Universidad Católica 'Nuestra Señora de la Asunción' Campus Itapúa

Facultad de Ciencias y Tecnologías

Trabajo final de grado

Ingeniería Informática

Entrenamiento de redes neuronales para la detección en tiempo real de amenazas y agresiones humanas en imágenes secuenciales

Gustavo Enrique Escobar Krug

Docente tutor:

Lic. Nidia Gagliardi

Dedicatoria

A mis abuelos,
donde quiera que estén.

Agradecimientos

A toda mi familia,
que supo ayudarme y apoyarme
en los tiempos más difíciles.
Mi deuda con ustedes es inmensa.

Índice general

Índice de figuras

Capítulo 1

Introducción

1.1. Planteamiento General

La delincuencia es un mal que afecta a toda sociedad en crecimiento sin importar a veces el estrato social de la víctima, a quien, en muchas ocasiones, el perjuicio puede suponer la pérdida completa de sus bienes de alto valor, o incluso hasta la vida.

Numerosas personas y empresas de la zona (como en otras partes del mundo) han optado por disponer de cámaras de seguridad y vigilancia en sus casas o locales comerciales, fábricas, depósitos, etc. , y aunque se ha comprobado que esto no contribuye totalmente a la disminución de la cantidad de delitos ocurridos, puede usarse como medio de monitoreo para la detección de intrusos o la ocurrencia de delitos, mediante la visualización remota en tiempo real por parte de personas contratadas para tal efecto. O como ocurre comúnmente en nuestra zona: permite visualizar el hecho delictivo una vez consumado, en caso de no contar con vigilancia en tiempo real.

Los servicios de vigilancia remota se ven sobrepasados en ciertas ocasiones, por la cantidad de empresas interesadas en contratar estos servicios. En ciudades mas grandes, el monitoreo remoto mediante el uso de cámaras ha acaparado prácticamente todos los rincones de la ciudad, no solamente dentro de locales comerciales o viviendas, sino que también calles, plazas, peatonales, espacios de esparcimiento en general o en algún otro lugar donde puedan ocurrir delitos, y ésto muchas veces supone un alto coste a dichas ciudades (Ejemplo: ciudad de Buenos Aires, República Argentina).

En ocasiones, las empresas medianas o pequeñas no pueden, por motivos económicos, contratar servicios de vigilancia las 24hs al día los 365 días del año, porque

esto supone, como mínimo, el pago de un sueldo extra (o mas, debido al trabajo nocturno y días no laborales) para la contratación de personal que pueda vigilar la actividad captada por las cámaras instaladas a tal efecto.

1.2. Planteamiento del problema

El uso de la tecnología, ha permitido a las personas, detectar o prevenir en el mejor de los casos la actividad delincuencial.

Mediante el entrenamiento de redes neuronales de computadoras, se intenta detectar, de forma básica, elementos que supongan actividades delincuenciales (como agresiones o amenazas a la integridad de objetos) captadas por cámaras de seguridad, brindando no una solución final y completa al problema, sino como una solución asistencial primaria, estableciendo una metodología básica para estudios posteriores mas avanzados o para casos de investigación sobre otro tipo de actividades.

Las redes neuronales, debidamente entrenadas para la detección de actividades humanas, serían capaces de detectar y ofrecer una alerta temprana, sobre algún hecho ocurriendo en tiempo real en el instante preciso en que el hecho se esté realizando fuera de la vista del personal de vigilancia.

1.3. Justificación

La tarea de monitoreo de circuitos cerrados de televisión supone la contratación de personal permanente y además, en la mayoría de los casos, la cobertura de turnos completos de trabajo, sin descanso. Esto, como dijimos anteriormente, supone el desembolso de grandes sumas de dinero constante para el pago de estos servicios.

Esto sumado a el hecho de que el ser humano tiene falencias naturales, como el cansancio o la distracción, la implementación de un sistema inteligente que permita ayudar al personal de vigilancia a detectar ciertos tipos amenazas o agresiones humanas sería de gran utilidad: se podrían utilizar computadores potentes que sean capaces de 'monitorear' múltiples escenarios todo el tiempo, sin descanso, sin dis-

tracciones.

1.4. Objetivos

1.4.1. General

Desarrollar una solución computacional que permita la detección básica de delitos de agresión o amenazas, mediante el entrenamiento de redes neuronales de computadoras.

1.4.2. Específicos

1. Entrenamiento de redes neuronales mediante secuencias de poses humanas establecidas para el fin.
2. Analizar las imágenes secuenciales captadas por cámaras.
3. Obtener patrones de poses humanas ocurrentes en las imágenes.
4. Verificar patrones coincidentes con los patrones de poses humanas preestablecidos a través del entrenamiento previo.

1.5. Alcance

Se intenta establecer una metodología básica a través del desarrollo de software para la detección de posibles delitos de agresión o amenazas humanas.

Este trabajo de investigación no contemplará el aviso de actividad delincuencial a entes de seguridad, tampoco se prevee una detección completa de toda actividad humana conocida por el hombre, sino que pretende establecer bases o estrategias para la detección primaria de actividades expresadas a través de la pose para la cual la red neuronal haya sido previamente entrenada.

El trabajo no contemplará el uso de varios orígenes de imágenes secuenciales de manera simultánea ni tampoco el reconocimiento de características de objetos o personas que participan en las secuencias.

1.6. Metodología

1.6.1. Diseño y tipo de investigación

Según la finalidad de este trabajo, se pretende llevar a cabo un proceso de investigación experimental verdadero para el entrenamiento de redes neuronales y posterior detección de patrones de poses humanas utilizando las mismas redes.

1.6.2. Procedimiento

Para que las redes neuronales sean capaces de detectar patrones, es necesario primeramente entrenarlas mediante la extracción de características de poses en seres humanos analizando imágenes secuenciales, una a una.

Para llevar a cabo esto:

- Se obtendrán patrones existentes de poses en imágenes secuenciales que sugieran algún tipo de actividad humana sobre hechos delictivos relacionados a amenazas o agresiones humanas, mediante el tratamiento digital de imágenes. Esta información será utilizada para entrenar la red neuronal.
- Se entrenará una red neuronal de computadoras, utilizando técnicas actuales de inteligencia artificial, para que la red pueda almacenar los patrones previamente obtenidos de las imágenes secuenciales. Estos patrones establecerán, los parámetros para la identificación de actividad humana en las imágenes secuenciales a través de las poses detectadas.
- Se analizarán nuevos patrones, y se compararán los mismos con los patrones establecidos anteriormente, a fin de que pueda ser, o no, confirmado como un patrón ocurrente de una actividad humana similar a la entrenada.

Capítulo 2

Estado del Arte

2.1. Introducción

En la actualidad, la seguridad personal se está volviendo cada vez mas importante: los atracos, asaltos y robos se producen a plena luz del día, muchas veces a la vista de todos y de una manera cada vez mas violenta. La violencia utilizada durante estos actos es cada vez mayor, esto debido a múltiples factores tales como para evitar la resistencia de la victima, el tiempo en cometer el delito, etc. Las condiciones de seguridad existentes no parecen frenar esta violencia y hasta resulta inevitable en muchos casos.

Con el transcurrir de éstos últimos años, muchas personas y empresas dedicadas a la tecnología han centrado sus esfuerzos en ofrecer soluciones que puedan brindar ayuda incluso a los organismos de seguridad. Las cámaras de circuito cerrado de televisión (CCTV) han tomado un protagonismo mas evidente en los sistemas de vigilancia dada la ventaja que ofrecen.

2.1.1. Evolución de los sistemas de vigilancia

A lo largo del tiempo, los sistemas de vigilancia ha mejorado la tecnología utilizada en los dispositivos de monitoreo, sensores, etc. Según M. Valera y S.A. Velastin. [1], pueden diferenciarse hasta el momento tres generaciones de sistemas de vigilancia:

1. Primera generación:

   Consistían en cámaras de circuito cerrado analógicas que transmitían la señal

de video en blanco y negro a través de un cable coaxial que se conectaba a un solo monitor. Entonces si tenías 10 cámaras, necesitabas 10 cables conectados a 10 monitores distintos. Un operador debía estar observando constantemente todos los monitores.

Con el transcurrir del tiempo, fueron introducidos los conmutadores, que eran dispositivos capaces de alternar la señal de video para transmitirla a un solo monitor

Esta tecnología, era incapaz de almacenar las imágenes, hasta la llegada de los VCRs (Video Camera Recorder) que grababan las imágenes en unidades de cinta (casetes). Los VCRs trajeron muchas ventajas a los sistemas de vigilancia pero aún así, la calidad de imagen almacenada era muy pobre, y con el tiempo tendía a estropearse. Con la llegada de la era digital, surgieron los dispositivos capaces de combinar y procesa múltiples señales de video almacenándolas en medios digitales: DVR (Digital Video Recorder). Los DVRs, que aún son utilizados en la actualidad, permiten la utilización de cámaras digitales para la captura de video, además de recibir varias señales de manera simultánea, también almacena las secuencias de video en formatos digitales, y además permite el acceso a través de redes IP.

2. Segunda generación:

   Con los sistemas de vigilancia computarizada, la utilización de diferentes sensores e inteligencia artificial deriva un concepto denominado Vigilancia Inteligente. Éste concepto, refiere a todos los componentes digitales que en su conjunto forman un sistema de vigilancia integral que responde a los estímulos captados por dichos componentes, prácticamente automáticos sin intervención humana.

   Ya no se trata de simples dispositivos analógicos o de cámaras antiguas conectadas a pantallas de rayos catódicos, sino que éstos sistemas utilizan dispositivos digitales y algoritmos computacionales para la extracción de datos a fin de detectar los estímulos que se producen en las señales transmitidas por los sensores o cámaras.

   La idea de éstos avances, es disminuir el trabajo realizado por agentes humanos, capaces de sufrir fatiga o cansancio, y sustituirlos por sistemas computacionales inteligentes que analicen en tiempo real los eventos captados por los distintos sensores y cámaras.

3. Tercera generación:

   Los sistemas de vigilancia de tercera generación solo difieren de los de segunda en el modo en que se procesan los datos: es una red de proceso distribuido con múltiples cámaras/sensores. Esto ofrece robustez, ya que si una cámara/sensor deja de funcionar, el sistema sigue funcionando con los demás.

2.1.2. Clasificación de los sistemas de vigilancia

No todos los sistemas de vigilancia utilizan los mismos recursos y enfoques, la tecnología va mejorando los dispositivos con el transcurrir del tiempo, la integración de componentes digitales inteligentes acaparan los procesos de vigilancia. Y es importante conocer adecuadamente cada una de las clases de sistemas de vigilancia para entender la razón de éste trabajo.

Dentro de la clasificación de sistemas de visión para vigilancia, R. Dautov et al [2]. cita cuatro tipos distintos de sistemas de visión:

- Sistemas de visión integrados: incluyen cámaras independientes, que realizan ASIP a bordo o en una unidad externa, como una computadora incorporada (ej., Raspberry Pi).

- Los sistemas de visión basados en PC: cámaras inteligentes que consisten en una cámara de video y una computadora realizando ASIP.

- Los sistemas de visión basados en red: compuestas de múltiples cámaras interconectadas (ej., sistemas de vigilancia CCTV).

- Los sistemas de visión híbrida: cámaras inteligentes, que pueden depender de la participación humana para proporcionar datos de alta precisión.

2.1.3. Reconocimiento de actividad humana

La detección de actividad humana, en inglés *Human Activity Recognition* (HAR), consiste en interpretar la postura (pose) y movimientos del ser humano a través de sensores (como cámaras) y determinar la actividad o acción que está realizando el sujeto en cuestión.

Forma parte de una de las categorías de IA con mas enfoque en los últimos años, ya que los campos de aplicación abarcan desde la seguridad en circuitos cerrados de televisión hasta hospitales y centros de rehabilitación.

Existen numerosos trabajos realizados a lo largo del mundo que intentan, mediante la pose humana, interpretar el tipo de actividad que realiza una persona en observación.

En el artículo de O.C.Ann y L.B.Theng [15] se citan algunas estructuras de detección de actividad humana utilizando cámaras infrarrojas o RGB, además de sensores de proximidad para mejorar la precisión en profundidad, ventajas y desventajas de cada método.

Capítulo 3

Representación de poses

3.1. Postura y Pose humanas

La postura es la forma natural que se establece con la disposición de las extremidades y el tronco del cuerpo humano. La pose, es la postura que sugiere la misma forma pero se dice que no es natural, es la postura de disposición artificial, buscada para cierto fin.

A través de la pose, el ser humano expresa una idea, una acción y hasta un mensaje reflejados en la forma dispuesta de todas las partes del cuerpo, y los ángulos formados entre sí.

Se puede establecer entonces, a través de la pose, qué tipo actividad está realizando una persona humana con solo observarla en una imágen, por ejemplo.

3.1.1. Elementos que conforman la pose

Se utiliza el término pose, para referirse a la postura buscada de manera artificial para denotar una acción. Analizando el cuerpo humano y todos los movimientos realizables por el mismo, podemos citar los elementos mas importantes del cuerpo que son tomados en cuenta para establecer una pose:

1. Tronco del cuerpo (torso): pecho y caderas
2. Brazos y antebrazos: manos, codos y hombros
3. Muslos y pantorrillas: pies, rodillas y caderas

Figura 3.1: Ejemplo de representación de una pose en un niño donde se dibujan las líneas sobre los elementos mas importantes que la conforman. Fuente: pixabay.com

En la figura 3.1, se muestra a un niño jugando al fútbol. La pose establecida por el cuerpo del niño, sugiere una acción en concreto. Es posible determinar la acción ejecutada por el niño solamente observando la pose expresada en su cuerpo.

3.1.2. Poses compatibles con amenazas y agresiones

La configuración de la disposición de las extremidades, y su relación con el tronco del cuerpo humano, puede definir qué pose forma, y en la mayoría de los casos, la acción que se está ejecutando.

Si bien, determinadas poses pueden sugerir que el sujeto observado pueda estar realizando múltiples acciones, existen determinadas poses que son mayormente compatibles para acciones concretas: las amenazas y agresiones humanas.

Por amenazas y agresiones humanas, se entiende como la utilización de extremidades del cuerpo para proferir amenazas y agresiones físicas a cualquier objeto presente en su entorno, o la disposición de las extremidades de manera que el sujeto pueda estar utilizando un arma para efectuar una agresión o amenaza.

Es posible que la agresión no sea efectuada hacia otra persona, o no se pueda visualizar qué o quién recibe la agresión, sin embargo se puede agredir a un objeto que pueda contener a un ser vivo dentro del mismo, ejemplo: una persona puede estar golpeando un auto, que en su interior contiene un niño (no visible para el observador).

Ejemplos de poses que puedan ser consideradas como poses compatibles con agresión:

1. Brazos levantados formando ángulos de entre 50 y 120 grados con el tronco (ej. una persona apuntando un arma de fuego).
2. Brazos levantados formando ángulos superiores a los 170 grados con el tronco (ej. una persona propinando golpes con los brazos)
3. Piernas levantadas formando ángulos de entre 250 y 80 grados con el tronco (ej. una persona propinando golpes con las piernas)
4. Pueden ocurrir combinaciones de las anteriores.
5. Otras poses.

Figura 3.2: Pose humana compatible con amenaza (uso de armas). Fuente: pixabay.com

3.1.3. Métodos de extracción de poses

Segun Doménech en [16] existen por lo menos tres categorías de métodos de extracción de poses humanas de acuerdo a como se interpreta la estructura del cuerpo:

1. Métodos con enfoque generativo: se utiliza un modelo del cuerpo conocido con anticipación, y éstos representan el modelo a través del conjunto de partes del cuerpo, unidas por restricciones impuestas a las articulaciones en la estructura del esqueleto, ver Amin et al.

2. Métodos con enfoque discriminativo: en éstos métodos no se conoce con anticipación el modelo del cuerpo, sino que se utilizan algoritmos que 'aprenden' a mapear la relación entre las imágenes y las poses. Se utiliza aprendizaje supervisado o también se puede buscar poses por similitud a partir de cierto número de candidatos.

3. Métodos híbridos: combinan los métodos generativos y discriminativos.

Capítulo 4

Inteligencia Artificial

4.1. Conceptos de Machine Learning

Se conoce como Machine Learning (en inglés), a la ciencia relacionada con Inteligencia Artificial en la cual se pueden configurar modelos matemáticos y probabilísticos para responder a situaciones de la vida real. Básicamente, se trata de entrenar algoritmos para ofrecer respuestas, esperadas o no, bajo ciertas condiciones de campo. Su traducción literal se refiere al proceso de 'enseñar' a una máquina a responder a situaciones de manera inteligente, sin el uso de condicionales.

4.1.1. Aprendizaje y entrenamiento

El aprendizaje se da solamente en ciertos seres vivos del planeta, por lo que el concepto era ajeno a las máquinas, hasta ahora: consiste en incorporar conocimientos de situaciones repetitivas para poder responder con una acción o simplemente establecer un recuerdo del mismo.

En el ámbito de la computación, se adoptó el concepto de aprendizaje y entrenamiento al proceso de preparar y parametrizar, con datos de entrada, un modelo matemático o algoritmo para que pueda responder, con datos de salida, de la misma manera que lo haría un ser vivo capaz de aprender.

Para entender el concepto, pensemos en los datos de entrada como estímulos para el aprendizaje, donde ciertos pesos (parámetros) influyen en la importancia de cada uno, y donde se genera un valor de salida aprendido.

En el entrenamiento de un modelo matemático, se parametriza el mismo con

estos estímulos y los pesos para cada estímulo, ajustando los mismos a modo de que el resultado se aproxime a un valor esperado.

Existen numerosos métodos que utilizan funciones para aproximar estos resultados (función costo, descenso de gradiente, etc), pero no se profundizará sobre los mismos.

A partir de este punto, nos referiremos a los datos de entrada para el proceso de aprendizaje, como Input Dataset -o Features-, y la obtención de resultados de salida -o simplemente Output-.

Aprendizaje supervisado

Aprendizaje supervisado es el proceso de entrenar un algoritmo con datos de entrada estructurados, y en donde se conoce o se espera algún tipo de dato de salida previamente conocido.

En el aprendizaje supervisado, conocemos acerca de la naturaleza de los datos de entrada, y esperamos datos de salida del mismo modo. Por ejemplo, podemos estimar el costo de una casa a partir de ciertos datos conocidos, como el área, cantidad de pisos, ubicación, etc.

Entonces tenemos datos precisos, catalogados sobre características de la casa, y además esperamos un valor costo, o sea, conocemos la naturaleza del resultado.

Aprendizaje no supervisado

Por el otro lado en el aprendizaje no supervisado, no conocemos la clasificación de los datos de entrada ni de salida. Los datos de entrada no poseen una estructura definida, no están catalogados y no sabemos acerca de los datos de salida esperados. Datos de entrada para el aprendizaje no supervisado podrían ser: datos de audio, imágenes o texto.

En el aprendizaje no supervisado priman los problemas de clasificación de patrones como la clasificación de textos extensos de acuerdo a un tema en particular.

4.2. Métodos de clasificación

A continuación, un breve resumen acerca de algunos modelos probabilísticos utilizados en Machine Learning, para clasificar datos. No se profundizarán conceptos sobre todos los modelos, solamente aquellos que sirvieron para la realización de este trabajo, para ahondar en detalles consulte material especializado.

S. Bailey et al [8] describe algunos métodos principales de clasificadores, también podemos agregar los citados por N. S. Kamarudin et al[7], y los métodos de regresión de Andrew Ng y su curso de Machine Learning [10].

4.2.1. Regresión Lineal

La regresión lineal es un modelo matemático utilizado para aproximar la dependencia de dos variables, se utiliza como clasificador binario: solo devuelve dos posibles datos de salida. Este tipo de clasificador se utiliza para determinar si un objeto es o no es de cierto tipo.

Representación de hipótesis: $h(x) = b + b_1x_1 + b_1x_1.... + b_nx_n$

Donde b son los pesos de cada entrada, también llamados parámetros (se utilizan para modificar la importancia de cada variable), x son los features, e y es el resultado devuelto/esperado. Fíjese que la función hipótesis tiene la forma de una ecuación de recta, de ahí su nombre.

Si se dibujaran los datos de salida del clasificador en un plano cartesiano, podríamos separar los valores por medio de una línea recta -denominada límite de decisión-.

Una unidad de datos de entrenamiento puede identificarse como *(x, y)* donde x representa los datos de entrada, e y representa el de salida esperado. Además, para la regresión lineal, se puede utilizar una variable de entrada, o múltiples variables de entrada, siendo m la cantidad de variables de entradas -cantidad de features- y n la cantidad de conjuntos de entrenamiento.

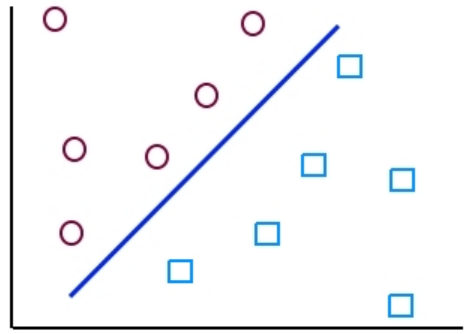

Figura 4.1: Regresión lineal, datos de salida en un plano cartesiano con el límite de decisión

4.2.2. Regresión Logística

La regresión logística también se utiliza como clasificador binario, pero se diferencia de la Regresión Lineal en que los datos de salida, en el plano cartesiano, están separados por una línea curva, o circular: esto es porque el límite de decisión esta representado por una ecuación cuadrática. La representación de su hipótesis utiliza la función denominada función logística o 'sigmoidea'.

También, como la regresión lineal, la logística puede utilizar una o varias variables de entrada denotadas por la letra x, y la salida por la letra y, además de m y n que continúan teniendo el mismo significado.

Representación de hipótesis: $h(x) = g(O^t x)$ donde $z = O^t x$ entonces

$$g(z) = \frac{1}{1 + e^{-z}} \tag{4.1}$$

Donde z es la función cuadrática con los valores de los datos de entradas.

Ejemplo: $z = O + O_1 x_1^2 + O_2 x_2^2$ es la ecuación de la circunferencia, que puede ser utilizado como límite de decisión.

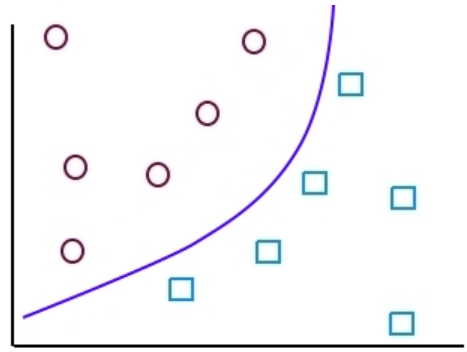

Figura 4.2: Regresión logística, datos de salida en un plano cartesiano con el límite de decisión

4.2.3. Máquinas de Vectores Soporte

En inglés '*Support Vector Machines*' (SVM) es un método de clasificación que utiliza funciones matemáticas: dado un conjunto de puntos donde cada uno pertenece a una de dos posibles categorías, se construye un modelo capaz de predecir si un punto nuevo pertenece a una de las dos categorías.

Pertenece a la categoría de los clasificadores lineales, puesto que inducen separadores lineales o hiperplanos.

Básicamente, se busca una linea que separe los puntos de entrenamiento en dos áreas diferentes en un gráfico. El algoritmo SVM halla la ecuación de recta óptima.

Mediante una función, se intenta determinar si un dato de entrada (nuevo punto en el gráfico) pertenece o no a uno de los dos grupos establecidos mediante la separación realizada.

Así, la separación de datos en el plano cartesiano se efectúa mediante un hiperplano, buscando aquel que sea óptimo: esto es, el hiperplano cuya distancia sea igual a los puntos mas cercanos a la recta.

SVM Multiclase

Recordemos que un clasificador lineal solo puede 'diferenciar' una clase de objeto, del resto de objetos. La separación por medio de una función lineal muchas veces no

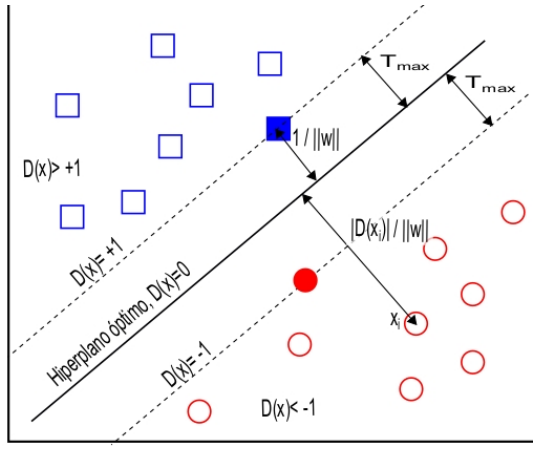

Figura 4.3: SVM con hiperplano óptimo. Fuente: Tutorial sobre Máquinas de Vectores Soporte (SVM) [13]

es aplicable a ciertos problemas, o las clases no son perfectamente separables.

Esto se soluciona con funciones kernel, que proyectan la información del universo de datos a una dimensión extra. Para darnos una idea, pensemos primero en el conjunto de datos en un plano cartesiano, que recordemos es un gráfico de dos dimensiones (x e y) con líneas abscisas y una línea recta divide los puntos que representan los datos. La función kernel permite agregar una dimensión extra, como un cubo de tres dimensiones donde ciertos datos tendrían valores en otra dimensión, como si el plano tuviese profundidad.

Algunos tipos de funciones kernel son:

1. Polinomial.
2. Perceptrón.
3. Gaussiana.
4. Sigmoidea.

Con las funciones kernel, se podría por ejemplo, mapear los valores de ciertos puntos, que bajo un plano de dos dimensiones no son perfectamente separables mediante una linea recta, haciendo de ese modo que los nuevos puntos sean separables.

4.2.4. Redes Neuronales Artificiales

En biología, una neurona es un tipo de célula del sistema nervioso que es capaz de comunicar pulsos eléctricos a otras células. El cerebro humano esta compuesto por miles de millones de estas células, y en la combinación de estas es donde se producen los procesos neuronales que permiten controlar partes del cuerpo, aprender y sentir.

La estructura básica de una neurona consiste en:

- Dendritas: conexiones que permiten la comunicación de otras neuronas con la actual.
- Núcleo: donde se realiza el proceso celular.
- Axón: comunica la señal generada por el núcleo a otra neurona en la red.

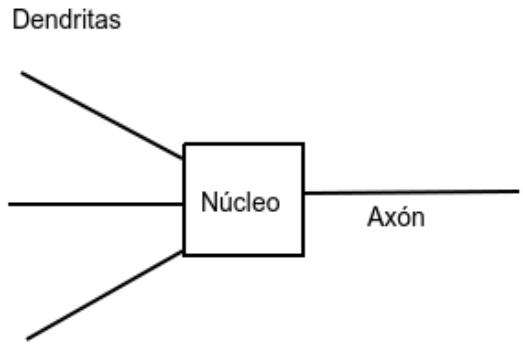

Figura 4.4: Estructura superficial de una neurona natural.

La utilización de redes neuronales para clasificación es uno de los métodos mas utilizados actualmente. Éstas simulan la interconexión de las neuronas cerebrales.

En las redes neuronales artificiales se asignan un conjunto de variables de entrada a diferentes resultados de salida a través de nodos intermedios por los cuales van circulando los datos. Los nodos intermedios pueden ser muchos, o simplemente pueden no existir, dependiendo de las necesidades del modelo.

Arquitectura básica de una red neuronal artificial

Ahora que conocemos el concepto básico de una neurona, imaginemos tener varias neuronas interconectadas entre sí, donde, teniendo datos de entrada (estímulos), estos de propagan a través de la estructura hasta llegar al final, devolviendo un valor de salida

Entre cada conexión de neuronas existe un parámetro denominado *peso* de la conexión, que es nada mas que un parámetro modificable y adaptable al proceso de aprendizaje. La modificación de los pesos, es lo que se denomina aprendizaje, y existen diferentes métodos de modificar los pesos a medida que se va iterando la red.

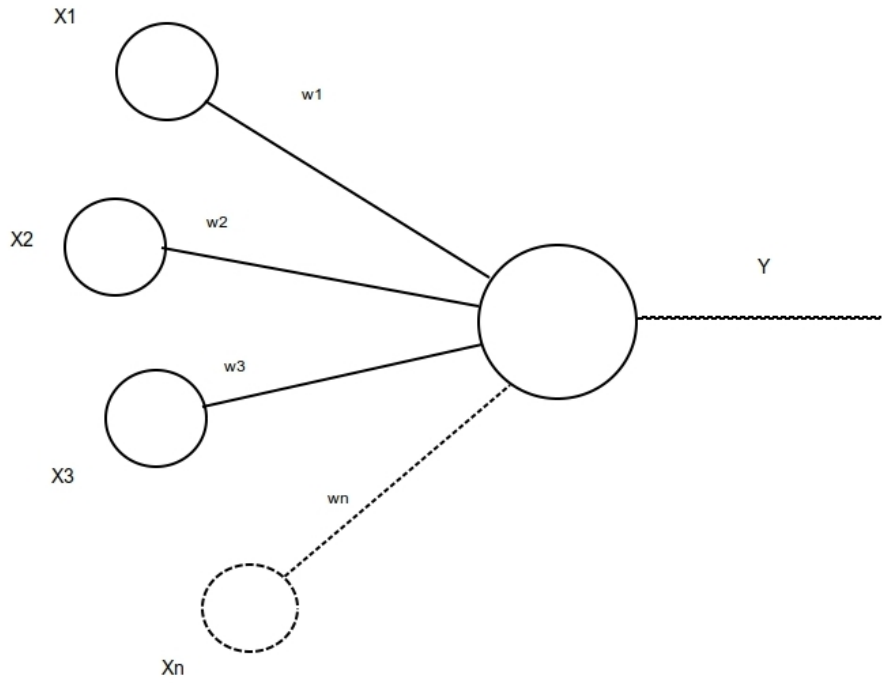

Figura 4.5: Estructura de una red neuronal simple.

En la figura 4.5, se puede observar una estructura neuronal simple, de una capa de neuronas de entrada, y otra capa de una sola neurona de salida, donde $\mathbf{X}$ son las entradas para las neuronas (los estímulos), $\mathbf{w}$ son los pesos (parámetros adaptables) que existen entre las neuronas de entrada y la neurona de salida e $\mathbf{Y}$ es el resultado del proceso.

Arquitectura multicapa

Otros modelos de redes neuronales pueden agregar mas capas de neuronas entre la capa de entrada y la de salida, las cuales son denominadas *capas ocultas*. Una red neuronal con varias capas ocultas se denomina multicapa. En la figura 4.5, no existen capas intermedias ocultas.

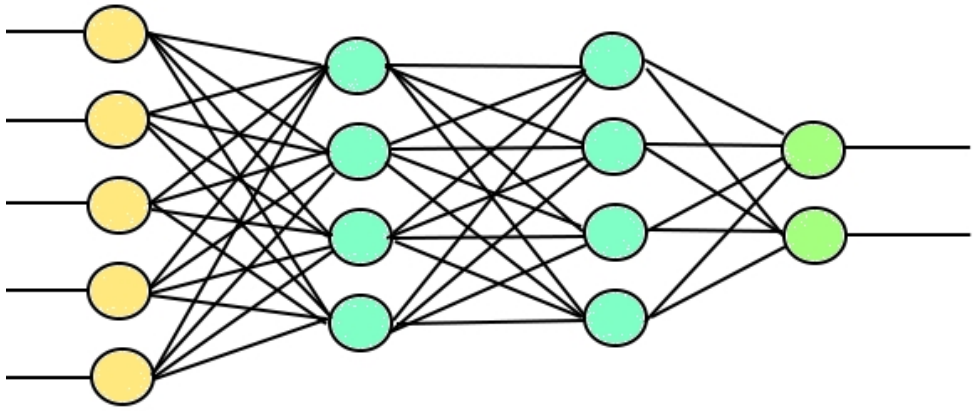

Figura 4.6: Estructura de una red neuronal con múltiples capas.

En la figura 4.6, se puede observar una estructura de red neuronal con dos capas ocultas (intermedias). La capa de entrada (color naranja), con cinco neuronas, las dos capas ocultas (color celeste) con cuatro neuronas cada una, y la capa de salida (color verde) con dos neuronas.

Perceptrón

Es un modelo de red neuronal que permite diferenciar solo tipos de clases de forma lineal. Fué uno de los primeros modelos de redes artificiales ideados. Originalmente el perceptrón fue creado para resolver problemas de visión por computadoras, tratando de imitar cómo el cerebro aprende patrones a partir de las imágenes captadas por los ojos.

El problema del perceptrón, es que solamente podía clasificar patrones linealmente separables, y no era posible ajustar los pesos de todas las capas, ya que no se diseñó un algoritmo para tal fin.

Con el correr de los años, el perceptrón quedó olvidado ya que sus limitaciones lo hacían inservible para problemas mas complejos. Durante mas de dos décadas, no hubieron avances sobre redes neuronales hasta la llegada del algoritmo de *backpropagation.*

Backpropagation

Del inglés *back propagation*, o propagación hacia atrás, es un algoritmo de aprendizaje supervisado (para el ajuste de pesos) que, dado un valor de salida de la red neuronal, calcula la diferencia con respecto a la salida deseada y ajusta los pesos de las conexiones de las capas de adelante hacia atrás.

El algoritmo de backpropagation empieza cuando ha terminado la propagación hacia adelante, es decir, cuando se ha conseguido valores en la capa de salida. Una vez obtenidos los valores, se comparan los mismos con los valores de salida deseados -ésto se denomina *cálculo de error*-, y un algoritmo recursivo va ajustando los pesos en las capas anteriores hasta llegar a la capa de entrada donde termina el backpropagation.

No se profundizará sobre el algoritmo matemático del backpropagation ya que no es la finalidad de este trabajo explicar detalladamente sobre estos conceptos, solo dar una breve descripción de su funcionamiento. Consulte otros artículos mas especializados para mas detalles.

Funciones de entrada

Hasta ahora nos hemos enfocado en la estructura de una red neuronal y el modo en que se ajustan los pesos de acuerdo a una salida pero no mencionamos el modo en que se obtiene la misma.

Cada capa de la red neuronal esta compuesta por neuronas interconectadas con las capas adyacentes, como ya sabemos, pero además, cada neurona de cada capa implementa una función de acuerdo al tipo de capa (entrada, oculta o salida). Esta función es la encargada de procesar el valor de entrada y devolver un valor de salida, o estado de la neurona.

La capa de entrada implementa lo que se denominan funciones de entrada, y solamente existen ciertas funciones que pueden aplicarse a las neuronas de la capa de entrada.

De Matich [11] podemos entresacar los tipos de funciones de entrada mas comunes:

1. Sumatoria de entradas pesadas: suma de todos los valores de entrada de la neurona, multiplicados por sus correspondientes pesos.

$$\sum (n_i W_i)$$

2. Productoria de entradas pesadas: producto de todos los valores de entrada de la neurona, multiplicados por sus correspondientes pesos.

$$\prod (n_i W_i)$$

3. Máximo de las entradas pesadas: toma el producto mas alto de la multiplicación de valores y sus pesos correspondientes.

$$Max(n_i W_i)$$

Funciones de activación

Ahora, una capa oculta también puede tener una o mas neuronas interconectadas con las capas adyacentes: la neurona recibe datos de la capa anterior, procesa los datos, y envía su estado a la siguiente capa.

Así como las funciones de entrada, las funciones de activación solamente son aplicables a las capas intermedias ocultas, aunque existen solo un par de funciones aplicables a mas de una capa.

En biología, una neurona puede estar activa o inactiva, en las redes artificiales sucede lo mismo: una neurona puede estar activa o inactiva de acuerdo al valor devuelto por la función de activación de la misma.

Una función de activación recibe datos de entrada y genera un resultado de activación. Generalmente pueden tomar valores de 0 y 1, o -1 y 1 (desactivado y activado). Existen funciones que pueden devolver mas estados.

De Enyinna et al[12] podemos citar tres de los tipos de funciones de activación mas conocidos:

1. Función sigmoidea: es una función de activación no lineal, conocido también como función logística. Su uso mayor se da en redes neuronales aunque se ha comprobado que tiene ciertos problemas con backpropagation, y otras funciones mas modernas se utilizan en su lugar. Esta función tiene algunas variantes que no se detallarán en este trabajo. Su ecuación es:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{4.2}$$

2. Función tangente hiperbólica: es una función de activación utilizada muchas veces en lugar de la función sigmoidea ya que ofrece mejores velocidades de entrenamiento en redes neuronales multicapa. Su ecuación:

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{4.3}$$

3. ReLU: *Rectified linear unit*, viene del inglés, o Unidad linear rectificada es una función de activación mas moderna y que se ha comprobado ofrece velocidades de entrenamiento muy superiores a las anteriores.

   Representa algo parecido a una función linear por lo que resulta rápido para procesos de entrenamiento intensivos. Actualmente es la función de activación mas utilizada ya que ha demostrado una eficiencia de velocidad y generalización (aplicable a múltiples modelos) bastante altos.

   ReLU aplica un umbral simple a los valores de entrada separando los mismos de acuerdo a una regla establecida por su ecuación:

$$f(x) = Max(0, x)$$

   donde:

$$x,\ si\ x >= 0$$

$$0,\ si\ x < 0$$

ReLU además cuenta con otras variantes que aplican distintos tipos de umbrales pero no se detallarán en el presente trabajo.

Funciones de salida

La capa de salida también implementa ciertas funciones, citando:

1. Función lineal: También llamada función identidad.
2. Función sigmoidea: ya detallada anteriormente.

Tasa de aprendizaje

En inglés *learning rate*, es un parámetro de redes neuronales que determina la cantidad de ciclos de entrenamiento en la que la red actualizará sus pesos, así, a menor tasa los parámetros se actualizarán una mayor cantidad de veces y a tasa learning rate superior, la red actualizará sus pesos con menor frecuencia.

Optimizadores de redes neuronales artificiales

Los optimizadores son algoritmos utilizados para minimizar la función objetivo, que es básicamente una función matemática dependiente de los parámetros de aprendizaje del modelo de la red neuronal. No se explicarán detalles profundos sobre el funcionamiento de los optimizadores, consulte documentación especializada para ahondar en detalles.

Algunos de los optimizadores mas conocidos:

- Descenso de gradiente: utilizado para ajustar los pesos en los ciclos de entrenamiento.
- Adam: calcula la tasa de aprendizaje de una red manteniendo un promedio de gradientes de ciclos anteriores.
- Adagrad: simplemente la tasa de aprendizaje de una red de acuerdo a los parámetros de la misma.

Capítulo 5

Procesamiento gráfico

5.1. Visión por computadora

En los últimos años, los problemas de visión por computadora han acaparado la mayoría de los trabajos e investigaciones dada la cantidad de situaciones en las cuales es necesario utilizar ayuda no humana para resolverlas. Los sistemas computacionales han evolucionado y la inteligencia artificial ha surgido como una ciencia de estudio capaz de proveer la ayuda necesaria.

Sin embargo, un conjunto de componentes electrónicos no es capaz de entender una imagen. La representación de una imagen, en términos informáticos, es la agrupación de números distribuidos en una matriz. Éstos números representan intensidades de color y luminosidad.

5.1.1. Detección y reconocimiento de objetos en imágenes

Los sistemas de vigilancia en auge actualmente son los de segunda generación: dispositivos que capturan imágenes que son enviadas a procesadores que utilizan algoritmos computacionales para obtener la información relevante de las mismas.

Si bien existen investigaciones enfocadas a la utilización de Inteligencia Artificial para la Vigilancia Inteligente, ésto resulta por ahora una tarea difícil de implementar debido a la cantidad de combinaciones posibles de escenarios que puedan darse, la cantidad de datos de entrada y los patrones que se deben analizar en las diferentes fases. Normalmente, el proceso de identificación de objetos en imágenes se da en diferentes etapas descritas en [3] y [7], ellas son:

1. Detección de objetos.

2. Clasificación (identificación) de objetos.

3. Extracción de características (features) de objetos.

4. Análisis del comportamiento de los objetos.

Detección de objetos en imágenes

La detección de objetos en imágenes es una de las áreas de mayor investigación en visión por computadora. La tarea no resulta fácil, ya que existen numerosos elementos a tener en cuenta para la detección de objetos que sean de relevancia.

El principal obstáculo, es determinar si un conjunto de pixeles forman o no un objeto tomando en cuenta que existen innumerables combinaciones de posición, luminosidad, color y formas, como asi tambien el ruido de interferencia y el tamaño del objeto (o su lejanía de la cámara).

Enfocado a la seguridad, la detección de objetos trata de identificar solamente unas pocas categorías de objetos en imágenes, lo que lo hace un proceso mas viable: la detección de formas humanas, también denominado detección de peatones (Pedestrian detection [4]).

Técnicas de detección de objetos en imágenes

A través de los años, como consecuencia de las investigaciones en visión por computadora, han surgido diferentes enfoques en la utilización de técnicas de detección de objetos, M. Valera y S.A. Velastin [1] citan dos enfoques a lo que se pueden agregar ademas el presentado en [5] y el citado en [6]:

1. Diferencia temporal: este enfoque se basa en la comparación de un frame (imágen de video) con el frame anterior. Esta acción permite identificar objetos en movimiento cambiante dentro del conjunto de frames, aunque sugieren que es

un proceso mas lento que otros.

2. Substracción de fondo: utiliza una imagen de fondo que se compara con frames pixel a pixel para determinar los cambios ocurridos en la imagen, lo que permite obtener contornos de objetos.

3. Filtrado: este método es quizás el menos utilizado ya que las características de los objetos suelen ser variantes. Este método sugiere la detección de objetos basados en el filtrado de color del mismo: se extraen los pixeles que concuerdan con el color de un objeto previamente establecido. Como los objetos pueden variar su color, o luminosidad, este método resulta poco efectivo. Puede utilizarse en condiciones muy controladas.

4. Flujo óptico: este método es el más costoso computacionalmente hablando, ya que determina vectores de movimiento de cada uno de los pixeles para determinar el contorno de un objeto en movimiento dentro de un frame. Cada pixel en movimiento (posición inicial y posición final) determina un vector, un sentido de movimiento. Se puede tomar el conjunto de vectores de todos los pixeles para determinar la existencia de un objeto en la imágen.

Clasificación de objetos

La última etapa, luego de la detección de objetos, es la clasificación de los mismos: se debe determinar la clase de objeto detectado. Es de vital importancia poder indentificar el tipo de objeto detectado, especialmente cuando se trata de personas: necesitamos registrar y catalogar el comportamiento de objetos de tipo persona para determinar si se produce una forma compatible de agresión.

Técnicas de clasificación

Para determinar la clase de objeto extraído de una imágen, se utilizan dos tipos principales de técnicas citadas por [3]:

1. Clasificación basada en formas: es la técnica mas sencilla y consiste en, una vez identificado un objeto en una imágen, compararlo con formas de objetos existentes. Se asocia un valor numérico para identificar el grado de similitud entre las imágenes comparadas. El valor más alto asociado definirá la clase de objeto.

2. Clasificación basada en movimiento: estudia el movimiento hecho por un objeto en particular, con respecto a su forma o silueta. Se sabe por ejemplo, que el cuerpo humano va cambiando su forma -esto es, existe movimiento- a través de las imágenes. Todo lo contrario a lo que ocurre con los automóviles, que no suelen cambiar su forma.

Métodos de clasificación

Ya hemos citado las técnicas que mayormente se utilizan en la clasificación de objetos, además de que existen otras técnicas menos conocidas o que producen resultados menos certeros, ahora hay que hablar sobre los métodos de clasificación utilizados actualmente por programas computacionales para separar objetos según su clase utilizando las técnicas citadas previamente.

Aunque algunos de ellos ya fueron citados y explicados en otros capítulos de este trabajo, citaremos aquellos que fueron implementados en la clasificación de objetos en imágenes como ser: Árboles de decisiones, Support Vector Machines, Redes bayesianas y Redes Neuronales Artificiales.

5.1.2. Histograma de Gradientes Orientados

En inglés '*Histograms of Oriented Gradients*' (HOG), es una técnica de detección de objetos en imágenes, mas precisamente para la detección de humanos, según Dalal y Trigg [14]. Básicamente, se trata de un descriptor de objetos, que permite caracterizarlos en un mapa de gradientes. Se suele utilizar HOG como descriptor de objetos y SVM como clasificador para una amplia gama de herramientas de detección y clasificación.

La idea básica del HOG, es dividir una imagen en ventanas (denominadas blocks,

o *detection windows* en inglés), obteniendo los gradientes que se forman dentro de cada ventana, así, una imagen puede contener decenas de pequeñas ventanas con sus gradientes. El gradiente contiene el vector dirección, lo que permite caracterizar la zona dentro de la ventana.

Un gradiente es básicamente, un vector que indica la dirección donde ocurre el cambio de luminosidad de oscuro a claro. Para un gradiente dentro de una ventana, se obtiene la representación a través de una línea que indica la separación de luminosidad ocurrida dentro de la ventana, y la dirección en la que ocurre el cambio. De esta manera, con el conjunto total de gradientes de la figura, se logra obtener una representación aproximada del contorno del objeto.

Además, una ventana puede contener diferentes cantidades de gradientes, de acuerdo a la cantidad de líneas tangentes a un punto. Sin embargo, las herramientas de extracción de HOGs tienen parámetros para la configuración de la cantidad de gradientes que se desean obtener, como también el tamaño de las ventanas a procesar (generalmente en pixeles). Cuanto mas gradientes, más generalizado es el caso, y por el contrario, cuanto menos gradientes se obtengan, mas especializado será.

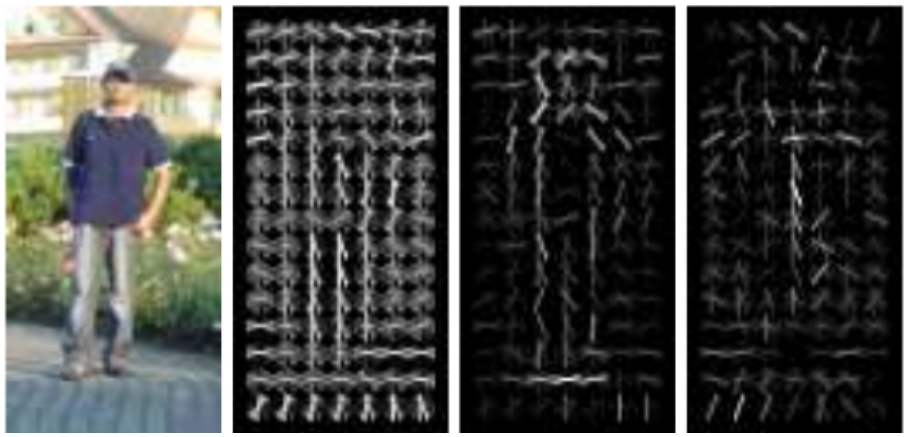

Figura 5.1: Imágen de un hombre, y los gradientes obtenidos de la misma. Fuente Dalal y Trigg [14]

5.2. Herramientas de visión por computadora

5.2.1. OpenCV

OpenCV (Open Computer Vision, en inglés) es una librería escrita con participación de investigadores de Intel Corp. para el procesamiento de imágenes por computadora según lo descrito en la referencia oficial [9].

Provee una serie de procedimientos y funciones estándar para la manipulación de imágenes en, hasta la fecha de este trabajo, tres lenguajes de programación de computadoras: C++, Python y Java. Provee herramientas preestablecidas y preconfiguradas para que la tarea del programador sea mas fácil y rápida.

OpenCV se escribió con la finalidad de ser eficiente y aprovechar las instrucciones de bajo nivel de los procesadores Intel, que tienen una cuota muy alta de ventas en el mercado mundial. Esto, sumado a que fue escrito en C y C++, hace que el código se ejecute mucho mas rápido que otras librerias de procesamiento de imágenes.

Además, la librería ofrece bibliotecas de funciones y procedimientos parametrizables para resolver problemas de Inteligencia Artificial. Los componentes prefabricados con un alto nivel de abstracción hacen posible al programador implementar, por ejemplo, un clasificador de objetos en unas pocas líneas de instrucciones Python.

Módulos de Inteligencia Artificial disponibles en OpenCV:

- Clasificadores en cascada
- Clasificadores Bayes
- Support Vector Machines
- Decision Trees
- Redes neuronales

En este trabajo, no se explicarán ni detallarán sobre los módulos de tratamiento de imágenes de OpenCV, ya que no se utilizaron durante el desarrollo. En su lugar, se

abordarán conceptos y detalles sobre algunas de las librerías OpenCV de Inteligencia Artificial que fueron utilizadas. Para ahondar en herramientas para el tratamiento de imágenes, consulte la documentación oficial.

Clasificadores en Cascada

Llamados en inglés '*cascade of boosted classifiers working with haar-like features*', es un módulo OpenCV que implementa una serie de clasificadores que generalmente son utilizados para detección de objetos en imágenes.

Una serie de clasificadores simples están dispuestos en una estructura de cascada de manera que puedan funcionar más rápidamente que un clasificador único y pesado. La salida de unos clasificadores pasan a ser las entradas de otros, formando así como una especie de puertas lógicas donde se cumplen condiciones, para tener un resultado final.

La palabra *boosted* además, hace referencia a que estos clasificadores implementan técnicas de Machine Learning que aceleran la ejecución de manera eficiente, actualmente se implementan cuatro técnicas conocidas como: *Discrete Adaboost, Real Adaboost, Gentle Adaboost y Logitboost.* No se detallarán sobre estas técnicas en el presente documento.

Las entradas de datos para estos clasificadores, se conocen en inglés como '*Haar-like features*'. Cada clasificador simple que compone la cascada recibe solo un tipo de entrada (feature), que en combinación con los demás clasificadores, pueden determinar la clase de objeto en proceso. La entrada es un conjunto de pixeles con una distribución en particular, como se detalla en la siguiente imagen.

Redes Neuronales Profundas

En inglés *Deep Neural Networks*, y se las conocen por sus siglas como DNN. Son redes neuronales multicapa donde existen mas de dos capas ocultas intermedias, de ahí el adjetivo de 'profundas'.

Existen DNNs que pueden llegar a implementar 256 capas intermedias con hasta

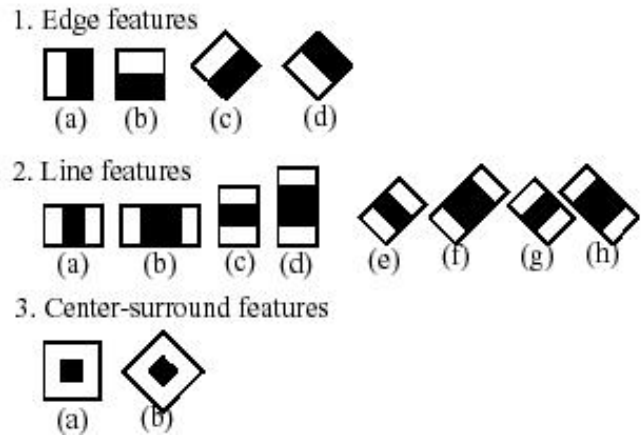

Figura 5.2: Entradas (Features) para los distintos clasificadores en cascada Fuente: OpenCV Reference Manual.

128 entradas y el mismo número de neuronas, y es solo un ejemplo básico. Estas redes neuronales tienen modelos extremadamente enormes y requieren procesadores potentes, sin mencionar que los ciclos de entrenamiento pueden durar un par de días.

La ventaja de utilizar DNNs es que permite resolver problemas complejos no lineales, aunque la exactitud muchas veces depende de cuan entrenado esté el modelo.

En visión por computadora, se utilizan DNNs para identificar múltiples objetos (ej. YOLO), también o para extraer poses humanas en imágenes (OpenPose, AlphaPose, etc).

Redes Neuronales Convolucionales

En inglés *Convolutional Neural Networks*, DNN por sus siglas, es otro tipo de redes neuronales cuyo modelo es parecido al del perceptrón multicapa. Este tipo de redes es utilizado para la segmentación y clasificación de imágenes debido a que, por la naturaleza de su modelo, es aplicable en mayor grado a los problemas de visión por computadoras.

En las redes convolucionales, se aplican ciertos filtros a las imágenes mediante neuronas convolucionales resumiendo así ciertas porciones de la imágen y obteniendo

una representación mas detallada de la misma, este proceso de resumir porciones se denomina reducción de muestreo, y en inglés se lo conoce como *pooling* (pooling layer, es la capa encargada de resumir porciones de la imágen).

La idea detrás de las redes convolucionales, según Y.LeCun et all en [18] es simular el comportamiento de los receptores de la corteza visual en los cerebros biológicos mediante modelos de neuronas artificiales.

Capítulo 6

Desarrollo de software

6.1. Pruebas de estimación de poses humanas

Dentro de los objetivos específicos de este trabajo, citábamos el de obtener patrones de poses humanas captadas en las imágenes secuenciales.

La estimación de pose es, en términos simples, la extracción de datos de los puntos que conforman ciertas partes del cuerpo humano, con las cuales podemos diagramar la ubicación de los mismos, y su dependencia con las demás partes, logrando así graficar la estructura de una pose humana.

Otros trabajos lo denominan 'esqueletización', u 'obtener el esqueleto', pero la idea es siempre la misma: obtener un mapa de la disposición de las extremidades del cuerpo humano, como si se tratase del mismo esqueleto.

Diferentes trabajos se han realizado a través de los años, en general, el de obtener una pose de cualquier objeto, formando un esqueleto de los puntos mas representativos de la forma del objeto en cuestión.

6.1.1. Prueba 1: Clasificador SVM-HOG

Justificación

Las herramientas actuales de extracción de poses requieren grandes prestaciones de hardware para funcionar en tiempo real y sin retrasos importantes: en la mayoría de las páginas consultadas se recomendaban procesadores de 8 núcleos, tarjetas

gráficas (GPU) nVidia para el proceso en paralelo mediante la tecnología CUDA de nVidia que permite la ejecución en paralelo utilizando los procesadores gráficos de éstas tarjetas.

Debido a que el desarrollo y ejecución del software se realizó sobre una computadora de dos núcleos, sin tarjetas gráficas nVidia que permitan el multiprocesamiento en paralelo se optó primero por crear un extractor de pose basado en un clasificador SVM-HOG.

La idea era sencillamente desarrollar un extractor de poses capaz de ejecutarse en máquinas de bajo rendimiento.

Teoría

Ya mencionamos en capítulos anteriores el significado y funcionamiento de los descriptores HOG y de los clasificadores SVM, que permiten la detección de objetos en máquinas de gama media, sin la necesidad de contar con unidades de procesamiento de alto rendimiento.

La idea del extractor de poses era, idealmente, desarrollar un detector de objetos personalizado, entrenado con imágenes de cuerpos humanos para luego determinar la ubicación de los miembros del cuerpo a fin de generar una pose asociada. Una vez entrenado el detector de objetos personalizado, éste sería capaz de detectar objetos (humanos) en imágenes y devolver cierta información de las anotaciones enlazadas al tipo detectado.

Se estableció que cada clase de objetos, fuera nada mas que una persona en una configuración de pose distinta a las anteriores, cada forma humana sería una clase de objetos diferente, aprovechando el término 'multiclase'. Es decir, no se detectarían diferentes tipos de objetos conceptualmente hablando, sino que solamente se detectarían humanos en situaciones distintas según su pose.

Los siguientes pasos fueron llevados a cabo para obtener un extractor de poses personalizado:

1. Recolección de imágenes del cuerpo humano.

2. Categorización de acuerdo a, si es parte superior del cuerpo (tronco, hombros y brazos), o parte inferior del cuerpo (caderas, piernas y pies).
3. Generación de anotaciones con información de los puntos relevantes.
4. Implementación de múltiples detectores HOG-SVM.
5. Entrenamiento del detector para su utilización posterior.

Descripción

Recolección de imágenes del cuerpo humano

Consiste, como el título de la tarea describe, en recolectar cuadros de imágenes de diferentes secuencias que describan el cuerpo humano en alguna pose determinada.

Para la recolección de los cuadros, se utilizaron imágenes de un video descargado de Internet en el que, justamente, se podía observar a ciertos individuos con poses compatibles con amenazas o agresiones.

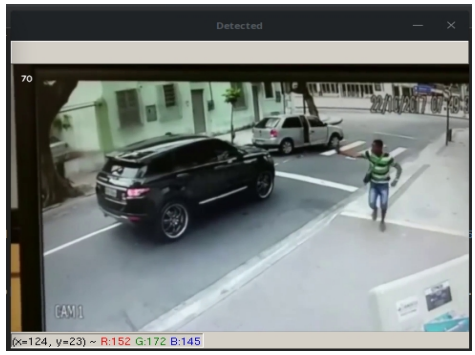

Figura 6.1: Una de las imágenes del video de pruebas donde se observa una persona posiblemente en pose de amenaza con arma de fuego corta.

Para que la tarea fuera mas sencilla, se escribió un pequeño programa en lenguaje Python capaz de extraer las imágenes separadas de una secuencia en vídeo.

Primero, se procedió a individualizar cada imágen del conjunto de imágenes del video para almacenarlas en archivos de imágen en el disco.

Categorización

Estas imágenes, catalogadas en el punto anterior, luego serían cargadas con otra herramienta escrita en Python creada para establecer la ubicación de ciertos puntos que puedan ayudar a definir la pose. Cada punto ubicado en la imágen se guardaba asociándolo con, ademas de sus coordenadas, una etiqueta descriptiva:

1. *mi md*: manos izquierda y derecha.
2. *ci cd*: codos izquierdo y derecho.
3. *p*: pecho.
4. *c*: cadera.
5. *ri rd*: rodillas izquierda y derecha.
6. *pi pd*: manos izquierda y derecha.

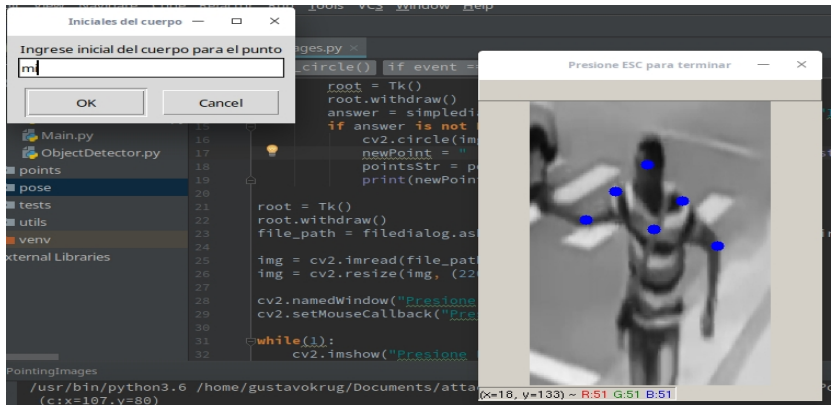

Figura 6.2: Herramienta que permite ubicar los puntos relevantes y asociarlos a una etiqueta descriptiva.

Generación de anotaciones con información de los puntos relevantes

Una vez obtenidas las coordenadas de los puntos, y la etiqueta que identifica la configuración de la disposición de estos, se procedía a almacenarlos en un archivo de texto en disco.

Cada configuración de pose distinta, con sus imágenes y anotaciones era almacenada en un directorio de disco diferente a fin de poder simular la representación de un objeto con sus anotaciones.

Implementación de múltiples detectores HOG-SVM

Existen numerosas librerías para IA escritas para el lenguaje Python, una de las más utilizadas es *dlib*. Esta herramienta fue originalmente escrita para proveer librerías Machine Learning en C++ (aunque se puede compilar su utilización para Python). Se utiliza en la industria y en el ámbito de la educación para facilitar el desarrollo de software enfocado a la inteligencia artificial (ver la página oficial http://www.dlib.net para mas información).

En la documentación de la página oficial de dlib, pueden encontrarse numerosos ejemplos de casos de uso, como el que compete a este trabajo: entrenar un detector de objetos personalizado.

dlib provee por defecto, un detector de objetos que implementa HOG-SVM prefabricado, configurable de acuerdo a las necesidades del desarrollador. Y aunque esto facilita mucho la tarea ya que no se necesita escribir desde cero, tiene como principal limitación el hecho de que el desarrollador no puede hacer uso de un clasificador multiclase. El clasificador embebido solamente puede clasificar uno-a-muchos, es decir, puede determinar, de un conjunto de objetos, aquel objeto que coincide con los datos de entrenamiento, pero no puede determinar la clase de objetos a la cual pertenecen los demás elementos.

Para solucionar esta limitación, se escribió código Python utilizando el detector embebido de dlib, de manera a simular un clasificador multiclase, creando diferentes instancias del objeto POO que respondan a una clase en particular. Es decir, ya que cada instancia del objeto dlib puede detectar y clasificar una sola clase de objetos, se crearon varias instancias, de acuerdo al número de tipos de poses obtenidas de las imágenes de entrenamiento.

Así, cada elemento de pose utilizado para el entrenamiento, era considerado una clase individual para un clasificador, de manera a que si tuviéramos cinco poses distintas, se instanciarían cinco clasificadores de dlib.

Entrenamiento del detector para su utilización posterior

El entrenamiento de clasificadores SVM, como ya se explicó en un capítulo anterior, consiste en introducir valores al clasificador de manera que pueda establecer un límite de decisión para los datos de salida, de manera que pueda responder a

un nuevo valor clasificándolo de acuerdo a si corresponde o no, a las características de los valores de entrenamiento. En las SVM, el límite de decisión es llamado un hiperplano.

De esta manera, las entradas para los datos de entrenamiento serían simplemente la disposición de gradientes obtenidos mediante HOG.

Para el entrenamiento, entonces, se utilizaron las imágenes individuales de cada 'clase' de pose humana almacenadas en carpetas, con sus anotaciones. Cada instancia del clasificador dlib responde a solo una de las clases de poses almacenadas.

Figura 6.3: Ejemplo de secuencias de imágenes utilizadas para el entrenamiento del detector de objetos dlib

Resultados

Durante la recolección de datos de entrenamiento, se clasificaron tres tipos de poses humanas: dos de la parte superior del cuerpo, y una de la parte inferior. Cada clase de pose además contaba con su archivo de anotaciones donde se especificaban las posiciones de los puntos relevantes.

Una vez desarrollado todo el proceso de entrenamiento, el detector podría ejecutarse analizando un video en cuestión, y detectando las clases de objetos persona para la cual fue entrenado previamente.

Para la implementación del detector de objetos personalizado, se escribió código una aplicación en Python que cargaba los modelos, entrenaba el clasificador dlib, y además cargaba los archivos de anotaciones asociando un clasificador a cada anotación, de modo que al ejecutar la aplicación, y se detecte una clase de objeto, se obtenían los datos de las anotaciones asociadas a la clase en cuestión.

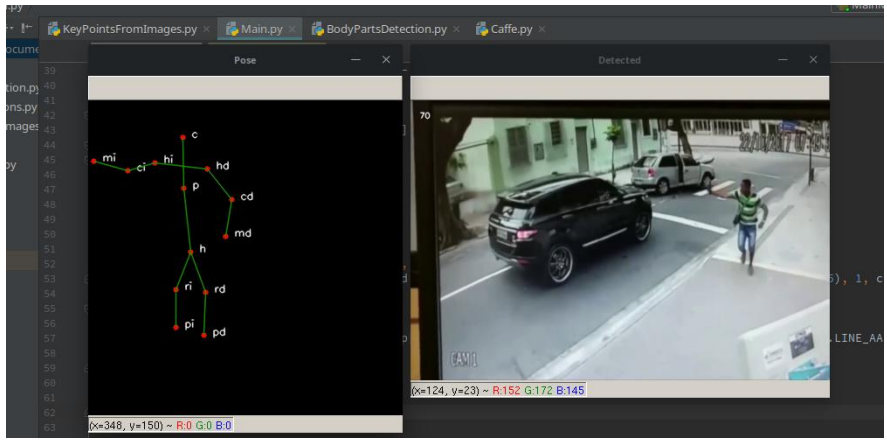

Figura 6.4: Se visualiza la pose descripta con los puntos de las anotaciones asociadas a la clase de pose (objeto persona) detectada, junto con la imágen original del video en proceso.

Los puntos, una vez diagramados en pantalla, eran unidos por líneas de acuerdo a relaciones de partes preestablecidas: hombros se unían a sus respectivos codos, codos con manos, pies con rodillas, etc.

Conclusión

El clasificador de objetos dlib entrenado con algunas decenas de imágenes resultaba bastante eficiente en términos de velocidad al ejecutarse prácticamente sin retrasos, tomando en cuenta que solamente existían tres clases de poses para las cuales fue entrenado. No se agregaron mas clases por falta de tiempo. Aunque se entiende que a mayor cantidad de instancias de un clasificador, mayor consumición de recursos de procesamiento, lo que podría ocasionar retrasos en caso de agregarse algunas decenas de clases de poses a los datos de entrenamiento.

Para verificar el grado simple de precisión del detector de poses, se utilizó el mismo para:

1. **Detectar poses en tiempo real a través de la cámara incorporada al computador:** los resultados fueron para nada precisos ya que no detectaba poses humanas teniendo en observación a al menos una persona completa.

2. **Detectar poses en videos descargados de la Internet:** se detectaban un número considerable de falsos positivos, es decir, detectaba poses humanas en donde no existían seres humanos en escena.

3. **Detectar poses en videos descargados o mediante la cámara incorporada con baja iluminación:** no se detectaban poses humanas bajo condiciones de poca luminosidad escénica en donde, precisamente, existía al menos un cuerpo humano.

Se resolvió descartar un detector de poses usando los módulos de HOG-SVM de la librería dlib debido a que se necesitarían meses de trabajo recopilando imágenes de entrenamiento mas nítidas y además calibrar los parámetros necesarios para obtener una precisión aceptable.

6.1.2. Prueba 2: AlphaPose

Descripción

AlphaPose es un proyecto de estimación precisa de poses humanas en Python que implementa algoritmos basados en el paper de H.Fang [17] descrito como Estimación Regional Multipersona.

Según la descripción encontrada en github.com, que aloja proyectos de desarrollo a nivel mundial, AlphaPose fue el primer sistema de detección de poses humanas de código abierto.

Este sistema utiliza el COCO dataset, que es un conjunto de aproximadamente 80 clases de objetos, 80.000 imágenes de entrenamiento y 40.000 imágenes de validación (ver sitio oficial `www.cocodataset.org`).

Además, puede utilizar el MPII dataset, otro conjunto extenso de imágenes que incluye aproximadamente 25.000 imágenes que contienen cerca de 40.000 anotaciones

de cuerpos humanos cubriendo 410 tipos de actividades humanas catalogadas (ver sitio oficial human-pose.mpi-inf.mpg.de).

AlphaPose implementa modelos de CNNs entrenados mediante los dos conjuntos de datos mencionados, haciendo necesario el uso de la tecnología CUDA de nVidia, ya mencionado anteriormente. Afortunadamente, AlphaPose puede correr en computadores que no cuenten con GPUs nVidia omitiendo asi el uso de CUDA, aunque esto resulta ser mas lento ya que no se cuenta con la característica de multiprocesamiento.

Para los resultados, se utilizó código de ejemplo alojado en su página oficial, que permite la ejecución de pruebas para determinar su grado de precisión y velocidad de procesamiento.

Resultados

Se utilizó código ejemplo oficial de AlphaPose que permitía la ejecución de pruebas sin la utilización de GPUs especializados para ejecución en paralelo, lo que resulta conveniente para este trabajo debido a la imposibilidad de contar con una unidad potente de procesamiento. Sin embargo, las pruebas sin GPUs sugirieron una baja tasa de procesamiento de imágenes por segundo (en inglés *Frames Per Second*, o FPS). Ésto sumado a la imposibilidad de cambiar parámetros de la red neuronal embebida, hicieron de este sistema un candidato no utilizable para el entorno con el que se cuenta en este trabajo.

La tasa de procesamiento arrojó, en un computador con dos núcleos y sin GPU dedicado, un valor aproximado de 0.066666667 FPS, o lo que es igual, un promedio de 15 segundos de procesamiento por cada imágen.

Se aclara, que para las pruebas de las distintas herramientas, se utilizó un único video descargado de Internet, a fin de tener una visión mas certera de velocidad. La utilización independiente de diferentes videos no sería apropiada para establecer un resultado coherente con el objetivo.

Conclusión

Debido a la baja tasa de procesamiento de AlphaPose sin GPU, se descartó la utilización del mismo para el presente trabajo.

Sin embargo, AlphaPose podría ser utilizado en proyectos que implementen hardware de procesamiento con mas potencia, múltiples GPUs, y 8 núcleos de procesador como mínimo.

6.1.3. Prueba 3: Modelo DNN Caffe en OpenCV

Descripción

Caffe es el acrónimo de *Convolutional Architecture for Fast Feature Embedding* en inglés, o Arquitectura convolucional para incrustación rápida de características. Es un framework para desarrollo de software enfocado a Machine Learning, desarrollado en la Universidad de Berkeley en California, Jia et all explican su desarrollo en [20].

Se trata de un conjunto de librerías C++, aunque ya existe una API para Python, según se puede leer en su página web oficial `caffe.berkeleyvision.org`, para generar modelos, entrenar algoritmos e inferir utilizando diferentes elementos propios de Machine Learning.

Existen diferentes modelos de redes neuronales generadas utilizando Caffe en Internet, es así que uno de estos modelos fue creado para la estimación de poses

humanas en imágenes.

Un modelo de red neuronal consiste en un conjunto de configuraciones preestablecidas de la estructura de una red neuronal, previamente entrenada por un desarrollador, con sus pesos, capas intermedias y las funciones de entrada, activación y salida correspondientes.

En el sitio web oficial de entrenamiento de OpenCV www.learnopencv.com, se puede encontrar un modelo Caffe de CNNs utilizado para la estimación de poses humanas, simplemente mediante el uso de una red neuronal incluida en las herramientas OpenCV.

De esta manera, es posible cargar el modelo preestablecido Caffe utilizando herramientas de redes neuronales profundas embebidas en OpenCV, evitando así, que otro desarrollador pase meses o posiblemente años recopilando información y generando una estructura de red que pueda ser usada por otra persona.

En la documentación oficial de OpenCV podemos encontrar ejemplos de cómo utilizar estos modelos pre entrenados, que, siendo implementados en unas lineas de código Python, permitieron poder utilizar el modelo para estimar poses humanas en imágenes (recordemos que un video es un conjunto de imágenes).

Así es, que se desarrollo una pequeña aplicación Python para utilizar el modelo Caffe y así, procesar las imágenes del video de pruebas.

Resultados

Ejecutar una CNN, como dijimos anteriormente, no es una tarea que requiera pocos recursos, pero a diferencia de las demás pruebas citadas anteriormente, utilizar un modelo preestablecido en OpenCV no requiere de un computador con múltiples GPUs ejecutando CUDA, aunque esto sería óptimo.

Sin embargo, implementar las herramientas de CNNs de OpenCV tiene ciertamente una ventaja significativa con respecto a las opciones anteriores: la capacidad de ajustar ciertos parámetros propios de la red, lo que puede incrementar significativamente la velocidad de procesamiento en un computador sin GPU.

Conclusión

Tomando en cuenta que el la velocidad de procesamiento utilizando Caffe es ligeramente superiores a las pruebas anteriores, se optó por utilizar el modelo para la extracción de poses: una imágen procesada en un computador con dos núcleos tomó en promedio 8 segundos, sin GPUs para efectuar multiprocesamiento.

Aunque el tiempo utilizado para procesar una imágen no sería suficiente para obtener una fluidez adecuada para catalogarlo como 'en tiempo real', el resultado es bastante acertado, tomando en cuenta las limitaciones del hardware utilizado. Se podría obtener un mejor desempeño utilizando máquinas mas potentes, o simplemente haciendo uso de librerías de paralelismo de Python, que permiten distribuir trabajos no solamente a través de hilos de ejecución, sino también a través de múltiples computadores conectados entre sí por medio de una red de área local.

6.2. Extracción de poses humanas

Luego de las pruebas, se decidió utilizar el modelo Caffe mediante la implementación de las funciones DNN de OpenCV. En este apartado se dará una breve explicación de los detalles del mismo.

La configuración del modelo de red previamente entrenada se grafica en la siguiente figura:

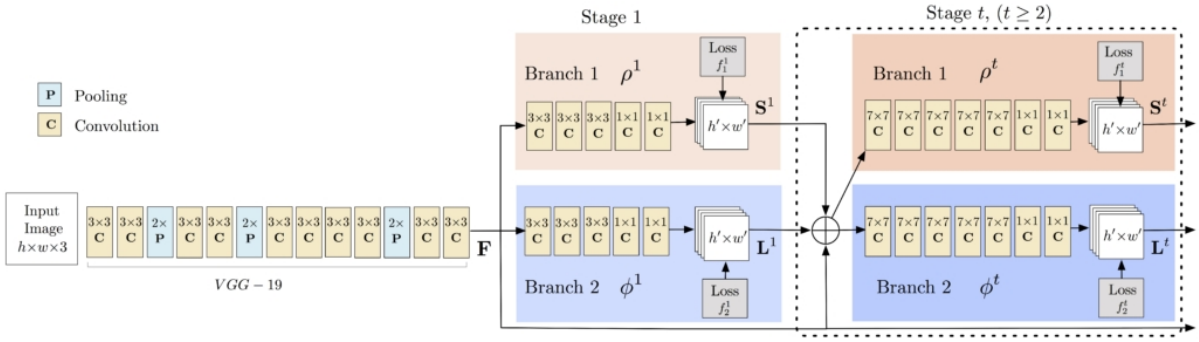

Figura 6.5: Estructura visual del modelo de red neuronal artificial Caffe para estimación de poses humanas, Fuente learnopencv.com.

Detalles de la estructura:

1. Las primeras 10 capas de la red se utilizan para crear un mapa de características de la imágen. Contiene capas pooling y convolucionales.
2. En la segunda fase (Stage 1), el Branch1 se utiliza para crear un confidence map, en inglés, que es básicamente un mapa de la información ya obtenida. El mapa se realiza en base a las ubicaciones de partes del cuerpo. En el Branch2 se predicen puntos de afinidades, esto es, se trata de predecir a que punto cercano corresponde un punto determinado (ej.: hombro con codo, pie con rodilla, etc.)
3. En la última fase, en realidad se repite una estructura cierta cantidad de veces: se toman los puntos del mapa de afinidades y del mapa de características y se aplica lo que se denomina algoritmo voraz, cuya finalidad es elegir la opción óptima dentro de un grupo de opciones.

El modelo devuelve un conjunto ordenado de puntos que conforman el cuerpo humano comenzando por la cabeza, hasta llegar a los pies. Así, la cabeza tendría el índice 0 en el arreglo, el cuello sería índice 1, hombros 2 y 3, y así sucesivamente hasta llegar a los pies.

6.3. Cálculo angular de extremidades

Una vez establecidos los puntos relevantes del cuerpo humano a través de la utilización de una herramienta para la estimación de poses, es necesario establecer los ángulos que las extremidades del cuerpo forman con el tronco del mismo. De ésta manera podemos caracterizar una pose y guardar la configuración de la disposición de los mismos.

Para el cálculo angular, se utilizaron ecuaciones trigonométricas simples como ser: pendiente de una recta, diferencia de pendientes entre dos rectas, distancia existente entre dos puntos en un plano, entre otras. Del lado derecho del cuerpo, las extremidades inferiores, los ángulos se miden en sentido antihorario tomando como eje línea de origen, el tronco humano, y en sentido horario para las extremidades

superiores. Del lado derecho se invierte el sentido, las extremidades superiores tienen sentido antihorario y los inferiores sentido horario.

Cada extremidad cuenta con dos partes, teniendo cuatro extremidades, se proceden a calcular ocho ángulos: antebrazos izquierdo y derecho, brazos izquierdo y derecho (sección del músculo biceps), muslos izquierdo y derecho, y pantorrillas izquierda y derecha. Todos estos suman ocho partes a considerar.

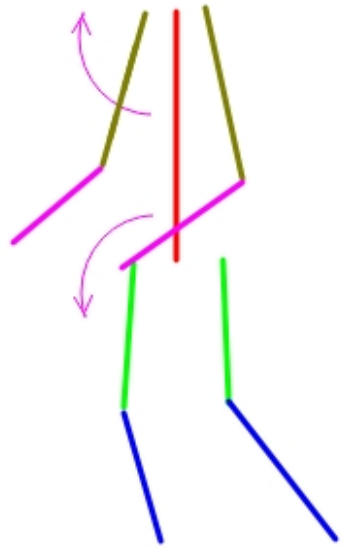

Figura 6.6: Ejemplo de medición del lado izquierdo del cuerpo: los miembros superiores, con la flecha que indica el sentido de medición angular, y los inferiores, en sentido contrario. Del lado derecho se invierten los sentidos.

6.3.1. Estimación de tronco

En un escenario perfecto, las imágenes captadas por una cámara pueden describir todas las partes del cuerpo humano de una persona en escena, pero no siempre sucede. Existen ocasiones en que, o el sujeto se encuentra muy cerca de la cámara, o algún objeto se interpone entre el cuerpo y la cámara.

El concepto de medición de ángulo entre dos rectas sugiere que ambas deben existir obligatoriamente, no solamente es necesario que se visualice una extremidad, sino también el tronco del cuerpo humano, que es básicamente el origen de la

medición angular en los dos casos citados en la figura.

Estimar o suponer la presencia de una extremidad, es mas complicado porque no se sabe si la extremidad existe en realidad, o si está dispuesto de una forma que no podemos imaginar o deducir, así que la única opción es intentar estableces la ubicación del tronco, si este no existiese en la imágen.

En caso de que no pueda visualizarse el tronco, se siguieron las siguientes premisas:

- Si los puntos de ambos hombros del cuerpo existen, elegir como ubicación superior del tronco, al punto medio comprendido entre los puntos de los hombros, esto es, utilizando una ecuación trigonométrica, obtener el punto medio entre dos puntos.
- Si los puntos de ambas caderas existen, elegir como ubicación inferior del tronco al punto medio comprendido entre ambos puntos.
- Si unicamente existe un punto hombro, utilizarlo como ubicación superior, ya que el tronco no se encuentra tan lejos del mismo. Además, esto cubre los casos en que una persona puede no estar mirando a la cámara, y uno de sus hombros es invisible.
- Si unicamente existe un punto cadera, utilizarlo como ubicación inferior del tronco, del mismo modo que se utiliza en el ítem anterior para hombro.

De esta manera, si no contamos con los datos explícitos de los puntos que conforman el tronco, es posible deducir su ubicación matemáticamente.

6.3.2. Representación interna de partes del cuerpo

Una vez deducido el tronco, se agrupan las ubicaciones de los puntos con sus etiquetas de acuerdo al índice devuelto por el modelo Caffe, y de acuerdo a la categoría de extremidades a la que pertenecen, ejemplo:

'brazoDerecho ' : [('codoDerecho': (323, 42)), ('hombroDerecho': (421, 25))]
'piernaIzquierda' : [('pieIzquierdo': (234, 124)), ('rodillaIzquierda': (242, 75))]

De este modo, se envían como argumentos a una función que obtiene el ángulo entre cada extremidad del cuerpo, con el tronco del mismo, devolviendo un arreglo con la lista de ángulos:

['brazoDerecho' : 27.3], ['piernaIzquierda' : 12.5]

Esto se interpreta como: brazoDerecho forma un ángulo de 27.3° con el tronco, piernaIzquierda forma ángulo de 12.5° con el tronco. Esto sucede con las demás extremidades por igual. Al final de todo este proceso, debemos obtener un conjunto de ocho ángulos ordenados para poder ser utilizados como valores individuales de cada feature de la red neuronal:

[81.2, 41.5, 12.2, 0.0, 155.4, 13.0, 75.3, 43.0]

6.3.3. Generación de datos de entrenamiento

Se busca ahora generar el conjunto de todos los ángulos posibles para dos clases distintas de poses: poses compatibles con ataque, y poses no compatibles con ataque. La red neuronal debe entrenarse con un mínimo de dos clases para poder devolver una predicción válida, de esta manera, debemos ingresar valores que la red pueda predecir como casos positivos, o de lo contrario, casos negativos.

El método de entrenamiento es sencillo: se deben generar primero, grupos de poses compatibles con agresiones, para el caso de valores positivos, y grupos de poses no compatibles con agresiones para los negativos.

Para el efecto, se procedió a la captura mediante la cámara incorporada del cuerpo humano de una persona en una acción cuya pose sugiera 1. algún tipo de amenaza con arma o 2. agresión con brazos (se podría entrenar con poses de mas tipos de agresiones en un trabajo futuro). No se contabilizó el número de casos positivos, ya que simplemente eran almacenados en un archivo. Luego se procedió a la captura de un cuerpo humano cuya pose esté fuera del margen considerado como amenaza o agresión. Ambas exposiciones a la cámara para la recolección de ángulos fueron realizadas a lo largo de varias horas, varios días.

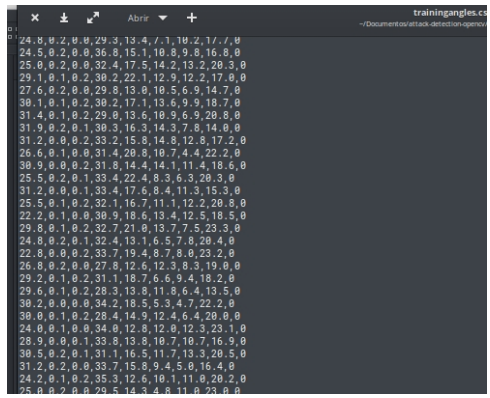

Figura 6.7: Ejemplo de un conjunto de valores de entrenamiento generados.

Se generaron en total 716.310 combinaciones de poses positivas y negativas para poses que se utilizarán para entrenar la red neuronal.

6.4. Entrenamiento de una red neuronal artificial

6.4.1. Búsqueda de un modelo óptimo

Los modelos de redes neuronales son variables, y no existe un modelo preestablecido para la predicción de valores en un problema particular. La búsqueda de un modelo implica la prueba de configuraciones de modelos, cambiando el número de capas, el número de neuronas y en algunos casos las funciones de entrada, activación y salida de la red.

La finalidad de la búsqueda de un modelo óptimo, es encontrar una configuración de disposición de capas y funciones que reduzcan el margen de error y aumenten la precisión de las predicciones del modelo.

Pruebas con modelos de red

Para determinar el modelo óptimo, se hicieron básicamente 6 pruebas distintas utilizando para la mayoría de los casos 8 entradas (ángulos de la pose), 3 capas en

total, 1 neurona en la capa de salida y el optimizador Adam por defecto. Solo en una prueba se utilizaron 4 capas en total.

Ya que el optimizador solamente interfiere en la etapa de entrenamiento, no se decidió probar con otros optimizadores: el modelo de red resulta sencillo y rápido de inferir.

Se citan únicamente las configuraciones de capas y neuronas, y el resultado del valor de precisión obtenido en las pruebas:

Prueba 1 : Red neuronal con capa de entrada de 14 neuronas, 1 capa oculta de 12 neuronas. Resultado precisión: 97.7

Prueba 2 : Red neuronal con capa de entrada de 12 neuronas, 2 capas ocultas de 14 y 10 neuronas respectivamente. Resultado precisión: 96.8

Prueba 3 : Red neuronal con capa de entrada de 14 neuronas, 1 capa oculta de 14 neuronas. Resultado precisión: 97.3

Prueba 4 : Red neuronal con capa de entrada de 14 neuronas, 1 capa oculta de 8 neuronas. Resultado precisión: 98.4

Prueba 5 : Red neuronal con capa de entrada de 12 neuronas, 1 capa oculta de 10 neuronas. Resultado precisión: 98.2

Prueba 6 : Red neuronal con capa de entrada de 16 neuronas, 1 capa oculta de 8 neuronas. Resultado precisión: 99.1

La conclusión: se decidió por un modelo sencillo de 3 capas, detallado en la Prueba 6. No se realizaron mas pruebas.

6.5. Predicción de poses compatibles con amenazas o agresiones

En este punto, la red entrenada ya está lista para inferir. Para el efecto, se escribió una aplicación en Python que pueda cargar el modelo pre entrenado de poses y, a

medida que vaya captando imágenes de la cámara incorporada, pueda determinar si existe o no una pose compatible con agresión en la imágen.

6.6. Detección de armas utilizando detectores en cascada

Para que una amenaza o agresión humana sea considerada como tal, no es suficiente contar con una pose compatible, sino además, la presencia en escena de un arma. Si bien la aplicación puede determinar mediante redes neuronales la ocurrencia de una pose compatible con una agresión, también se agregó una característica que permite detectar armas de fuego en la mano de la persona cuya pose resultó positiva para la clasificación.

Para el efecto, se descargó un conjunto de datos pre entrenados de https://github.com/JYang25/opencv_demo que contiene características de features Haar-like que describen armas de fuego.

Figura 6.8: Detección de armas de fuego cortas en la mano de una persona cuya pose es compatible con una agresión.

La detección se efectúa mediante clasificadores en cascada que por defecto se implementan en OpenCV. Este se activa luego de que una pose compatible con agresión es detectada, y solamente se genera una alerta extra en caso de que el arma esté ubicada cerca de los puntos de la mano de aquella pose positiva para agresión.

No se incluyeron datos de todas las armas disponibles para efectuar agresiones

humanas (cuchillos, hachas, arcos y flechas, etc), solamente fueron utilizados descriptores de armas de fuego cortas.

Capítulo 7

Conclusiones

Si bien, la extracción de poses humanas en el presente trabajo no es el indicado para ejecutarse en computadores de gama baja, se puede lograr una velocidad de respuesta mas adecuada para ser considerada como una herramienta de tiempo real utilizando multiprocesamiento, o en todo caso, algún computador disponible en el mercado que contenga características mas avanzadas de cómputo.

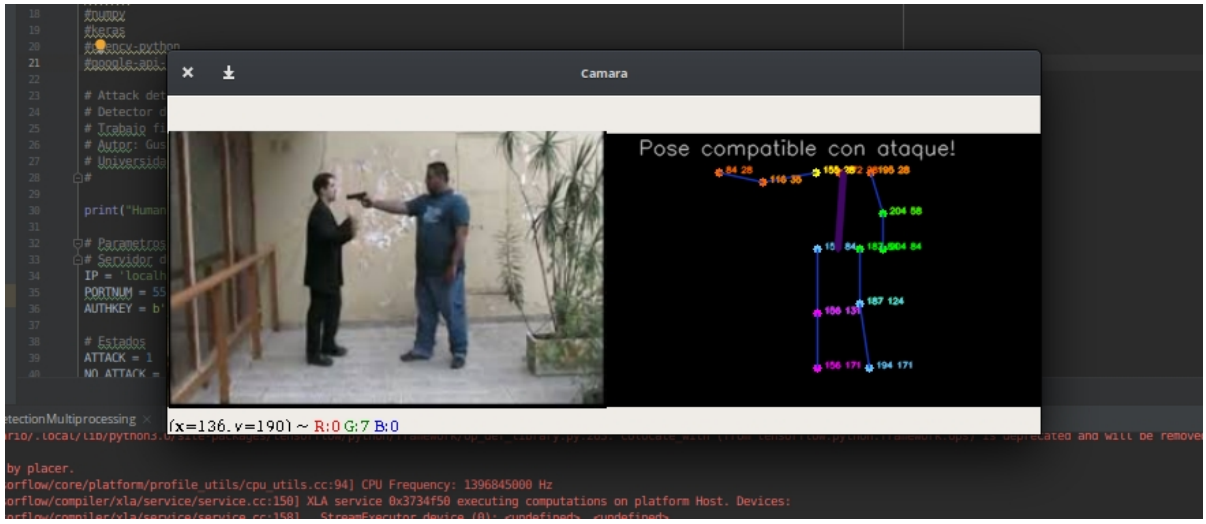

Figura 7.1: Aplicación ejecutándose, en la que se puede observar un caso positivo para pose compatible con agresión

Sin embargo, aún con un retraso de unos pocos segundos, esta aplicación podría utilizarse en el monitoreo de espacios donde puedan ocurrir agresiones humanas y así poder alertar de manera temprana sin intervención humana sobre posibles delitos a la integridad llevándose a cabo en un momento dado.

La utilización de hardware con prestaciones altas de cómputo definitivamente lograría una respuesta en tiempo real de la estimación de poses, y por ende, de la red entrenada para la detección de poses compatibles con amenazas.

7.1. Trabajo a futuro

Una alternativa al presente trabajo consistiría en la utilización de múltiples fuentes (cámaras) para la recolección de imágenes haciendo de la aplicación una herramienta para la detección completa en entornos visuales mas amplios.

Además, no todas las poses existentes consideradas como agresiones o amenazas fueron establecidas para el entrenamiento, ya que la gama de posibilidades es muy amplia. Se podría también, incluir una mayor cantidad de tipos de armas que puedan ser utilizadas por el humano para proferir agresiones.

Otra posibilidad sería la de implementar un detector de poses mediante redes neuronales mas sencillas que no requieran el uso de GPUs. Las opciones son amplias.

Bibliografía

[1] M. Valera, S.A. Velastin. *Intelligent distributed surveillance system: a review. IEE Proc. Vis. Image Signal Process.* 152(3):192–204, 2005.

[2] 1.R. Dautov, S. Distefano, D. Bruneo, F. Longo, G. Merlino, A. Puliafito, R. Buyya. *Metropolitan intelligent surveillance systems for urban areas by harnessing IoT and edge computing paradigms.* 48(2):1475–1492, 2018.

[3] M. D. Ruiz Lozano. *Un modelo para el desarrollo de sistemas de detección de situaciones de riesgo capaces de integrar información de fuentes heterogéneas. Aplicaciones.* Granada, 2010.

[4] Zhang, G.D., Jiang, P.L., Matsumoto, K., Yoshida, M. and Kita, K. *An Improvement of Pedestrian Detection Method with Multiple Resolutions.* Journal of Computer and Communications. 5(1) 102-116, 2017

[5] Nidhi. Dept. of Computer Applications, NIT Kurukshetra, Haryana, India. *Image Processing and Object Detection.* International Journal of Applied Research 2015; 1(9): 396-399, 2015.

[6] J.L. Barron, D.J. Fleet, S.S. Beauchemin *Performance of Optical Flow Techniques.* IJCV 12:1 pp43-77, 2015.

[7] N. S. Kamarudin, M. Makhtar, S. A. Fadzli, M. Mohamad, F. S. Mohamad, M. F. A. Kadir *Comparison of Image Classification Techniques using Caltech 101 Dataset.* Journal of Theoretical and Applied Information Technology, 71(2):1992-8645, 2015.

[8] S. Bailey, C. Aragon, R. Romano, R. C. Thomas, B. A. Weaver, D. Wong *How to find more Supernovae with less work: Object Classification Techniques for Difference Imaging.* Journal of Theoretical and Applied Information Technology, 665(2):1246-1253, 2007.

[9] Intel Corp. *The OpenCV Reference Manual, Release 3.0.0-dev* 2014.

[10] Andrew Ng*Machine Learning Course*, Coursera, www.coursera.org.

[11] Damián Jorge Matich *Redes Neuronales: Conceptos Básicos y Aplicaciones*, Universidad Tecnológica Nacional, 2001.

[12] C.Enyinna Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall *Activation Functions: Comparison of Trends in Practice and Research for Deep Learning*, 2018.

[13] Enrique J. Carmona Suárez *Tutorial sobre Máquinas de Vectores Soporte (SVM)*, Universidad Nacional de Educación a Distancia (UNED), 28040-Madrid España, 2014.

[14] N. Dalal, B. Trigg *Histograms of Oriented Gradients for Human Detection*, EEE Computer Society Conference on Computer Vision and Pattern Recognition, (1):886–893, 2005.

[15] Ong Chin Ann, Lau Bee Theng *Human Activity Recognition: A Review*, 2014 IEEE International Conference on Control System, Computing and Engineering, 2014.

[16] Joan Reig Doménech *Estudio del estado del arte de los métodos de estimación de la pose humana en 3D*, Universidad de Alicante, España, 2018.

[17] H.Fang, S.Xie, Y.Tai, C.Lu *RMPE: Regional Multi-Person Pose Estimation*, Shanghai Jiao Tong University, China, 2018.

[18] Y.LeCun, L.Bottou, Y.Bengio, P.Haffner *Gradient-Based Learning Applied to Document Recognition*, IEEE, 1998.

[19] Z.Cao, T.Simon, S.Wei, Y.Sheikh *Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields*, The Robotics Institute, Carnegie Mellon University, 2017.

[20] Jia, Yangqing and Shelhamer, Evan and Donahue, Jeff and Karayev, Sergey and Long, Jonathan and Girshick, Ross and Guadarrama, Sergio and Darrell, Trevor *Caffe: Convolutional Architecture for Fast Feature Embedding*, arXiv preprint arXiv:1408.5093, 2014.