor de w para el entrenamiento y fase de predicción con el conjunto de datos completo *GesturePebbleZ1*.

Se decidió por Python como el lenguaje de programación utilizado para la implementación y prueba de los modelos por su gran popularidad, sencillez de implementación y la gran disponibilidad de bibliotecas relacionadas con modelos de *Machine Learning* y especializadas en la Clasificación de Series de Tiempo. Aún con la variedad de bibliotecas existente la gran mayoría de algoritmos descritos fueron desarrollados, tomando únicamente como referencia los existentes en bibliotecas.

De los algoritmos utilizados en el trabajo, aquellos que son implementados desde bibliotecas y no se desarrollaron directamente, son el algoritmo para el muestreo de las señales en el proceso de optimización de la ventana DTW con la biblioteca *resampy*, y la función DBA la cual forma parte de la biblioteca nombrada *tslearn*. Igualmente a lo que refiere a los modelos clasificadores se implementaron funciones parte de las bibliotecas desarrolladas encargadas de ejecutar el sencillo algoritmo 1-NN con las variaciones necesarias para la adición de las diferentes funciones de medición con el modelo referencia y el modelo propuesto en el trabajo de reconocimiento de gestos utilizando dispositivos comunes[13]. La excepción se encuentra en el modelo SVC, para el cual se utilizó la función que ofrece scikit-learn y que requiere únicamente el ingreso de los conjuntos de entrenamiento, conjunto de clases y especificación de parámetros en la fase de entrenamiento, finalizando con la etapa de predicción donde es necesario alimentar el vector por predecir en el modelo entrenado.

Tabla 1. Tabla que muestra los resultados obtenidos para cada método con la técnica de optimización del tamaño de ventana w.

Optimización del tamaño de ventana w .

Métodos	Precisión	Tiempo(min)	Valor w
DTW 1-NN	85.83 %	25.28	**16**
SAX-DTW	89.83 %	6.28	**2**
Atributos SAX-DTW	**95.25** %	43.45	**4**

En este trabajo aunque se aborda teóricamente el modelo de del trabajo[22], no se reporta experimentalmente en esta sección. La razón principal de esta decisión es la inconsistencia mostrada en la implementación lograda, y que otorga resultados de entre 33.33 % y 50 % de precisión, muy por debajo del margen mínimo que se alcanza con los otros métodos.

Las respectivas implementaciones del algoritmo DTW, como función de medida o parámetro de caracterización, son evaluados previo aprendizaje del valor óptimo de la ventana para el contexto específico de cada modelo, tal como expone el artículo[17] no existe un valor único de w intercambiable entre contextos, con factores que afectan a su universalidad como la forma de los datos y el tamaño del conjunto.

Esta optimización del tamaño de ventana se realizó utilizando el conjunto de datos *GesturePeeble*, del que también se dispuso durante el entrenamiento y predicción final de los 3 modelos. Es requisito para resultados fidedignos utilizar un conjunto de datos estratificado, razón por la que el conjunto pasó de 132 entradas para 6 etiquetas distintas, a un total de 120 entradas con 20 secuencias temporales por cada clase. Los demás parámetros que se consideran comunes para todos los modelos con el número de iteraciones N igual a 10 y *k-Fold Cross Validation* igual a 10.

Los límites inferiores y superiores del cálculo de w si cambian en función del modelo y se definieron después de realizarse una iteración de prueba con un limite superior relativamente alto para identificar el rango de valores donde el modelo aumenta su precisión, evitando así el cálculo ocioso de un dominio exagerado de valores w. Con la intención de permitir la replicación de los resultados, se reportan los límites utilizados en la función que calcula el valor óptimo de la ventana DTW como: entre $[11, 17]$ para DTW 1-NN, entre $[2, 8]$ para SAX-DTW y finalmente entre $[2, 10]$ para vectores de atributos distancia SAX-DTW.

En el Cuadro 1 se reportan los resultados obtenidos de la técnica de optimización del valor de la ventana w, siendo importante rescatar el reducido tamaño obtenido para la ventana en el par de métodos que incluyen una transformación de espacio a un dominio simbólico mediante SAX. La sola reducción del valor de w ya es favorable en una evaluación que traduzca el número de operaciones en relación al tamaño de su ventana para series de tiempo hipotéticamente de longitud similar, y más sin embargo la situación es distinta

Tabla 2. Valores de los parámetros utilizados para la evaluación del desempeño de los diferentes métodos.

Parámetros	**Valores**
Tamaño *dataset* entrenamiento	120
Tamaño *dataset* prueba	150
Número de corridas	10
Tamaño de ventana w	Aprendido para cada método

Tabla 3. Resultados promedio obtenidos en la precisión de la predicción para cada método.

Precisión en la predicción		
Métodos	Precisión	Tiempo(seg)
DTW 1-NN	75.20 %	57.91
SAX-DTW	90.33 %	12.89
Atributos SAX-DTW	**94.06** %	**5.181**

al problema real que se estudia, pero afortunadamente para la causa, esto resulta aún más beneficioso, permitiendo inferir según lo reportado una relación en donde una longitud menor y un dominio más pequeño le corresponden un valor de tamaño de ventana menor y por lo tanto menos operaciones por realizarse.

Una vez se conoce el parámetro de ventana del algoritmo DTW, los modelos se encuentran en condiciones para realizar la evaluación final de precisión en la predicción, etapa donde ambos conjuntos de datos se utilizaron en su respectivo contexto: *GesturePebbleZ1_TRAIN* y *GesturePebbleZ1_TEST*. Como se mencionó anteriormente, el conjunto de entrenamiento cuenta con 120 instancias, mientras que la prueba de precisión en predicción se realiza para las 150 instancias del conjunto de prueba. Los datos que se reportan se realizaron fijando los parámetros como se muestra a continuación(Cuadro 2):

Con las cantidades resultantes de las evaluaciones en precisión durante la predicción, se nota una clara ventaja de los modelos basados en características frente al modelo de referencia. Se destaca el éxito del desempeño alcanzado con el nuevo modelo propuesto, siendo notorio en la métrica de precisión al adelantar a SAX-DTW[13] por 3.73 puntos porcentuales; Así como sucede también en el tiempo de predicción, que contrario a las tendencias durante el entrenamiento, se muestra menor en hasta 7.7 segundos o más del doble de rapidez en la predicción.

7. Conclusión y trabajo futuro

Reflexionando sobre los resultados obtenidos en el trabajo, se cumplen las afirmaciones de los artículos en el estado de arte que indican la efectividad y utilidad de algoritmos que propiciaron la simplificación de la complejidad mediante una transformación espacial como SAX, especialmente cuando se comparó con el método de referencia, llegando a ser tan beneficioso que mantiene un valor de tamaño de ventana debajo de 5 y además aporta mayor precisión en

la predicción. El contraste en la complejidad se acentúa con el valor de $w = 16$ cuando se toma en cuenta que la longitud potencial de las series de tiempo que evaluará el algoritmo referencia son de hasta 455 mediciones, mientras que para los métodos SAX-DTW y Atributos SAX-DTW son constantes en 32 y 64, respectivamente.

Se observa para el par de métodos que incluyen la transformación espacial y reducción dimensional de SAX, un aumento de la precisión para la tarea de predicción y una disminución de la complejidad en comparación con el método referencia, por lo que el tiempo de clasificación para obtener la clase de un nuevo gesto también se minimiza. Ayudado de una correcta optimización del algoritmo DTW, las propiedades combinadas de los algoritmos en estos métodos son adecuados para las capacidades de hardware y requerimientos de la propuesta de herramienta embebida.

Por su parte, el modelo de vectores de atributos distancia DTW[22] aún cuando es posible su replicación e implementación experimental, hubiera resultado más costoso que las alternativas evaluadas. Su mayor inconveniente para las restricciones existentes en este proyecto, es la generación del vector de atributos, obligando el cálculo de la similitud DTW para cada una de las instancias de entrenamiento con la serie de tiempo a predecir. Esta metodología podría ser viable cuando se aplica en problemas donde el conjunto de datos de entrenamiento es pequeño y el tiempo de predicción no es crucial.

En términos generales los resultados obtenidos son suficientes para cumplir el objetivo y expectativas esperadas del trabajo con su propuesta Atributos SAX-DTW. El nuevo modelo aprovecha las ventajas que ofrecen cada algoritmo que lo componen y las integra exitosamente bajo el marco metodológico planteado en el trabajo, dotándolo de las capacidades necesarias para superar al estado del arte y presentarse como una alternativa excelente para aplicaciones con restricciones de hardware y rendimiento. Se destaca principalmente su baja complejidad temporal, su disminuido tiempo de cálculo y las modestas exigencias en recursos computacionales, sobre todo para la etapa de predicción.

Retomando las condiciones más generales de la tarea de reconocimiento de gestos, es de notarse que aunque el porcentaje de predicción para ambos modelos es aceptable, e incluso bueno, no se pueden considerar competitivos partiendo desde los resultados reportados en el propio trabajo de Mezari y Maglogiannis[13]. Los modestos resultados obtenidos, se atribuyen a la naturaleza del conjunto de datos con los que se realizó la evaluación, conjunto que incluye únicamente las mediciones de los gestos para el eje espacial Z. El trabajar con series de tiempo multivariables en este tipo de problemas es propicio para conseguir una alta precisión de predicción si los patrones entre ellos son los suficientemente distintos, y esto se deriva de una mayor presencia de rasgos discriminatorios que el clasificador puede aprovechar para definir los límites entre clases.

Con esto en mente se plantea un trabajo a futuro: La evaluación de los 3 modelos bajo los mismos parámetros, o similares, y la misma metodología, utilizando un conjunto de datos propio con instancias de gestos completamente

representadas en los 3 ejes espaciales. El rendimiento obtenido durante este proyecto futuro, permitirá concluir la competitividad del modelo propuesto Atributos SAX-DTW.

El resultado del trabajo a futuro, una vez concluido, se anexará al repositorio público del artículo alojado en la siguiente dirección de GitHub[4].

Referencias

1. National Institute on Deafness and Other Communication Disorders: La afasia. (2017) [En línea]. Disponible en https://www.nidcd.nih.gov/es/espanol/afasia.
2. Instituto Nacional de Trastornos Neurológicos y Accidentes Cerebrovasculares: Apraxia. (2022) [En línea]. Disponible en https://espanol.ninds.nih.gov/es/trastornos/apraxia.
3. American Speech Language Hearing Association: La Disartria. (2021) [En línea]. Disponible en https://www.asha.org/public/speech/Spanish/La-Disartria/.
4. Ismail Fawaz, H., Forestier, G., Weber, J. et al.: Deep learning for time series classification: a review. Data Min Knowl Disc 33, 917–963 (2019). https://doi.org/10.1007/s10618-019-00619-1.
5. D. Peña, G. C. Tiao, R. S. Tsay: Introduction. In: A Course in Time Serie Analysis. New York: J. Wiley (2001)
6. Lines, J., Bagnall, A.: Time series classification with ensembles of elastic distance measures. Data Min Knowl Disc 29, 565–592 (2015). https://doi.org/10.1007/s10618-014-0361-2
7. Hoang Anh Dau, Eamonn Keogh, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, Yanping Chen, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, Gustavo Batista: Hexagon-ML. The UCR Time Series Classification Archive. (2019) https://www.cs.ucr.edu/ eamonn/time_series_data_2018/
8. A. Bagnall, J. Lines, J. Hills, A. Bostrom: Time-Series Classification with COTE: The Collective of Transformation-Based Ensembles. In: IEEE Transactions on Knowledge and Data Engineering, vol. 27, no. 9, pp. 2522–2535 (2015) doi: 10.1109/TKDE.2015.2416723.
9. C. J .G. Ayala Aburto, "Guante traductor de señas para sordomudos,"Tésis título licenciatura, ESIME, unidad Azcapotzalco,. Ciudad de México, México, 2018.
10. E. D. Jiménez Carbajal, G. E. Rivera Taboada, "Sistema de comunicación auditiva para personas con problemas del habla", Tésis para título de licenciatura, ESCOM, Ciudad de México, México, 2013.
11. D. Vishal, H. M. Aishwarya, K. Nishkala, B. T. Royan and T. K. Ramesh, "Sign Language to Speech Conversion,"(en inglés) 2017 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC), 2017, pp. 1-4, doi: 10.1109/ICCIC.2017.8523832.
12. M. M. Chandra, S. Rajkumar and L. S. Kumar, "Sign Languages to Speech Conversion Prototype using the SVM Classifier,"TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON), 2019, pp. 1803-1807, doi: 10.1109/TENCON.2019.8929356.
13. Mezari, Antigoni & Maglogiannis, Ilias. (2017). Gesture recognition using symbolic aggregate approximation and dynamic time warping on motion data. 342-347. 10.1145/3154862.3154927.

[4] https://github.com/LaloValle/SAX-DTW-Features

14. M Müller. "Dynamic Time Warping," *Information Retrieval for Music and Motion*, 4. Alemania: Springer, 2007, pp. 69-73.
15. Sakoe, H., & Chiba, S. (1978). Dynamic programming algorithm optimization for spoken word recognition. IEEE Transactions on Acoustics, Speech, and Signal Processing, 26(1), 43–49. doi:10.1109/tassp.1978.1163055.
16. Tan, Chang Wei & Herrmann, Matthieu & Forestier, Germain & Webb, Geoffrey & Petitjean, François. (2018). Efficient search of the best warping window for Dynamic Time Warping.
17. Dau, H.A., Silva, D.F., Petitjean, F. et al. Optimizing dynamic time warping's window width for time series data mining applications. Data Min Knowl Disc 32, 1074–1120 (2018). https://doi.org/10.1007/s10618-018-0565-y.
18. Krishnamoorthy, V., 2022. Piecewise Aggregate Approximation. [online] Vigne.sh. Available at: ¡https://vigne.sh/posts/piecewise-aggregate-approx/¿[Accessed 10 May 2022].
19. Krishnamoorthy, V., 2022. Symbolic Aggregate Approximation. [online] Vigne.sh. Available at: ¡https://vigne.sh/posts/symbolic-aggregate-approximation/¿[Accessed 10 May 2022].
20. Petitjean, François & Ketterlin, Alain & Gancarski, Pierre. (2011). A global averaging method for dynamic time warping, with applications to clustering. Pattern Recognition. 44. 678-. 10.1016/j.patcog.2010.09.013.
21. Minnaar, A., 2022. Time Series Classification and Clustering with Python –. [online] AlexMinnar. Available at: ¡http://alexminnaar.com/2014/04/16/Time-Series- Classification-and-Clustering -with-Python.html¿[Accessed 10 May 2022].
22. Kate, R.J. Using dynamic time warping distances as features for improved time series classification. Data Min Knowl Disc 30, 283–312 (2016). https://doi.org/10.1007/s10618-015-0418-x

ISSN 1870-4069

Uso de redes neuronales convolucionales en YOLO para la detección de robos armados en establecimientos de ventas

Mario Alberto Muro-Barraza, Aldonso Becerra-Sánchez, Gustavo Zepeda-Valles, Santiago Esparza-Guerrero, Nancy Delgado-Salazar

Universidad Autónoma de Zacatecas, México

sir_mario97@hotmail.com, {a7donso, gzepeda, chago, nancydesal}@uaz.edu.mx

Resumen. El sector comercial es uno de los más afectados por la inseguridad en muchos países, ocasionado principalmente por el robo armado, el cual va creciendo a un ritmo acelerado. El objetivo de este trabajo es diseñar una arquitectura lógica para una herramienta de software capaz de apoyar en la detección de robos cometidos usando armas de fuego al interior de establecimientos de ventas, esto a través del análisis de imágenes. El diseño generado se basa en el empleo de redes neuronales convolucionales, usando en ello el esquema preentrenado "YOLOv4", el cual funciona sobre el "Darknet framework", mismo que utiliza las bibliotecas "OpenCV" para operar. Los resultados obtenidos permitieron la creación de una arquitectura de software detectora de objetos capaz de identificar armas de tres categorías: pistolas, rifles y armas punzo cortantes (tales como cuchillos, cúteres, entre otros), obteniendo un índice de precisión del 75 %; además de lograr una velocidad de 52.63 FPS, la cual es una velocidad en solicitud-respuesta con mejoras a lo existente.

Palabras clave: CNN, NN, visión computacional, detector de objetos, detección de robos armados.

Use of Convolutional Neural Networks in YOLO for the Detection of Armed Robberies in Retail Establishments

Abstract. The commercial sector is one of the most affected by insecurity in many countries, mainly due to armed robbery, which continues to grow at an accelerated pace. The objective of this work is to design a logical architecture for a software tool capable of supporting the detection of robberies committed with firearms inside retail

establishments through image analysis. The proposed design is based on the use of convolutional neural networks, employing the pre-trained "YOLOv4" scheme, which operates on the "Darknet framework" and utilizes the "OpenCV" libraries. The results obtained allowed the development of an object detection software architecture capable of identifying weapons in three categories: handguns, rifles, and sharp weapons (such as knives, box cutters, among others), achieving a precision rate of 75 %. Additionally, a speed of 52.63 FPS was obtained, which represents an improved request-response speed compared to existing solutions.

Keywords: CNN, NN, computer vision, object detector, armed robbery detection.

1. Introducción

La delincuencia evoluciona al mismo ritmo al que lo hace la sociedad [1]. Cada día nuevos métodos son desarrollados por la delincuencia para cometer asaltos, asesinatos o fraudes financieros; y ante este hecho, los gobiernos hacen, aparentemente, todo lo posible para detener y prevenir este tipo de actos delictivos, especialmente aquellos realizados con armas de fuego. No hay región en el mundo que esté exenta de las dramáticas consecuencias generadas por la violencia usando las armas de fuego [2].

Si bien el número de muertos en el contexto de los conflictos armados es bien conocido, menos evidente pero aún más dramático es el hecho de que se pierden más vidas en todo el mundo por eventos con armas de fuego fuera de los conflictos bélicos. The Global Economy [3] recopiló datos de las publicaciones de la Oficina de las Naciones Unidas contra la Droga y el Delito (UNODC), los cuales muestran las tasas y clasificaciones de los países en diferentes aspectos de la seguridad (ver Fig. 1). Se observa que los 20 países con mayor índice de homicidios en el mundo (por cada 100 mil habitantes) están conformados por áreas sudamericanas que no están en guerra.

La baja factibilidad e ineficiencia de los métodos actualmente utilizados para evitar y detener los robos a mano armada provocan pérdidas tanto materiales, económicas y humanas en la mayoría de los establecimientos de ventas, desde los pequeños y medianos, hasta incluso los grandes. Los "mejores" mecanismos de seguridad son demasiado costosos, además de que no son perfectos, principalmente debido al error humano, y los más económicos son aún menos viables por la misma razón [4].

Por otro lado, los últimos avances de tecnología tienen un gran potencial en cuanto a aplicativos nos referimos, y terminarían siendo bastante útiles; lamentablemente, solo se implementan en unos cuantos aspectos de nuestra vida [5]. En pocas palabras, se puede concluir que los ataques con armas de fuego se han vuelto (y seguirán haciéndolo) muy comunes, lo que representa un gran problema social [6]. Todas estas situaciones mencionadas dejan en claro la urgencia de un mecanismo tecnológico novedoso capaz de ayudar en la detección

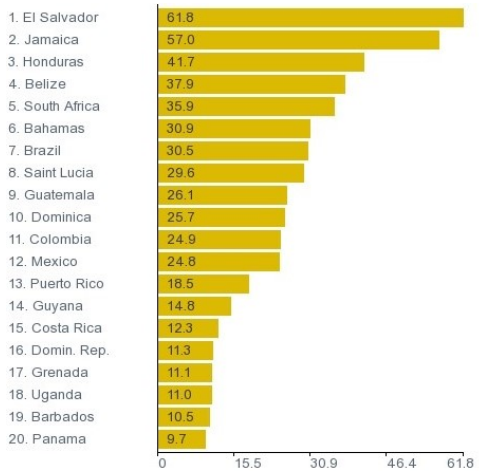

Fig. 1. Países con mayor índice de homicidios (2017).

de estos delitos sin necesidad de invertir una gran cantidad de recurso económico, principalmente orientado para pequeños o medianos establecimientos de ventas.

Las últimas afirmaciones conducen a desarrollar una aplicación de software que utilice algoritmos basados en redes neuronales convolucionales (en lo sucesivo CNN, convolutional neural networks), mediante "YOLOv4" y "Darknet framework" con propósitos de apoyar en la detección de robos a mano armada en establecimientos de ventas, lo anterior a través del análisis de imágenes (pudiendo conseguir igualmente de video). La meta a lograr es apoyar en el proceso de seguridad en establecimientos comerciales, esto mediante una sistema de alerta que notifique cuando se detecta una arma de fuego, avisando al personal de seguridad del lugar, el cual puede tomar acciones al respecto, como llamar al cuerpo policial o actuar como miembro de él. Donde los principales resultados obtenidos (detectando pistolas, rifles y armas punzo cortantes) arrojan un índice de precisión del 75 %; además de lograr una velocidad de 52.63 FPS (frames per second). Es de notar que uno de los puntos importantes a considerar en el diseño es agilizar los tiempos de respuesta, sacrificando un poco las tasas de eficiencia en ello, esto debido a la prontitud de respuesta que un sistema de este tipo debe tener en situaciones de seguridad. Este tipo de sistema podría usarse como una opción de seguridad debido al análisis rápido, la alta eficiencia y el bajo costo que brinda la CNN [7,8]. Asimismo, si se cuenta con personal de vigilancia, el sistema podría ayudar a incrementar la velocidad de detección, ayudando a señalar a través de imágenes que contengan alertas, actos sospechosos que probablemente sean ataques con armas de fuego. Esto ayudaría a reducir el impacto de las pérdidas monetarias y aumentaría la seguridad del personal del establecimiento al denunciar el delito más rápido para poder detenerlo.

El resto del documento está organizado como sigue. En la sección 2 se describen algunos trabajos relacionados relevantes, en la sección 3 se detalla el

esquema propuesto, en la sección 4 se discuten los resultados obtenidos, mientras que en la sección 5 se incluyen las conclusiones y trabajo futuro.

2. Trabajos relacionados

Existen varios trabajos e implementaciones que han sido desarrollados en la línea de la detección de intentos de robo, mientras que otros en la detección del uso de armas como cuchillos y pistolas. Estos desarrollos adoptaron distintos enfoques de la IA como la visión computacional (VC), el deep learning (DL) o lógica difusa. Por ejemplo, Morales et al. [9] crearon un sistema para apoyar en detectar robos violentos efectuados con armas de fuego o cuchillos. Utilizaron un extractor de características (CNN) llamado VGG-16, el cual se encuentra ya pre-entrenado; obteniendo una precisión del 96.69 %. Al mismo tiempo, ellos también desarrollaron un conjunto especializado de videos llamado 'UNI-Crime Dataset' [10], el cual contiene datos de robo y no robo. Este conjunto de datos es un subconjunto del original creado por Sultani, Chen y Shan [11], el cual contiene videos de CCTV (circuito cerrado de televisión, usados para la videovigilancia) de comportamientos normales, robos, peleas y explosiones, entre muchos otros. En este sentido, Vajhala et al. [12] desarrollaron un sistema para detectar personas con cuchillos o pistolas a partir de fotogramas obtenidos de un video de CCTV. Utilizaron el histograma de gradientes orientados (HOG) como vector de características y el algoritmo de propagación inversa de CNN para la clasificación. Sus rendimientos obtenidos ascienden a un índice de precisión del 83.0 %, mientras que en los tiempos de respuesta del sistema no se obtuvieron tiempos de respuesta tan rápido debido a sus dos etapas de procesamiento (verificación e identificación). Para lograr el cometido, se utilizó la base de datos INRAI [13] en el proceso de entrenar la fase de la detección humana.

En contraparte a los trabajos anteriores, Navalgund y Priyadharshini [14] crearon un sistema con la capacidad de detectar cuchillos y pistolas en la mano de una persona apuntando a otra. Ellos compararon el modelo pre-entrenado VGGNET19 con el modelo GoogleNet Inception V3, concluyendo que VGG19 obtuvo mejores resultados en la precisión del entrenamiento en menos tiempo de procesamiento (tasa de precisión del 92 % frente a un 69 % respectivamente). Bajo la misma idea, Romero y Salamea [15] propusieron un sistema basado en VGG Net utilizando imágenes en escala de grises como entradas para múltiples arquitecturas de CNN. En sus resultados obtuvieron un 86 % de precisión y un 86 % de exhaustividad. Mientras que Yahya y Ullah [16] presentaron una arquitectura de red neuronal profunda de tres flujos para la clasificación y localización temporal de eventos de robo en videos de CCTV. Con esto, obtuvieron una tasa de precisión del 75 % y una tasa de precisión de localización temporal del 73.89 %. A diferencia de otros trabajos, este sistema puede localizar el videoclip que contiene el hecho del robo.

Adicionalmente, Grega et al. [17] desarrollaron un algoritmo capaz de detectar a una persona que lleva un arma de fuego al descubierto. El algoritmo se basa en una red neuronal convencional (NN) y un descriptor MPEG-7 para

clasificar transmisiones de video de un CCTV. Los resultados obtenidos se dividen en dos categorías, con una precisión del 95.58 % para la transmisión que contenía un arma, y un 99.32 % para la que no la contenía. En pruebas posteriores obtuvieron una especificidad del 100 % para una transmisión sin arma y del 96.69 % para una que sí la contiene. Algunos años más tarde, Grega et al. [18] propusieron un nuevo enfoque para el mismo problema de este trabajo, que incluía muchas implementaciones arquitectónicas nuevas, aunque sin resultados tan diferentes. Por otra parte, Tiwari y Verma [19] presentaron y desarrollaron un marco que utiliza la segmentación basada en el color para eliminar objetos no relacionados de una imagen utilizando el algoritmo de agrupamiento k-means. Implementaron dos herramientas, la primera se denomina Harris Interest Point, mientras que la segunda es llamada Fast Retina Keypoint (FREAK), las cuales se usaron para la detección de armas, obteniendo una precisión global del sistema del 84.26 %.

3. Esquema propuesto

Para el desarrollo se siguió la metodología ágil conocida como *Prototyping*, la cual se define como un método de desarrollo de sistemas de software en el que se construye, prueba y luego se re-inventa un prototipo según sea necesario [20]. El proceso continúa hasta que finalmente se logra una versión aceptable a partir del cual se puede desarrollar el sistema o producto completo [21]. El enfoque del *Prototyping* sigue un proceso que se repite en cada iteración [22], [23] (ver Fig. 2). Durante la definición de los objetivos del prototipo (primera etapa), se busca seleccionar una de las varias funcionalidades a desarrollar para que, a continuación, se establezca hasta dónde abarcará dicha funcionalidad (segunda etapa), de tal manera que, una vez desarrollada (tercera etapa), exista concordancia entre los resultados y las métricas usadas para su evaluación (última etapa).

3.1. Definición de objetivos

El sistema plantea una meta de apoyo en detección de sucesos ilícitos como asaltos a mano armada, para ello usa imágenes obtenidas de una transmisión de video de cámaras instaladas en establecimientos comerciales o de ventas, de tal manera que puede detectar armas en escenario equiparables a tiempo real. Este proceso implica avisar a un miembro del cuerpo de seguridad del establecimiento sobre sucesos sospechosos, tomando acciones al respecto. Al seguir la metodología de *Prototyping* se plantearon los siguientes puntos para crear un producto que pueda usarse en escenarios reales; además, las métricas aumentarán a medida que mejore el prototipo: i) el sistema debe aceptar imágenes como entrada provenientes de cámaras de establecimientos de ventas o comerciales; ii) el sistema debe detectar a una velocidad considerablemente rápida la presencia de armas en la imagen analizada; iii) el sistema debe mostrar el resultado del análisis realizado, priorizando más la prontitud de respuesta que la eficiencia total; iv)

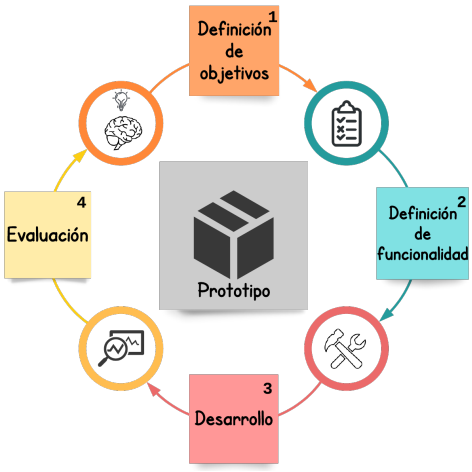

Fig. 2. Ciclo de vida del *Prototyping*.

si se detecta un arma, se debe mostrar una alerta visual de dónde se localiza; y v) el sistema consta con las bases para implementarse en un escenario real.

3.2. Definición de funcionalidad

El sistema está conformado de 3 módulos: i) receptor de imágenes, el cual conecta el prototipo a las imágenes, revisándolas y verificando formatos para pasarlas al módulo de IA; ii) agente de inteligencia artificial, el cual hace la detección, identificando en cada en imagen un objeto de tipo arma como elemento potencial; iii) transmisor de imágenes, el cual muestra la alerta, transmitiendo la cadena de imágenes, incluyendo una alerta visual que localiza el arma.

La distribución física se puede ver en la Fig. 3, donde se muestra la estructura del sistema propuesto. Dicha figura muestra cómo se deben conectar las cámaras al receptor de imágenes, el cual obtendrá la imagen o cadena de imágenes y las pasará en un formato correcto al agente de IA, quien analizará cada imagen para detectar la presencia de armas. Finalmente, el resultado será enviado al monitor de seguridad (para mostrar alertas visuales) y al disco duro (para ser guardado). Los últimos pasos son realizados por el transmisor de imágenes.

3.3. Desarrollo

Cada uno de los módulos de que consta el prototipo (Fig. 3) fue desarrollado para trabajar en un entorno de escritorio. La aplicación se ejecuta en una máquina local, donde las interfaces (tanto del receptor como del transmisor de imagen) se crearon utilizando Python y OpenCV.

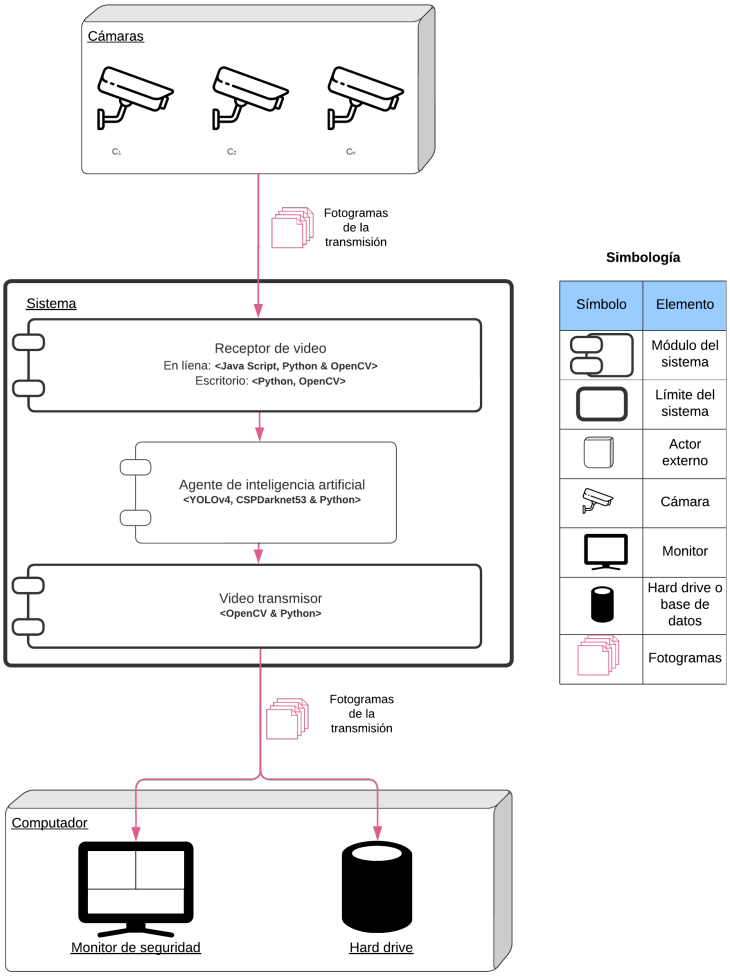

Fig. 3. Distribución de los módulos del prototipo.

Descripción del dataset Todas las imágenes utilizadas se obtuvieron a partir de dos fuentes diferentes, la primera fue de Open Images V6 (OIV6) [24]; este dataset consta de 9 millones de imágenes etiquetadas, además cuentan con sus cuadros delimitadores y máscaras de segmentación de objetos, relaciones visuales y narraciones localizadas. Contiene un total de 16 millones de cuadros delimitadores para 600 clases de objetos. El segundo repositorio fue Kaggle [25], que alberga miles de datasets diferentes, creados por su propia comunidad, en donde cada uno de ellos contiene su propio tipo de etiquetado. Para OIV6 se utilizó un script que recopila automáticamente todas las imágenes que tienen el objeto especificado en el código, siendo los siguientes los seleccionados: pistola, rifle y arma; obteniendo alrededor de 4,000 imágenes.

Fig. 4. Algunos métodos de aumento de datos aplicados en imágenes.

Después de eso se utilizaron funciones de *aumento de datos (DA - data augmentation)*, no solo para incrementar el tamaño del conjunto de datos, sino también la variedad y complejidad (tales como flip, rotation, crop, shear, grayscale, hue, saturation, brightness, exposure, blur, noise, cutout y mosaic) [26]; algunos de ellos se pueden ver en la Fig. 4. DA consiste en crear una copia del dataset original, y después aplicarle múltiples procedimientos de modificación a todos los datos (en este caso imágenes), de tal manera que incrementa el tamaño y variedad del conjunto (ya que ahora el dataset consta de las imágenes originales y las imágenes editadas). Las funciones aplicadas fueron: *'flip'*, que es una función que voltea 180° una imagen. Otra fue *'rotation'*, la cual permite rotar las imágenes en un ángulo personalizado, en este caso de 45°. La función *'crop'* genera una imagen recortada de la original, para este caso se utilizó un 30 %. La función *'shear'* añade una variabilidad en la perspectiva tridimensional de la imagen, que sería como girar la imagen en el eje Y así como en el eje X, usando parámetros como 15 % y 15 % respectivamente. La función *'grayscale'* genera imágenes extras que solo se encuentran en escala de grises, usando en ello un porcentaje del 90 % de escala. La variante de *'hue'* aplica colores distintos a los de la imagen original, al aplicarse se le especificó un parámetro del 51, lo cual genera dos variantes, una positiva del 51° y otra del -51°. La función de *'blur'* genera imágenes con una difusión, al usarse se le especificó 2px. El *'noise'* genera ruido en la imagen, para ello usamos un porcentaje del 5 % [26].

Este procedimiento se realiza también para evitar el sobre-entrenamiento e incrementar la precisión durante la fase de entrenamiento. Una vez que se aplicó el DA, el conjunto de datos incrementó su tamaño a 13,192 imágenes (ver Tablas 1 y 2); se pueden ver en las Fig. 5, 6, 7 y 8 dos ejemplos del dataset al haberse aplicado DA, donde se nota cómo una imagen cambia al aplicarse los procedimientos como la rotación, el voltear, recortar, aumento de ruido entre otras.

Fig. 5. Imagen 1 de ejemplo antes DA.

Fig. 6. Imagen 1 de ejemplo después de DA.

Fig. 7. Imagen 2 de ejemplo antes de DA.

Fig. 8. Imagen 2 de ejemplo después de DA.

Por otro lado, el segundo dataset (Kaggle) fue creado por la comunidad, el cual precisamente usa la estructura y la forma de etiquetado que necesita Darknet y YOLOv4 para administrar las imágenes y sus meta-datos (la clase del objeto en la imagen y sus coordenadas). Una vez que fue procesado usando DA, se obtuvieron 2,158 imágenes. El conjunto de datos descargado de OIV6 se usó

para el entrenamiento y el descargado de Kaggle se usó para la evaluación. Ambos suman un total de 15,350 imágenes, donde el dataset de entrenamiento representa el 85.94 % del total, y el dataset de prueba representa el 14.06 %.
Según se puede apreciar en las Tablas 1 y 2, se le dio más importancia a conseguir un dataset bastante amplio antes de que este estuviera balanceado (en cuanto a las clases contenidas dentro de él). Así pues, la variabilidad de las formas de los objetos, así como la desproporción de cantidades de estos, generó resultados bastante variados por cada una de las versiones del entrenamiento. En la Tabla 1 se aprecia como se busca incrementar el número total de imágenes del dataset (Num). Esto fue con el objetivo de realizar entrenamientos mucho más largos que terminan mejorando el modelo en general. Una vez que se experimentó con la *v_ 3*, se entendió que entre más clases estuvieran contenidas dentro del dataset, menos exacto y preciso sería el modelo de IA. En seguida se procedió a aumentar la cantidad de imágenes en las clases consideradas más importantes, quitando a la vez las de menor relevancia. En este caso, aumentando las clases de las armas, rifles y pistolas; así como de forma contraria, suprimiendo las clases de cuchillos y escopetas.

Una vez descubierto esto esto, se intentó experimentar también con las dimensiones de normalización de las imágenes. Dichas dimensiones, a las cuales YOLOv4 normaliza (Dim_N) para su procesamiento, también impactan en el tiempo de entrenamiento, así como en la precisión que se puede obtener. Ya que al aumentar el tamaño de la imagen, el proceso de convolución termina requiriendo más tiempo y procesamiento, por ende, si aumentamos mucho el tamaño y no damos el tiempo suficiente, termina siendo contraproducente. Por el contrario, al bajar la resolución, el proceso aumenta en velocidad, sin embargo minimiza su precisión debido a que los objetos empiezan a perder forma. En ambas tablas se puede observar que la suma del total de objetos de cada clase (Objs) supera a la cantidad de imágenes, y esto se debe a que varias de las imágenes contienen más de un objeto dentro de ellas.

YOLOv4 YOLO es un modelo detector de objetos para el paradigma CV. De hecho, Bochkovskiy et al. [27] explicaron que el objetivo de YOLOv4 era concebir un sistema detector de objetos de alta velocidad usando una optimización de cálculos paralelos. Es por esto que la estructura base de YOLOv4 es de tipo *detector de una etapa* (*one-stage*). Un modelo de *one-stage* es capaz de detectar objetos sin necesidad de un paso preliminar. Por el contrario, un detector de *dos etapas* utiliza una fase preliminar en donde se detectan regiones de importancia y luego, en la fase final, se analizan dichas regiones en busca del objeto.

Los detectores de una etapa pueden hacer predicciones rápidamente, lo que permite una percepción del proceso en **tiempo real**. Como se puede ver en la Fig. 9, cada parte del modelo es un subsistema que tiene un propósito muy específico. El **backbone** es la parte del detector encargada de obtener las características esenciales de la entrada; normalmente compuesto por una CNN, en este caso CSPDarknet53, la cual está compuesta por 29 capas convolucionales 3×3, un campo receptivo de 725×725 y 27.6 M de parámetros. El papel esencial

Tabla 1. Características de las diferentes versiones del dataset de entrenamiento usado para YOLOv4. Donde: Num es la cantidad total de imágenes del dataset, Objs refiere a la cantidad de objetos que contienen las imágenes (una imagen puede contener una o más armas), Dim_N representa las dimensiones de las imágenes una vez normalizadas, N_A1 representa la cantidad objetos identificados como pistolas, N_A2 es la cantidad de objetos identificados como rifles, N_A3 refiere a la cantidad de objetos etiquetados armas, N_A4 indica la cantidad de objetos marcados como escopetas y N_A5 es la cantidad de objetos marcados como cuchillos.

Versión	**Num**	**Objs**	**Dim_N**	**N_A1**	**N_A2**	**N_A3**	**N_A4**	**N_A5**
v_1	3123	3423	416×416	1737	0	1686	0	0
v_2	1486	1686	512×512	0	0	1686	0	0
v_3	4662	7657	512×512	727	2540	2960	580	850
v_4	4662	7657	512×512	727	2540	2960	0	0
v_5	13192	20964	416×416	2465	8472	10027	0	0
v_6	13192	20964	512×512	2465	8472	10027	0	0

Tabla 2. Características de las diferentes versiones del dataset de pruebas usado para YOLOv4. Donde: Num es la cantidad total de imágenes del dataset, Objs refiere a la cantidad de objetos que contienen las imágenes (una imagen puede contener una o más armas), Dim_N representa las dimensiones de las imágenes una vez normalizadas, N_A1 representa la cantidad objetos identificados como pistolas, N_A2 es la cantidad de objetos identificados como rifles, N_A3 refiere a la cantidad de objetos etiquetados armas, N_A4 indica la cantidad de objetos marcados como escopetas y N_A5 es la cantidad de objetos marcados como cuchillos.

Versión	**Num**	**Objs**	**Dim_N**	**N_A1**	**N_A2**	**N_A3**	**N_A4**	**N_A5**
v_1	241	179	416×416	24	0	155	0	0
v_2	241	179	512×512	24	0	155	0	0
v_3	241	392	512×512	24	110	155	26	77
v_4	188	289	512×512	24	155	0	0	0
v_5	188	289	512×512	24	155	0	0	0
v_6	188	289	512×512	24	155	0	0	0

del **neck** es recopilar mapas de características de diferentes etapas. Por lo general, un neck se compone de varios caminos de abajo hacia arriba y varios caminos de arriba hacia abajo; en este caso se utilizaron SPP (Spatial Pyramid Pooling) y PAN [28] (PaNet o Path Aggregation Network). SPP [29] ayuda a administrar mapas de características de la red troncal y PAN permite una mejor propagación de la información de la capa del backbone de abajo hacia arriba o de arriba hacia abajo. Ambos aumentan la precisión del detector. El rol de **head** (para el caso de un detector de una etapa) es realizar una predicción densa (dense prediction). La **dense prediction** es la predicción final, que se compone de un vector con las coordenadas del cuadro delimitador predicho (centro, alto, ancho),

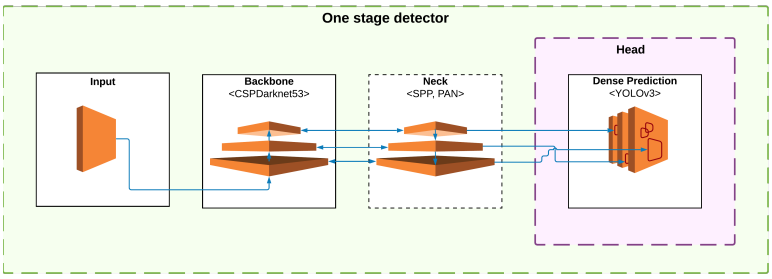

Fig. 9. Estructura de YOLOv4.

la puntuación de confianza de la predicción y la etiqueta; normalmente está conformado por una NN simple, en este caso, la misma usada para YOLOv3 [30].

Receptor de imágenes Este módulo (ver Fig. 3) es el encargado de acceder directamente al directorio donde se encuentran todas las imágenes a analizar; esto usando funciones de Python y OpenCV que automáticamente verifican y decodifican la cadena de estas para finalmente pasarlas al agente de IA.

Agente de inteligencia artificial Los aspectos importantes a mencionar son los parámetros utilizados para el enfoque del proyecto; estos parámetros son las *subdivisiones*, *anchura* - *altura*, *max_ batches* y los *pasos*. Las *subdivisiones* son subgrupos de imágenes tomadas de un conjunto mayor (llamados lotes); los lotes, por otro lado, son un subconjunto de todo el dataset de entrenamiento. Un valor estándar para lotes es de 64 imágenes y, para subdivisiones, se pueden dar una variedad de parámetros completamente válidos; dicho valor dividirá el lote creando los subgrupos de imágenes. Para este prototipo, el valor (para subdivisiones) que dio mejores resultados fue de 32; significa que el lote de 64 imágenes se dividió entre 32, dando como resultado 2 subgrupos de 32 imágenes cada uno. El parámetro *batch* define cuántas imágenes se procesarán por cada iteración de la fase de entrenamiento. Una vez que se procesaron todas las subdivisiones, la iteración finaliza. La *anchura* y la *altura* son parámetros que definen el tamaño que debe tener cada imagen para ser procesada en cada iteración; si la imagen no cumple con eso, debe ser redimensionada. A mayor tamaño, mayor precisión del modelo; por eso la resolución utilizada para la anchura y la altura fue de 512×512 píxeles. El parámetro *max_ batches* indica cuántos lotes se procesarán en toda la fase de entrenamiento; y solo un lote es procesado por una iteración. Esto significa que max_batches indica cuántas iteraciones realizará la fase de entrenamiento. Para este prototipo, el valor de max_batches fue de 6,000. El valor de los *pasos* establece en qué iteración (o número de lote) se cambiará la tasa de aprendizaje; esto se hace multiplicando la tasa de aprendizaje por una escala (esto ayuda a variar la tasa de aprendizaje y obtener mejores resultados). Para el valor de los pasos se recomienda utilizar

el 80 % y el 90 % del total de iteraciones (max_batches), por lo que el valor establecido para la fase de entrenamiento fue de 4,800 y 5,400; eso significa que la tasa de aprendizaje cambiará (se multiplicará por la escala) una vez que las iteraciones alcancen esos números.

Transmisor de imágenes Este módulo (ver Fig. 3) muestra la imagen o imágenes que fueron analizadas por el agente de IA, añadiendo además la alerta sobre el objeto encontrado.

3.4. Evaluación

Los resultados obtenidos (ver Tabla 3) son una señal de un buen progreso. El dataset de prueba incluyó la utilización de 2,158 imágenes, obteniendo resultados interesantes en métricas de precisión, exhaustividad, F1, mAP_{50} (medición de la precisión promedio de identificación) y Average IoU (promedio de la intersección sobre la unión). Estos valores hacen evidente que el modelo no tiene el mejor nivel en cuanto a precisión (cada métrica redondea del 60 % al 70 %); sin embargo, esto no quiere decir que está deficiente, principalmente porque esta investigación y la tecnología utilizada no tienen como objetivo principal alcanzar la mejor precisión máxima, si no el obtener una gran eficiencia en términos de tiempo. Esto se debe a que, si bien los resultados parciales de este trabajo aún no son tan elevados como el estado del arte, muestran la muy alta tasa de análisis de imágenes (19 ms) por cada instancia, lo cual podría proporcionar un sistema que, al ser implementado, proporcione un comportamiento en tiempo real, que de hecho es más útil que un modelo muy preciso pero lento.

Otro gran rasgo acerca de estos hallazgos radica en que están muy cerca de los publicados en la investigación oficial del proyecto YOLOv4 [27], que obtuvo un mAP_{50} de 41.66 % y tiempo por imagen de 19 ms. O incluso más en comparación con el modelo anterior (YOLOv3 [30]), con un mAP_{50} de 55.3 % y un tiempo por imagen de 35 ms (YOLOv4 y YOLOv3 fueron entrenados para el conjunto de datos MS COCO creado por [31]). Aclaremos que la columna de 'Imágenes' se refiere a la cantidad utilizada para este caso, mientras que la columna de 'Tiempo' indica el tiempo utilizado en promedio para analizar cada imagen. Teniendo en cuenta que se utilizó 'Google Colaboratory pro' para la experimentación, el cual según el uso y parámetros aleatorios, puede proveer un equipo conformado por una GPU Tesla V100-SXM2-16GB, 13 GB de RAM, CUDA 11.0, una CPU Xeon Processors @2.3Ghz, todo esto sosteniendo un sistema operativo Linux-5.4.104+-x86_64-with-Ubuntu-18.04-bionic. También puede cambiar la gráfica por una GPU Tesla T4. Además de la variabilidad del equipo, se encontró la limitación en el tiempo de uso, el cual no podía ser mayor de las 24 horas de trabajo continuo.

4. Discusión de resultados

Para obtener los resultados que se muestran en la Tabla 3, se realizaron seis fases de entrenamiento, todas ellas cambiando los parámetros de fine-tuning

Tabla 3. Resultados obtenidos en YOLOv4.

Imágenes	Precisión	Exhaustividad	F1	Average IoU	mAP_{50}	Tiempo (ms)
2158	0.75	0.63	0.68	61.13 %	41.66 %	19

Tabla 4. Parámetros de fine-tuning usados para todas las iteraciones de entrenamiento (versiones desde v1 hasta v6) en YOLOv4.

Batch	Momentum	Learning rate
64	0.949	0.001

Tabla 5. Resultados en las diferentes versiones de entrenamiento usando YOLOv4. Donde A1: pistola, A2: rifle, A3: Arma, A4: escopeta, A5: cuchillo; el símbolo '✓' indica si se utilizó y '×' si no fue utilizado.

Ver	sbdv	A&A	Maxb	Pasos	F1	AvIoU	mAP_{50}	A1	A2	A3	A4	A5	DA
v_1	16	416×416	6,000	4,800-5,400	0.33	26.62 %	49.42 %	✓	×	✓	×	×	×
v_2	16	512×512	10,000	8,000-9,000	0.02	2.06 %	1.52 %	×	×	✓	×	×	×
v_3	16	512×512	10,000	8,000-9,000	0.51	43.50 %	57.57 %	✓	✓	✓	✓	✓	×
v_4	32	512×512	10,000	8,000-9,000	0.48	38.56 %	52.78 %	✓	✓	✓	×	×	×
v_5	32	416×416	6,000	4,800-5,400	0.46	37.08 %	49.40 %	✓	✓	✓	×	×	✓
v_6	32	512×512	6,000	4,800-5,400	0.68	61.13 %	41.66 %	✓	✓	✓	×	×	✓

(quedando como los definitivos los mostrados en la Tabla 4) y variando las clases de armas; además, se buscó la orientación para utilizar y adaptar YOLOv4 [27] como detector de armas de fuego. Las últimas seis versiones previas del agente IA se pueden ver en la Tabla 5, junto con los parámetros del fine-tuning que fueron usados, así como los resultados obtenidos en cada una de ellas. Algunas versiones fueron entrenadas usando más, o incluso menos clases (pistola, rifle, arma, escopeta, cuchillo). El término 'DA' indica si se utilizó data augmentation, el término 'Sbdv' indica las subdivisiones, 'A&A' representa a la anchura y la altura, 'Maxb' refiere al valor usado para max_batches y 'AvIoU' para average IoU.

En la *v_3* se distingue que los resultados podrían verse como los más sobresalientes; pero de hecho, este alto valor se debe a que la clase *cuchillo* obtuvo un mAP_{50} individual de 89.41 %, que incrementa notoriamente el mAP_{50} total del modelo, aún cuando las otras clases no obtuvieron un gran resultado. Tomando esto en cuenta para las siguientes versiones, se establecieron las principales clases de objetos a tener en cuenta; *arma*, *rifle* y *pistola*. Una vez que se desarrolló la *v_5*, se descubrió que se pueden crear mejores modelos a partir del fine-tuning y el uso del DA. La versión *v_6* es la combinación del nuevo dataset usado en la *v_5* y el fine-tuning usado para *v_4*. Es así que la *v_6* se considera como el mejor modelo, dado que las dimensiones de entrada (512×512) permiten detectar objetos en imágenes de gran tamaño, conteniendo más obstrucciones, diferentes formas y posiciones en los objetos, obteniendo mejores resultados en la evaluación.

Los resultados mostrados en la Tabla 3 fueron los obtenidos en la *v_ 6* que se muestra en la Tabla 5. Este modelo obtuvo una precisión con el 75 % de las predicciones de presencia de armas de fuego correctas. Por otro lado, el modelo obtuvo una exhaustividad del 0.63, lo que significa que el agente de IA define correctamente el 63 % de presencia real de armas de fuego. El resultado F1 (0.68) significa que el modelo tiene una eficacia del 68 % en la detección de armas de fuego. El average IoU mostró que solo el 61.13 % de los objetos se superpusieron correctamente dentro del cuadro de predicción. El resultado de mAP_{50} indica que el modelo, para las tres clases, tiene una precisión del 41.66 %. Hay aspectos importantes que se detectaron que pudieron haber decrementado los valores obtenidos para cada métrica, uno de ellos es la integración de la clase 'pistola', pues esta es la que obtiene los peores resultados de las tres, disminuyendo de manera notoria todos los valores de todas las métricas.

Una vez que el entrenamiento del modelo del IA termina, arroja también los datos para armar una matriz de confusión, la cual es utilizada para mostrar cómo es que se está comportando. Si revisamos la Tabla 6 (*v_ 6*) encontramos ciertos datos peculiares. Los *verdaderos positivos* (VP), que aluden a todos los objetos que se clasificaron correctamente, resultan en un total de 1,119; vemos que la clase pistola identificó menos objetos (263 clasificaciones), debido probablemente a la distribución de estos en las diferentes clases, ya que precisamente la pistola es la clase que menos objetos tiene dentro del dataset (ver la Tabla 1). Cosa contraria a lo que pasa en la clase de arma (que identificó 347), siendo la que más objetos tiene dentro del dataset, pero obteniendo un resultado no tan favorable de VP (muy probablemente derivado de la complejidad de las formas de los objetos de esta clase). Lo anterior afecta el desempeño resultante de la *v_ 6*, ya que por un lado la clase pistola obtuvo resultados bajos; mientras que por otro lado, la case de arma consumió gran parte del recurso usado para el entrenamiento, arrojando también resultados bajos. Aún incluso cuando la clase rifle obtiene muy buenos resultados (509 identificaciones), en general el modelo de IA resulta afectado.

Continuando con la matriz, tenemos los *falsos positivos* (FP), que nos indican todas aquellas identificaciones erróneas por clase, que quiere decir que se identificó una clase que no se buscaba. Para el caso del arma, encontramos la mayor cantidad de identificaciones incorrectas, ya que se clasificaron erróneamente 122 objetos como rifle en vez de arma. Por otro lado tenemos a los *falsos negativos* (FN), que refiere a aquellos objetos que estaban presentes pero no fueron identificados, siendo un total de 656. Una curiosidad es la falta de los *verdaderos negativos* (VN), los cuales representan imágenes que no contenían ningún objeto de las tres clases y, de forma correcta, no se identificaron objetos en dichas imágenes. Pero dicha variable no se encuentra en nuestros resultados, ya que el conjunto de imágenes usadas pertenecen al dataset de entrenamiento, en el cual no existen imágenes que no contengan algún objeto. Al no existir imágenes vacías, no puede haber VN.

Cabe mencionar que en el diseño del prototipo se enfatiza la prontitud de respuesta del sistema antepuesto a la eficiencia total, esto debido a que en un

Tabla 6. Matriz de confusión generada de los resultados obtenidos en YOLOv4.

Clase	Pistola	Rifle	Arma
Pistola	263	67	31
Rifle	78	509	49
Arma	36	122	347

sistema de seguridad es de trascendental importancia una notificación inmediata de un suceso sospechoso. De manera adicional, se puede mencionar que existen otras arquitecturas que pueden tomarse como base para experimentación aparte de YOLOv4, como DeepSORT [32] y el prometedor YOLOv5 [33] (usando Pytorch [34]). Los resultados en dichas plataformas fueron muy inferiores a YOLOv4, por lo que aún falta experimentar con ellas para poder aplicarlo en entorno como el del presente trabajo.

5. Conclusiones y trabajo futuro

Este trabajo tuvo como objetivo presentar un esquema de software que permita apoyar en el proceso de detección de armas de fuego en asaltos en establecimientos comerciales, enfatizando en ello siempre una pronta respuesta por sobre una eficiencia total. Los resultados obtenidos demuestran (75 % de precisión con 52.63 FPS de velocidad) que el sistema podría ser utilizado en un entorno real, generando además un comportamiento en tiempo real. La estructura propuesta para todo el sistema permite la mejora para futuras iteraciones, ya que es un sistema modular que se puede adaptar a múltiples entornos.

Se empleó YOLOv4 como agente de IA, esto dado que proporciona una gran velocidad y precisión, atributos que constituyen la definición de alto 'rendimiento', el cual se busca atacar como requerimiento no funcional. Tomando en cuenta la velocidad y la taza de precisión, se hacen evidentes las ventajas que se podrían obtener automatizando las actividades del día a día mediante el uso de detectores de objetos en tiempo real. Otro aspecto importante radica en que los resultados obtenidos (precisión del 75 %, exhaustividad del 63 %, F1 con un 68 %, average IoU del 61.13 %, mAP_{50} del 41.66 %; incluyendo un 52.63 de FPS) son un base sólida para mejoras en el desarrollo de detectores de armas de fuego en tiempo real, coadyuvando en el desarrollo de metodologías precisas de *videovigilancia automatizada*. Es así que los nuevos mecanismos de IA pueden contribuir en garantizar el bienestar y la seguridad de las personas.

Para trabajo futuro, hay dos puntos a mencionar que se plantean como futuras iteraciones de acuerdo a Prototyping. Uno implica la arquitectura de agente de IA para producir una mayor precisión (Darknet framework para YOLOv4), la cual presenta algunas restricciones; donde la principal se basa en que su desarrollo está destinado a ser utilizado en la detección de objetos distintos a los propuestos en este trabajo. Es importante considerar el desarrollo de una nueva CNN en lugar de utilizar algunas de las ya desarrolladas por otros trabajos,

verificando otros parámetros para producir un mejor ajuste (fine-tuning) y crear un mejor dataset que podría mejorar mucho las métricas. El segundo punto involucra el hardware empleado, el cual produce limitaciones, principalmente porque la fase de entrenamiento es muy exigente con la GPU (Graphics Processing Unit). Teniendo en cuenta que se utilizó 'Google Colaboratory pro', se encontró la limitación en el tiempo de uso, el cual no podía cambiarse. En el futuro se podrían utilizar componentes que permitan utilizar toda la capacidad de los mismos sin ninguna restricción de tiempo.

Referencias

1. Herrera-Rodríguez, J., Vega-Zayas, J.M., Rodríguez-González, J.A.: Estrategias de afrontamiento a la criminalidad por microcomerciantes de León [Guanajuato (México)] ¿Indicador de cohesión o falla de la política criminal?. Derecho y Cambio Social (053), 1–18 (2018)
2. United Nations Office on Drugs and Crime: Firearms protocol, https://www.unodc.org/unodc/en/firearms-protocol/index.html. Last accessed 16 Feb 2020
3. The Global Economy: Homicides rate rankings, https://www.theglobaleconomy.com/rankings/homicide_rate. Last accessed 6 Mar 2020
4. Kleck, G., DeLone, M.A.: Victim resistance and offender weapon effects in robbery. Journal of Quantitative Criminology **9**(1), 55–81 (1993)
5. Xu, H.: Application of cloud computing information processing system in network education. In: Abawajy, J.H., Choo, K.K.R., Islam, R., Xu, Z., Atiquzzaman, M. (eds.) International Conference on Applications and Techniques in Cyber Intelligence ATCI 2019, vol 1017, pp. 1809-–1815. Springer International Publishing, Huainan (2020). https://doi.org/10.1007/978-3-030-25128-4_238
6. The Global Economy: Robbery rate rankings, https://www.theglobaleconomy.com/rankings/robery. Last accessed 16 Feb 2020
7. González-Arriaga, D.M., Vargas-Treviño, M.A.D., Guerrero-García, J., López-Gómez, J.: Development of a Platform for Generation of CNN and Multilayer Neural Networks. Computación y Sistemas **26**(1), 172–182 (2022)
8. Alquisiris-Quecha, O., Martinez-Carranza, J.: Depth Estimation Using Optical Flow and CNN for the NAO Robot. Research in Computing Science **148**(11), 49-58 (2019)
9. Morales, G., Salazar-Reque, I., Telles, J., Díaz, D.: Detecting Violent Robberies in CCTV Videos Using Deep Learning. In: MacIntyre, J., Maglogiannis, I., Iliadis, L., Pimenidis, E. (eds.) Artificial Intelligence Applications and Innovations AIAI 2019, IFIP Advances in Information and Communication Technology, vol. 559, pp. 282–291. Springer, Greece (2019). https://doi.org/10.1007/978-3-030-19823-7_23
10. UNI-Crime Dataset, URL: https://didt.inictel-uni.edu.pe/dataset/. Last accessed 30 Jun 2020
11. Sultani, W., Chen, C., Shah, M.: Real-world anomaly detection in surveillance videos. arXiv, 1–10 (2018)
12. Vajhala, R., Maddineni, R., Yeruva, P.R.: Weapon Detection In Surveillance Camera Images. Ph.D. thesis, Blekinge Institute of Technology (2016).

http://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1054902&dswid=-2377
13. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), pp. 886-–893. IEEE, California (2005)
14. Navalgund, U.V., Priyadharshini, P.K.: Crime Intention Detection System Using Deep Learning. In: 2018 International Conference on Circuits and Systems in Digital Enterprise Technology, ICCSDET, pp. 1–6. IEEE, India (2018)
15. Romero, D., Salamea, C.: Convolutional models for the detection of firearms in surveillance videos. Applied Sciences **9**(15), 1–11 (2019)
16. Yahya, Z., Ullah, M.M.: Classification and Temporal Localization of Robbery Events in CCTV Videos through Multi-Stream Deep Networks. In: HONET-ICT 2019 - IEEE 16th International Conference on Smart Cities: Improving Quality of Life using ICT, IoT and AI, pp. 28-–32. IEEE, Charlotte (2019)
17. Grega, M., Lach, S., Sieradzki, R.: Automated recognition of firearms in surveillance video. In: 2013 International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support, CogSIMA, pp. 45-–50. IEEE, California (2013)
18. Grega, M., Matiolański, A., Guzik, P., Leszczuk, M.: Automated detection of firearms and knives in a CCTV image. Sensors **16**(1) (2016)
19. Tiwari, R.K., Verma, G.K.: A Computer Vision based Framework for Visual Gun Detection Using Harris Interest Point Detector. Procedia Computer Science **54**, 703-–712 (2015)
20. Weitzenfeld, A., Guardati, S.: Capítulo 12: Ingeniería de software: el proceso para el desarrollo de software. Libro: Introducción a la Computación. CENGAGE Learning. México (2007)
21. Prototyping model, https://searchcio.techtarget.com/definition/Prototyping-Model. Last accessed 04 Sep 2021
22. Becerra, A., Zepeda, G., Pérez, A., Ramirez-García, U., Esparza, S.: Learning content management software personalized for a university environment. Pistas Educativas **40**(130), 347–362 (2018)
23. Sommerville, I.: Software engineering. 6th edn. Addison-Wesley, UK (2001)
24. Openimages: A public dataset for large-scale multi-label and multi-class image classification, https://storage.googleapis.com/openimages/web/index.html. Last accessed 19 Apr 2021
25. Kaggle datasets, https://www.kaggle.com/datasets. Last accessed 20 Jun 2021
26. Image leve augmentation, https://app.roboflow.com. Last accessed 10 Nov 2020
27. Bochkovskiy, A., Wang, C.Y., Liao, H.Y.M.: Yolov4: optimal speed and accuracy of object detection. arXiv, 1–17 (2020)
28. Liu, S., et al.: Path aggregation network for instance segmentation. arXiv, 1–11 (2018)
29. He, K., Zhang, X., Ren, S., Sun, J.: Spatial pyramid pooling in deep convolutional networks for visual recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence **37**(9), 1904–1916 (2015)
30. Redmon, J., Farhadi, A.: Yolov3: an incremental improvement. arXiv, 1–6 (2018)
31. Lin, T.Y. et al.: Microsoft coco: common objects in context. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) European conference on Computer Vision – ECCV 2014, vol. 8693, pp. 740-–755. Springer International Publishing, Zurich (2014). https://doi.org/10.1007/978-3-319-10602-1_48

32. Wojke, N., Bewley, A., Paulus, D.: Simple online and realtime tracking with a deep association metric. In: 2017 IEEE International Conference on Image Processing (ICIP), pp. 3645-–3649. IEEE, Beijing (2017)
33. AWS: ultralytics/yolov5: v5.0 - YOLOv5-P6 1280 models. (2021) https://doi.org/10.5281/zenodo.4679653.
34. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in pytorch. In: 31st Conference on Neural Information Processing Systems (NIPS 2017), pp. 1–4. Curran Associates, Long Beach (2017)

ISSN 1870-4069

Clasificación de estilos fotográficos utilizando una Red Neuronal Convolucional

Andre Martin Vera Valdez, Rogelio Florencia Juárez, Gilberto Rivera Zárate, Julia Patricia Sánchez-Solís, Francisco López-Orozco, Vicente García Jiménez

Universidad Autónoma de Ciudad Juárez, Departamento de Ingeniería Eléctrica y Computación, División Multidisciplinaria de Ciudad Universitaria, México

{al154007@alumnos.uacj.mx {rogelio.florencia, gilberto.rivera, julia.sanchez, francisco.orozco, vicente.jimenez}@uacj.mx

Resumen. La fotografía es una actividad que muchas personas realizan cotidianamente para capturar en una imagen momentos importantes de sus vidas. Los fotógrafos agregan etiquetas manualmente a sus imágenes al subirlas a sitios web con la finalidad de describir aspectos importantes, como el estilo fotográfico, impulsando de esta manera su visibilidad en las búsquedas que realizan los usuarios. Sin embargo, etiquetar manualmente cada fotografía se vuelve una tarea tediosa que consume demasiado tiempo cuando se trata de una gran cantidad de imágenes, además de requerir conocimientos profundos en fotografía. Este artículo presenta la implementación de diferentes redes neuronales convolucionales para clasificar fotografías de acuerdo con 14 estilos fotográficos contenidos en el *dataset* AVA. Se entrenaron ocho modelos diferentes: a) un modelo de una red neuronal convolucional simple; b) tres modelos basados en *VGG19*, *DenseNet201* y *MobileNetV2*; c) tres modelos mediante *aprendizaje por transferencia* y d) un modelo que, en base al mejor de los anteriores, fue adicionado con estrategias para reducir el *sobreajuste*. Los resultados indican que el mejor desempeño se obtuvo al utilizar *aprendizaje por transferencia* sobre *DenseNet201* adicionado con estrategias para reducir el *sobreajuste*, alcanzando un *promedio medio de precisión* de 60.69 %.

Palabras clave: Estilos fotográficos, redes neuronales convolucionales, aprendizaje por transferencia, aumento de datos, sobreajuste.

Photographic Style Classification Using a Convolutional Neural Network

Abstract. Photography is an activity that many people perform on a daily basis to capture important moments of their lives in an image.

Photographers often add manual tags to their images when uploading them to websites in order to describe relevant aspects, such as the photographic style, thereby increasing their visibility in user searches. However, manually tagging each photograph becomes a tedious and time-consuming task when dealing with large volumes of images, in addition to requiring deep knowledge of photography. This article presents the implementation of different convolutional neural networks to classify photographs according to 14 photographic styles contained in the AVA dataset. Eight different models were trained: (a) a simple convolutional neural network model; (b) three models based on VGG19, DenseNet201, and MobileNetV2; (c) three models using transfer learning; and (d) a model that, based on the best of the previous ones, incorporated strategies to reduce overfitting. The results indicate that the best performance was obtained by applying transfer learning on DenseNet201, enhanced with strategies to reduce overfitting, achieving an average accuracy of 60.69

Keywords: Photographic styles, convolutional neural networks, transfer learning, data augmentation, overfitting.

1. Introducción

La fotografía es una actividad que forma parte de nuestro día a día; principalmente por el fácil acceso que se tiene a una cámara y a que es una tarea muy sencilla de realizar, ya que solo se tiene que poner un sujeto delante de una cámara y presionar el botón de *disparar*. El sujeto (o sujetos) es el elemento principal de una fotografía que está dentro del marco, por ejemplo: una persona, un auto, una planta, un animal, etc. El sujeto es el motivo por el que se toma una foto [1].

Según [2], la fotografía se define como, 'un acto a través del cual se produce la grabación de una situación luminosa, en un lugar y momento determinado: es la huella de la acción de la luz'. La definición indica que el elemento esencial para realizar una fotografía es la luz, sin ella no hay fotografía.

Como lo menciona Freeman [1], otro de los elementos que están presentes dentro de una fotografía es el *estilo fotográfico*. El estilo es la forma en la que se toma la fotografía, determinando el aspecto visual de la imagen; es el resultado de ciertas decisiones técnicas propias del fotógrafo en cuanto a la composición de la imagen, la distancia focal, el tiempo de exposición, y la iluminación [1]. Se puede decir que si el sujeto es el '*qué*', el estilo es el '*cómo*'. Además, Freeman [3] menciona que el estilo puede ser intencionado o no intencionado. El estilo fotográfico es importante para un fotógrafo porque está relacionado con los detalles, características, colores, etc., que caracterizan su trabajo y permiten diferenciarlo de entre otros fotógrafos. Entre los diferentes estilos fotográficos existentes se definen los siguientes [1]:

– *Motion blur*. Se aprecia el movimiento del sujeto y esto genera un desenfoque en ella.

- *Shallow DOF*. Cierta parte de la fotografía se encuentre desenfocada.
- *High contrast*. Existe una diferencia muy grande entre las zonas más oscuras y las más claras de la imagen.
- *Vanishing point*. Convergen dos o más líneas paralelas a un mismo punto.

El reconocimiento de imágenes es una de las distintas aplicaciones del aprendizaje automático [4] en donde se emplea un conjunto de métodos y técnicas para detectar y analizar imágenes con el fin de identificar lugares, personas, objetos, entre otros elementos que se encuentran dentro de éstas. Entre las diferentes tareas relacionadas con el reconocimiento de imágenes se pueden mencionar:

- Clasificación. Determina la clase a la que pertenece una imagen (ver [5]).
- Etiquetado. Identifica la presencia de varios objetos dentro de una imagen (ejemplo [6]).
- Detección. Localiza un objeto en una imagen, delimitándolo mediante un cuadro que se sitúa alrededor del objeto detectado (ver [7]).
- Segmentación. Es capaz de ubicar un elemento en una imagen al píxel más cercano (ejemplo [8]).

Recientemente, el reconocimiento de imágenes se ha empezado a utilizar en la identificación tanto de sujetos como de estilos fotográficos. Murray et al. [9] conformaron un *dataset* a gran escala con un conjunto de más de 250,000 imágenes utilizado para análisis visual estético que denominaron *Aesthetic Visual Analysis*, AVA. Los autores implementaron *Máquinas de Vectores de Soporte* lineales sobre un subconjunto de 14,079 imágenes de AVA relacionadas con 14 anotaciones de estilo fotográfico, obteniendo un *Promedio Medio de Precisión* (mAP, por sus siglas en inglés) de 53.85 %. Posteriormente, Karayev et al. [10] también utilizaron el *dataset* AVA para realizar clasificación de estilos fotográficos utilizando un algoritmo de clasificación lineal y métodos de extracción de características como histogramas de color o GIST, siendo una *Red Neuronal Convolucional* (CNN, por sus siglas en inglés) el método más efectivo con un mAP de 57.9 % y con la fusión de múltiples características un 58.1 %. Además, los autores propusieron otro *dataset* basado en imágenes recolectadas de la red social *Flickr*, obteniendo un mAP de 33.6 %. Celona et al. [11] propusieron una multired para la predicción automática de la estética de una imagen en base al análisis del contenido semántico, el estilo artístico y la composición de la imagen. La red propuesta está compuesta por una red preentrenada para la extracción de características semánticas, un perceptrón multicapa para la predicción de los atributos de la imagen y una hiper-red autoadaptativa para predecir los parámetros de la red dedicada a la estimación estética. Los autores reportaron una *exactitud* de un 80.75 %. Por otra parte, debido a que el reconocimiento de estilo no está limitado solo al área de la fotografía, en el trabajo de Pérez [12] se desarrolló una CNN para la clasificación de estilos en pinturas. La efectividad obtenida de la red fue un 51 %.

Hoy en día, tanto fotógrafos novatos como profesionales comparten sus contenidos fotográficos en sitios como *Instagram*, *Flickr*, *500px*, o *Shutterstock*,

ya sea simplemente por gusto o por una cuestión de comercialización. Al subir contenidos a estas plataformas, los fotógrafos tienen la opción de colocar manualmente etiquetas descriptivas a sus fotografías para posicionarlas mejor dentro de las búsquedas y lograr que sus contenidos lleguen a más personas, siendo el estilo fotográfico una de estas etiquetas. Esto se vuelve una tarea tediosa y tardada para el fotógrafo cuando se trata de una gran cantidad de fotografías lo cual resalta la necesidad de automatizar su clasificación de acuerdo con su estilo fotográfico.

En este artículo se describe la implementación de diferentes arquitecturas de CNNs para la clasificación de fotografías según su estilo fotográfico. Las CNNs fueron entrenadas sobre los 14 estilos fotográficos contenidos en el *dataset* AVA [9]. Integrar el modelo entrenado por la CNN en una aplicación podría ayudar a los fotógrafos a etiquetar automáticamente sus fotografías, además de no requerir conocimientos profundos en fotografía.

El artículo está estructurado de la siguiente manera. La Sección 2 presenta una descripción del conjunto de imágenes utilizado para entrenar las CNNs. La Sección 3 describe las diferentes arquitecturas de las CNNs entrenadas para la clasificación de fotografías en base a su estilo fotográfico. La Sección 4 muestra los resultados de los diferentes modelos entrenados y presenta una discusión sobre los resultados obtenidos. Por último, la Sección 5 menciona los hallazgos encontrados en este trabajo y sugiere algunos trabajos futuros.

2. Materiales y métodos

Las CNNs se entrenaron utilizando el *dataset* AVA [9]. Este *dataset* es multipropósito y contiene 255,510 imágenes en formato JPG. Las imágenes están anotadas en base a los siguientes aspectos:

- Calificación de estética. Calificación dada por los usuarios del sitio web de donde se obtuvieron las imágenes con el fin de entrenar clasificadores que sean capaces de predecir lo buenas o malas que son las fotografías en base a una nota numérica.
- Anotaciones semánticas. Etiquetas que describen el contenido de las imágenes. Estas pueden ser utilizadas para realizar reconocimiento de sujetos dentro de las fotografías, algunas de las etiquetas son *naturaleza*, *arquitectura*, y *flora*.
- Anotaciones de estilo. Contiene 14 estilos fotográficos asociados a 14,079 imágenes dentro del *dataset*. Los estilos que contempla son *Complementary colors*, *Duotones*, *High Dynamic Range*, *Image grain*, *Light on white*, *Long exposure*, *Macro*, *Motion blur*, *Negative image*, *Rule of thirds*, *Shallow DOF*, *Silhouttes*, *Soft focus*, y *Vanishing point*.

Debido a que el interés de este artículo se centra en la clasificación de estilos fotográficos, solo se contemplaron las imágenes anotadas con los 14 estilos fotográficos donde cada estilo se consideró como una etiqueta de clase distinta. No obstante, la clasificación de estilos fotográficos se abordó desde

Tabla 1. Distribución de las imágenes utilizadas en cada estilo fotográfico.

Estilo	Imágenes
Complementary Colors	760
Duotones	1,040
High Dynamic Range	315
Image Grain	671
Light on White	960
Long Exposure	674
Macro	1,357
Motion Blur	486
Negative Image	766
Rule of Thirds	1,111
Shallow DOF	1,183
Silhouettes	824
Soft Focus	566
Vanishing Point	540

una perspectiva multiclase (donde una fotografía pertenece a una sola clase); por lo que las imágenes del *dataset* que pertenecían a más de un estilo se descartaron, dando como resultado un total de 11,253 imágenes utilizadas para el entrenamiento de las CNNs. En la Tabla 1 se muestra la distribución de las imágenes utilizadas en cada estilo fotográfico.

3. Arquitectura propuesta

En primera instancia, la Subsección 3.1 describe el preprocesamiento realizado al conjunto de imágenes utilizado. En las siguientes secciones se presentan las arquitecturas implementadas. La Subsección 3.2 describe la implementación de una CNN simple. La Subsección 3.3 presenta el uso de tres CNNs preentrenadas, existentes en la literatura. La Subsección 3.4 muestra la arquitectura de tres CNN utilizando *transfer learning* a partir de las CNNs preentrenadas. Por último, la Subsección 3.5 describe el uso de estrategias para reducir el sobreajuste en el mejor modelo de los anteriores para incrementar su rendimiento.

3.1. Preprocesamiento de las imágenes

Las 11,253 imágenes que se tomarón del *dataset* AVA [9] se separaron en tres conjuntos, *Train*, *Test* y *Valid*. El 80 % de las imágenes se almacenaron en una carpeta llamada *Train*, el 10 % en una carpeta llamada *Test* y el resto en una carpeta llamada *Valid*. Dentro de estas carpetas, las imágenes se almacenaron en

Tabla 2. Imágenes utilizadas para el entrenamiento de las CNNs.

Estilo	*Train*	*Test*	*Valid*
Complementary colors	608	76	76
Duotones	832	104	104
HDR	253	31	31
Image grain	537	67	67
Light on white	768	96	96
Long exposure	540	67	67
Macro	1,087	135	135
Motion blur	390	48	48
Negative image	614	76	76
Rule of thirds	889	111	111
Shallow DOF	947	118	118
Silhouettes	660	82	82
Soft focus	454	56	56
Vanishing point	432	54	54

carpetas de acuerdo a su estilo fotográfico. La cantidad de imágenes utilizadas se muestran en la Tabla 2.

Posteriormente, los valores de los píxeles de las imágenes se normalizaron entre 0 y 1, en lugar de 0 y 255. Adicionalmente, las imágenes se redimensionaron a un tamaño de 256×256 píxeles utilizando el módulo *ImageDataGenerator* de *Keras* [13].

3.2. Red Neuronal Convolucional

Como primera instancia, se implementó una nueva CNN simple siguiendo algunos de los principios de diseño como los presentados en [14] y [15], tales como crecimiento gradual de la red, comenzar con filtros pequeños, utilizar filtros de tamaño impar, entre otros.

De entre las diferentes arquitecturas diseñadas, la que mejor resultado proporcionó estuvo compuesta por: 3 capas convolucionales con 32, 64 y 128 filtros; 1 capa de *max pooling* entre cada una de estas; 1 capa densa con 14 neuronas de salida, una por cada estilo fotográfico. En cuanto al optimizador se utilizó *Adam* con una tasa de aprendizaje de 0.00001. Cabe mencionar que los parámetros de las CNNs que se presentan en este artículo se definieron en base a prueba y error. La Figura 1 muestra la arquitectura de la CNN implementada.

Después de entrenar esta CNN durante 100 épocas, los resultados no fueron aceptables, ya que se alcanzó un *mAP* de 31.75 %. Además, el modelo comenzó a mostrar que se empezó a *sobreajustar* (overfitting) (ver Figura 2). Con esto se pensó que el bajo desempeño se pudo deber a que el número de imágenes utilizadas para entrenar la CNN no fueron suficientes.

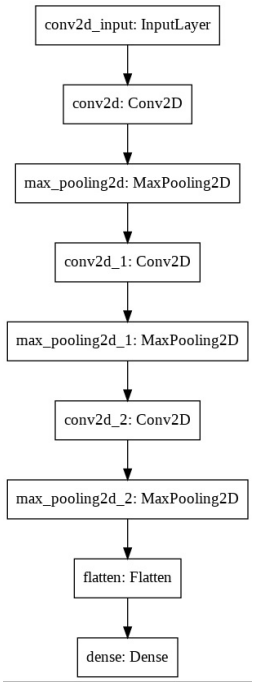

Fig. 1. Modelo de la CNN simple.

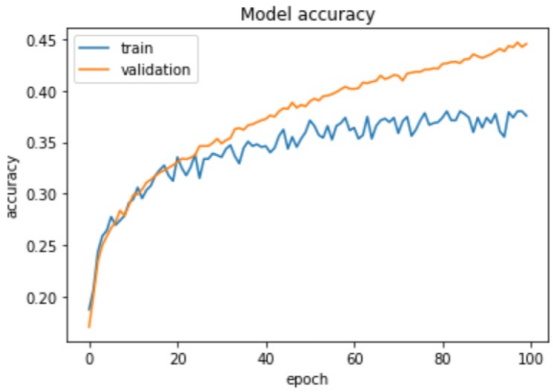

Fig. 2. Muestras de overfitting de la CNN simple.

3.3. Red Neuronal Convolucional preentrenada

Debido a que construir y entrenar nuevas CNNs requiere largos períodos de tiempo y grandes cantidades de imágenes, se optó por utilizar las CNNs

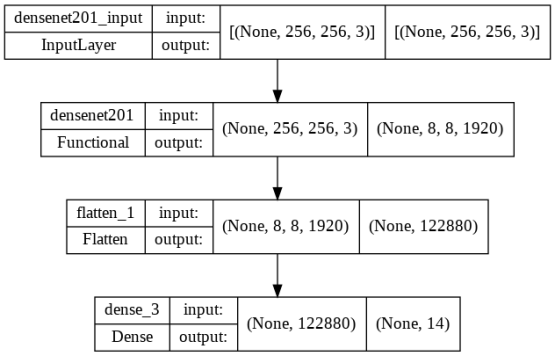

Fig. 3. Modelo basado en *DenseNet201*.

preentrenadas *VGG19* [16], *DenseNet201* [17], y *MobileNetV2* [18], existentes en la literatura.

Estos modelos se instanciaron desde *Keras* utilizando el módulo *applications* [19]. Para cada una de estas tres CNNs preentrenadas se creó un nuevo modelo que tiene como primera capa todo el contenido del modelo instanciado. Como salida de las tres CNNs se definió una capa densa de 14 neuronas, una tasa de aprendizaje (learning rate) de 0.0000065, se utilizó el optimizador *Adam* y se entrenaron por 12 épocas. Los modelos se entrenaron 12 épocas debido al indicio de overfitting mostrado en la Figura 2. El modelo que obtuvo el mejor desempeño fue *DenseNet201* con un *mAP* de 27.1 %. En la Figura 3 se presenta la arquitectura de esta CNN. Debido a que los resultados tampoco se consideraron como aceptables, se siguió pensando que posiblemente la causa fue el bajo número de imágenes.

3.4. Aprendizaje por transferencia

Con el fin de mejorar los resultados de clasificación que se habían obtenido hasta al momento, se utilizó la técnica de *aprendizaje por transferencia* (*transfer learning*) la cual consiste en utilizar un modelo previamente entrenado y '*transferir*' ese aprendizaje a un nuevo modelo [20].

Para implementar esta técnica se utilizaron los mismos modelos *VGG19*, *DenseNet201*, y *MobileNetV2*. Estos modelos de la literatura fueron preentrenados en *ImageNet* [21], por lo que se utilizaron los pesos aprendidos de ese *dataset*, los cuales son proporcionados por *Keras*.

Se puede implementar *transfer learning* de distintas maneras, por ejemplo: utilizar el modelo preentrenado directamente en el nuevo dominio, hacer un ajuste fino del modelo 'congelando' algunas de sus capas para que no se vean afectadas durante el entrenamiento, o utilizar una parte del modelo para integrarlo en uno nuevo [20].

Después de realizar distintas pruebas con algunas de las estrategias mencionadas anteriormente, utilizar y entrenar el modelo completo sin 'congelar' ninguna de sus capas y mantener una tasa de aprendizaje de 0.0000065 fue la estrategia que dio mejores resultados. Nuevamente, al igual que en la etapa anterior, el modelo con el mejor desempeño fue *DenseNet201* con un mAP del 52.15 %.

3.5. Reducción del sobreajuste

Apesar de que el modelo basado en *DenseNet201 + transfer learning* obtuvo un mejor desempeño, se decidió aplicarle técnicas de reducción del *overfitting* con el fin de mejorar su desempeño.

En primera instancia, se utilizó la estrategia *dropout*, la cual consiste en establecer en 0 de manera aleatoria cierto porcentaje de las neuronas de una o más capas durante el entrenamiento de la red [24]. Para la implementación de esta técnica se utilizó la clase *Dropout* de *Keras*, la cual viene incluida en el módulo *layers*. Se agregaron dos nuevas capas completamente conectadas antes de la capa de salida, y en medio de cada una de ellas, se agregó una capa de *dropout* con una tasa del 40 %. La arquitectura del modelo *DenseNet201* modificado se muestra en la Figura 4. De esta manera, el modelo alcanzó un mAP de 59.77 %.

Posteriormente, se utilizó el *aumento de datos* (data augmentation), el cual consiste en generar nuevos datos a partir de los existentes [25]. Algunas de las opciones para aplicar *data augmentation* son girar la imagen, invertirla, proyectarla sobre sus ejes, hacer zoom, entre otras [26]. Debido a que algunas de estas estrategias podría afectar el estilo fotográfico de las imágenes, solo se hicieron dos transformaciones por cada imagen: proyecciones en su eje horizontal y proyecciones en su eje vertical. Estas dos transformaciones se utilizaron para crear dos nuevos conjuntos de imágenes. Para generar el primer conjunto se aplicaron a todas las imágenes, dando como resultado un total de 33,759 imágenes (completo). Para el segundo sólo se aplicaron a las clases con menos de 1,000 imágenes hasta alcanzar dicha cantidad con la finalidad de solo lograr una menor tasa de desbalance entre las clases, resultando un total de 14,691 imágenes (balanceado). Para aplicar *data augmentation* se utilizó la librería *OpenCV* [23]. En la Figura 5 se aprecia una imagen de ejemplo aplicando estas dos transformaciones. El modelo se entrenó nuevamente pero ahora utilizando cada uno de estos dos nuevos conjuntos de imágenes, alcanzando un mAP de 60.69 % en el primer conjunto y un 60.1 % en el segundo.

4. Resultados y discusiones

Para la implementación de los modelos descritos en la Sección 3 se utilizó el lenguaje de programación *Python* y las librerías *Keras* y *Tensorflow* por medio del servicio en la nube de *Google Colab*. El resto de esta sección se estructura de la siguiente manera. La Subsección 4.1 presenta los resultados de la evaluación

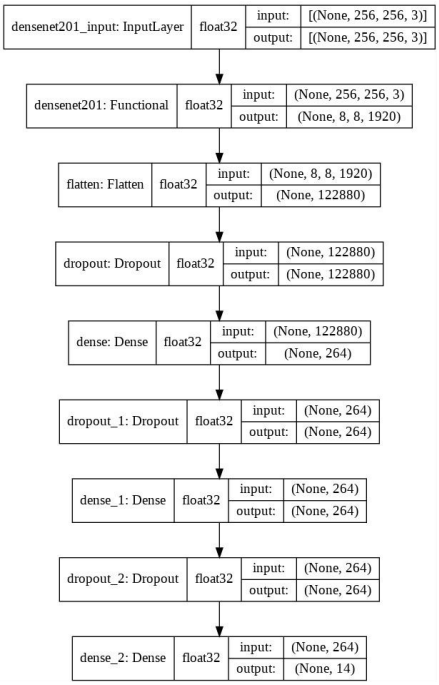

Fig. 4. Modelo *DenseNet201 + dropout + transfer learning*.

Fig. 5. Ejemplo de *data augmentation* de una imagen.

de los modelos desarrollados. La Subsección 4.2 presenta una discusión a partir de los resultados obtenidos.

4.1. Evaluación de los modelos desarrollados

En la Tabla 3 se presenta una comparativa del desempeño de los distintos modelos descritos en la Sección 3 en términos de la métrica *mAP*. La primera columna contiene el nombre del modelo entrenado; la segunda indica si se utilizó *transfer learning*; la tercera muestra el número de épocas utilizadas para entrenar cada uno de los modelos y, la última columna muestra el desempeño

Tabla 3. Resultados obtenidos de las distintas CNNs entrenadas.

Modelo	Transfer learning	Épocas	% mAP
CNN simple	No	100	31.75
VGG19	No	12	26.33
DenseNet201	No	12	27.1
MobileNetV2	No	12	12.19
VGG19	Imagenet	12	37.83
DenseNet201	Imagenet	12	52.15
MobileNetV2	Imagenet	12	42.41
DenseNet201 + dropout	**Imagenet**	**20**	**59.77**

Tabla 4. Resultados obtenidos del modelo *DenseNet201 + dropout + transfer learning + data augmentation.*

Modelo	Data augmentation	Épocas	% mAP
DenseNet201 + dropout	No	20	59.77
DenseNet201 + dropout	Balanceado	20	60.1
DenseNet201 + dropout	**Completo**	**10**	**60.69**

de cada modelo utilizando la métrica *mAP*. Como se puede observar, el modelo *DenseNet201 + dropout + transfer learning* obtuvo el mejor desempeño con un *mAP* de 59.77 %. Por otra parte, el modelo *MobileNetV2* obtuvo el peor desempeño con un 12.19 %.

La Tabla 4 muestra los resultados del mejor modelo, *DenseNet201 + dropout + transfer learning* utilizando los dos conjuntos de imágenes aumentados, descritos en la Subsección 3.5. Como se puede ver, haber entrenado éste modelo con ambos conjuntos de imágenes mejoran su desempeño con un *mAP* de 60.1 % y de 60.69 %.

La Figura 6 muestra la matriz de confusión del modelo *DenseNet201 + dropout + transfer learning + data augmentation* completo. Como se puede observar, las clases con un bajo número de predicciones correctas fueron *motion blur* con un 29 % y *rule of thirds* con 31 %. Por tal motivo, se realizó un análisis para determinar si el bajo desempeño de la red se debió a los datos.

El modelo confundió el estilo *motion blur* con el estilo *long exposure* posiblemente porque ambos estilos hacen referencia a movimiento dentro de la imagen [3], [1]. En la Figura 7 se aprecia la similitud entre estos estilos. Como se puede ver, ambos presentan movimiento, sin embargo, en *long exposure* no necesariamente hay un desenfoque completo, pues las teclas del piano están perfectamente enfocadas.

Por otra parte, *rule of thirds* también presentó confusión con todas las clases. Este estilo refiere a la posición en la que se coloca el sujeto principal de una

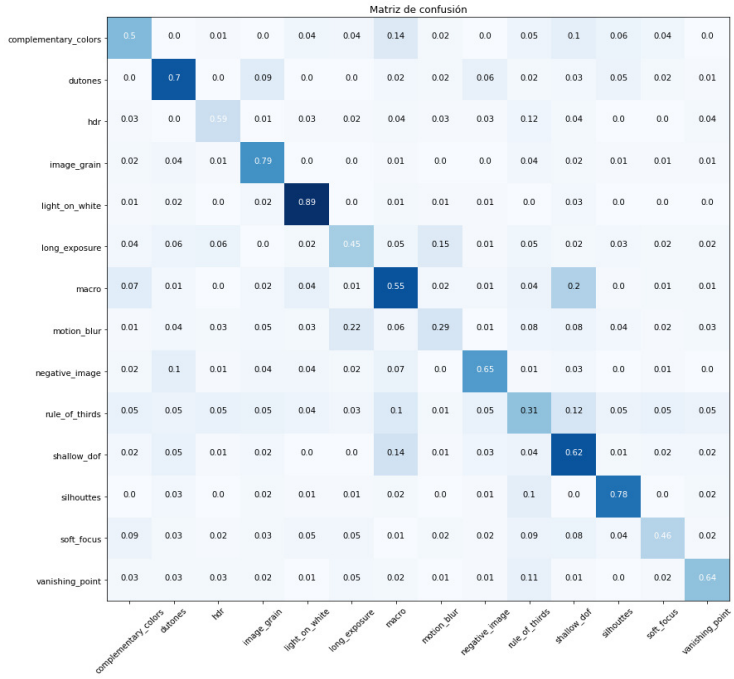

Fig. 6. Matriz de confusión del modelo *DenseNet201 + dropout + transfer learning + data augmentation* completo.

Fig. 7. a) Imagen izquierda, estilo *motion blur*. b) Imagen derecha, estilo *long exposure*

imagen, esta dice que el encuadre se debe dividir en tres partes iguales tanto horizontales como verticales y colocar el sujeto a fotografiar en alguna de las intersecciones de estas divisiones [27]. Este estilo puede confundir al modelo, pues al tratarse de posición, se puede dar el caso de que este estilo se encuentre embebido en alguna de las otras categorías. En la Figura 8 se muestra una imagen

Fig. 8. Imagen del estilo *shallow DOF* donde también se cumple el estilo *rule of thirds*.

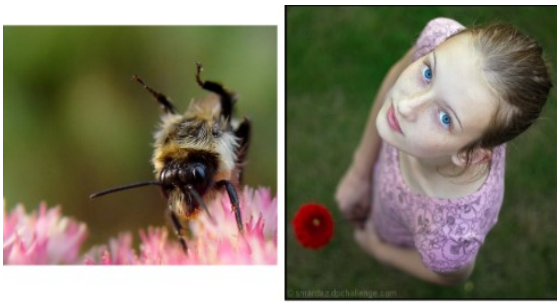

Fig. 9. Ejemplo del desenfoque del fondo en los estilos *macro* y *shallow DOF*.

que corresponde al estilo de *shallow DOF*, sin embargo, se observa que *rule of thirds* también se cumple.

Otro par de estilos en los que se puede apreciar confusión entre sí, son *macro* y *shallow DOF*, pues *macro* puede implicar una reducción de la profundidad de campo en algunos casos [28]. En la Figura 9 se aprecia cómo ambas imágenes tienen poca profundidad de campo, esto es, el efecto de desenfoque en el fondo [1].

Para verificar el impacto del 'ruido' que estos estilos podrían introducir al modelo, éste se entrenó nuevamente sin considerar los estilos *rule of thirds*, *motion blur*, y *shallow DOF*. En la Tabla 5 se muestra el *mAP* de los resultados obtenidos.

Como se puede observar, el modelo fue mejorando su desempeño conforme se fueron quitando cada uno de estos tres estilos, hasta alcanzar un *mAP* de 73.61 %. En la Figura 10 se muestra la matriz de confusión resultante al quitar los estilos *rule of thirds*, *motion blur* y *shallow DOF*.

Con el fin de obtener una mejor estimación de la habilidad del modelo para predecir datos no vistos durante el entrenamiento, se utilizó el método de validación cruzada denominado *stratified k-folds* [29]. Se utilizaron 10 folds

Tabla 5. Resultados de *DenseNet201 + dropout + transfer learning + data augmentation* completo sin los estilos *Rule of thirds, motion blur* y *shallow DOF*.

Clases removidas	% mAP
Ninguna	60.69
Rule of thirds	65.32
Rule of thirds + motion blur	68.66
Rule of thirds + motion blur + shallow DOF	**73.61**

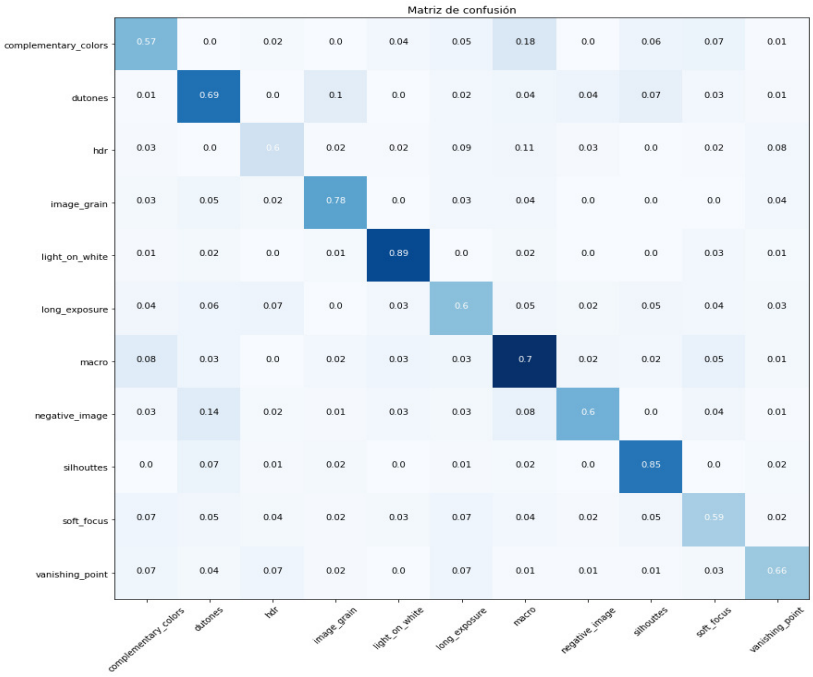

Fig. 10. Matriz de confusión del modelo *DenseNet201 + dropout + transfer learning + data augmentation* completo con 11 clases.

sobre el conjunto de datos aumentado completamente, los resultados se muestran en la Tabla 6.

4.2. Discusiones

Si bien, 11,253 imágenes pueden parecer muchas para entrenar una CNN, los resultados demuestran lo contrario, ya que aquellos modelos que se entrenaron sin ningún tipo de *transfer learning* se comportaron peor que aquellos que sí utilizaban pesos previamente aprendidos de otro *dataset*. Además, entrenar modelos desde cero requiere un tiempo mayor de entrenamiento.

Tabla 6. Resultados del modelo *DenseNet201 + dropout + transfer learning + data augmentation* completo utilizando 11 clases y validación cruzada de 10 folds.

Fold	% mAP
1	60.05
2	61.58
3	59.83
4	61.93
5	61.96
6	60.69
7	56.70
8	59.63
9	58.22
10	60.46

Por otra parte, en cuanto a las técnicas utilizadas para reducir el *overfitting*, *dropout* tuvo mayor impacto positivo en el desempeño del modelo que al utilizar *data augmentation*. Lo anterior puede ser debido al dominio de aplicación, pues al aumentar los datos se pudo haber modificado el estilo de la imagen, impactando negativamente en el desempeño del modelo.

El desempeño alcanzado por el modelo pudo deberse a que algunos de los estilos no son mutuamente excluyentes, es decir, una imagen puede contener más de un estilo, aunado a esto se encuentra la similitud entre clases como *motion blur* y *long exposure*. Lo anterior se ve reforzado con el hecho de que al quitar los estilos que introducían ruido, el modelo mejoró su desempeño, pasando de un mAP de 60.69 % a un 73.61 %.

5. Conclusiones y trabajos futuros

En este artículo se presentó la implementación de diferentes CNNs para clasificar fotografías de acuerdo con su estilo fotográfico. Se entrenaron ocho modelos de clasificación, uno de ellos fue una CNN simple y los demás se implementaron utilizando las CNNs preentrenadas *VGG19*, *DenseNet201*, y *MobileNetV2*, *transfer learning* y estrategias para reducir el *overfitting*.

Los modelos se entrenaron en 11,253 imágenes anotadas con 14 estilos fotográficos del *dataset* AVA, las cuales se normalizaron entre 0 y 1 y se redimensionaron a 256×256 píxeles. Además, se utilizó *data augmentation* para crear dos nuevos conjuntos de imágenes. En el primero se transformó completamente el conjunto de imágenes, resultando 33,759 imágenes (completo). En el segundo se aumentaron las clases con menos de 1,000 imágenes con la finalidad de balancear las clases, resultando 14,691 imágenes (balanceado).

Los resultados demuestran que el mejor modelo fue *DenseNet201 + dropout + transfer learning + data augmentation* completo obtuvo el mejor desempeño, alcanzando un mAP de 60.69 %.

Al analizar los resultados se identificó a través de la matriz de confusión que tres estilos fotográficos pudieron haber generado ruido en el modelo, los cuales fueron *rule of thirds*, *motion blur* y *shallow DOF*. Al eliminar estos estilos del conjunto de imágenes y volver a entrenar el modelo, éste alcanzó un mAP de 73.61 %. Esto debido a que existe similitud entre estos tres estilos y a que una fotografía puede pertenecer a más de un estilo en AVA.

Como trabajo futuro se sugiere recolectar y/o construir un *dataset* con una gran cantidad de imágenes para verificar si existe una mejora al entrenar CNNs simples sin la necesidad de utilizar *transfer learning*. Además, se pretende implementar diferentes arquitecturas de CNNs para abordar clasificación multietiqueta en casos en los que una misma imagen pudiera pertenecer a más de un estilo fotográfico.

Referencias

1. M. Freeman, The photographer's mind, Lewes, UK: Elsevier, 2011.
2. A. P. Martínez Lanz Durán, Memorias: fotgrafía pictórica, tesis, Cholula, Puebla: Universidad de las Américas de Publa, 2003.
3. M. Freeman, El estilo en fotografía, Madrid, España: H. Blume, 1986.
4. Y. LeCun, Y. Bengio y Geoffrey Hinton, Deep learning, de Nature 521, 2015.
5. Roldán, A. K. M., Sánchez-Solís, J. P., Jiménez, V. G., Juárez, R. F., & Zárate, G. R. (2020). Convolutional Neural Network in a Pseudo-Distributed Environment for Classification of Chest X-Ray Images of Patients with Pneumonia. Res. Comput. Sci., 149(5), 101-110.
6. Song, G., Wang, Z., Han, F., Ding, S., & Iqbal, M. A. (2018). Music auto-tagging using deep recurrent neural networks. Neurocomputing, 292, 104-110.
7. Srivastava, S., Divekar, A. V., Anilkumar, C., Naik, I., Kulkarni, V., & Pattabiraman, V. (2021). Comparative analysis of deep learning image detection algorithms. Journal of Big Data, 8(1), 1-27.
8. Xu, Y., Wang, Y., Yuan, J., Cheng, Q., Wang, X., & Carson, P. L. (2019). Medical breast ultrasound image segmentation by machine learning. Ultrasonics, 91, 1-9.
9. N. Murray, L. Marchesotti y F. Perronnin, AVA: A large-scale database for aesthetic visual analysis, de 2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, 2012.
10. S. Karayev, M. Trentacoste, H. Han, A. Agarwala, T. Darrell, A. Hertzmann y H. Winnemoeller, Recognizing Image Style, de British Machine Vision Conference, 2014.
11. Celona, L., Leonardi, M., Napoletano, P., & Rozza, A. (2022). Composition and style attributes guided image aesthetic assessment. IEEE Transactions on Image Processing, 31, 5009-5024.
12. I. Pérez Roldán, Clasificación de obras de arte por estilo artístico usando redes neuronales convolucionales, proyecto de fin de grado, Universisdad Politécnica de Madrid, 2019.
13. Keras, ≪Image data preprocessing,≫ Keras, [En línea]. Available: https://keras.io/api/preprocessing/image/. [Último acceso: 10 Marzo 2021].
14. S. Ramesh, ≪Towards Data Science,≫ 7 Mayo 2018. [En línea]. Available: https://towardsdatascience.com/a-guide-to -an-efficient-way-to-build-neural-network- architectures-part-ii-hyper-parameter-42efca01e5d7. [Último acceso: 4 Enero 2021].

15. H. H. Seyyed , R. Mohammad , F. Mohsen , S. Mohammad y A. Ehsan , ≪Towards Principled Design of Deep Convolutional Networks: Introducing SimpNet,≫ 17 Febrero 2018. [En línea]. Available: https://arxiv.org/abs/1802.06205. [Último acceso: 2 Enero 2021].
16. S. Karen y Z. Andrew, ≪Very deep convolutional networks for large-scale image recognition,≫ de International Conference on Learning Representations, Toulon, France, 2015.
17. G. Huang, Z. Liu, L. Van Der Maaten y W. Kilian, ≪Densely Connected Convolutional Networks,≫ de 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, 2018.
18. M. Sandler, A. Howard, M. Zhu, A. Zhmoginov y L.-C. Chen, ≪MobileNetV2: Inverted Residuals and Linear Bottlenecks,≫ de The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, Salt Lake City, Utah, 2018.
19. Keras, ≪Keras Applications,≫ Keras, [En línea]. Available: https://keras.io/api/applications/. [Último acceso: 10 Enero 2021].
20. J. Brownlee, Transfer Learning in Keras with Computer Vision Models, Machine Learning Mastery, 18 Agosto 2020. [En línea]. Available: https://machinelearningmastery.com/how-to-use-transfer-learning-when-developing-convolutional-neural-network-models/. [Último acceso: 20 Marzo 2021].
21. L. Fei-Fei, J. Deng, O. Russakovsky, A. Berg y K. Li, Imagenet, Imagenet, 19 Mayo 2015. [En línea]. Available: http://www.image-net.org/about. [Último acceso: 18 Abril 2021].
22. D. Rodvold, A software development proccess model for artificial neural networks in critical applications, de International Joint Conference on Neural Networks, Washington, DC, 2002.
23. OpenCV, OpenCV modules, OpenCV, [En línea]. Available: https://docs.opencv.org/master/. [Último acceso: 10 Abril 2021].
24. Keras, Dropout layer, Keras, [En línea]. Available: https://keras.io/api/layers/regularization_layers/dropout/. [Último acceso: 14 Enero 2021].
25. Rondón, C. V. N., Carvajal, D. A. C., Casadiego, S. A. C., Delgado, B. M., & Ibarra, D. G. Dataset para la detección de elementos de bioseguridad facial mediante técnicas de aprendizaje computacional.
26. TensorFlow, Aumento de datos, TensorFlow, 19 Marzo 2021. [En línea]. Available: https://www.tensorflow.org/tutorials/images/data_augmentation. [Último acceso: 10 Abril 2021].
27. S. A. Amirshahi, . U. G. Hayn-Leichsenring, D. Joachim y C. Redies, Evaluating the Rule of Thirds in Photographs and Paintings, Art & Perception, vol. 2, pp. 163-182, 2014.
28. R. Sheppard, Macro Photography From Snapshots To Great Shots, Peachpit Press, 2015.
29. Scikit Learn, 3.1. Cross-validation: evaluating estimator performance, [En línea]. Available: https://scikit-learn.org/stable/modules/cross_validation.html. [Último acceso: 20 Abril 2021].

Electronic edition
Available online: http://www.rcs.cic.ipn.mx

http://rcs.cic.ipn.mx

Centro de Investigación
en Computación