ACADEMIA JOURNALS
OPUS PRO SCIENTIA ET STUDIUM

Humanidades, Ciencia, Tecnología e Innovación en Puebla

ISSN 2644-0903 online
Vol. 5. No. 1, 2023
www.academiajournals.com

TRABAJO DE INVESTIGACIÓN AUSPICIADO POR EL CONVENIO CONCYTEP-ACADEMIA JOURNALS

Gobierno de Puebla
Hacer historia. Hacer futuro.

Secretaría de Educación
Gobierno de Puebla

CONCYTEP
Consejo de Ciencia y Tecnología del Estado de Puebla

Alfonso Hernandez Hernandez

Detección de Objetos y Acciones no Comunes Mediante Aprendizaje Profundo

Benemérita Universidad Autónoma de Puebla

Presidente: M.C. Rogelio Alfredo Campos Serapio
Secretario: Dr. Ivan Olmos Pineda
Vocal - Asesor: Dr. José Arturo Olvera López

Número de Secuencia 5-1-18

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

TESIS

“DETECCIÓN DE OBJETOS Y ACCIONES NO COMUNES MEDIANTE APRENDIZAJE PROFUNDO”

COMITÉ REVISOR:
PRESIDENTE M.C. ROGELIO ALFREDO CAMPOS SERAPIO
SECRETARIO DR. IVAN OLMOS PINEDA
VOCAL DR. JOSE ARTURO OLVERA LOPEZ

BUAP
FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

FECHA DE APROBACIÓN 5 DE ABRIL DEL 2022

QUE PARA OBTENER EL TÍTULO DE:

LICENCIADO EN INGENIERÍA EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA:

ALFONSO HERNANDEZ HERNANDEZ

ASESOR:

DR. JOSÉ ARTURO OLVERA LÓPEZ

PUEBLA, PUEBLA

RESUMEN

Detección de objetos y acciones no comunes mediante aprendizaje profundo

Autor: Alfonso Hernandez Hernandez

En la era actual los datos representan información valiosa pues son fundamentales principalmente para extraer información útil que sirve para generar evaluaciones o servicios. La informática ha tratado este tema desde los años 50 dándole cierto interés en diversas épocas, pero se dio un gran aumento a este campo con el auge de dispositivos con un buen nivel de procesamiento y algoritmos especializados.

Así surgen ramas como el machine learning o aprendizaje automático que trata de encontrar modelos matemáticos que permitirán hacer evaluaciones. Sin embargo, esto tiene un gran problema pues los investigadores deben elegir alguno de los métodos existentes y además deben elegir que datos se tomaran para entrenar. Por tal motivo surgieron nuevas técnicas para tratar estos problemas de una manera más general y tomar todos los datos con el fin de generar un modelo matemático, a esto se le denomina deep learning o aprendizaje profundo. Uno de los campos de deep learning que más investigación tiene es el estudio de imágenes donde se busca hacer clasificación, detección y segmentación.

En este trabajo que tiene por título "**Detección de objetos y acciones no comunes mediante aprendizaje profundo**" realizado por **Alfonso Hernandez Hernandez** se plantea crear una aplicación de apoyo en vigilancia a través de deep learning clasificando y detectando imágenes además de seguir regiones de objetos con el propósito de alertar cuando una persona olvida un objeto. Para lograr esta meta se utilizan tecnologías como la programación, el deep learning y un algoritmo de seguimiento.

Agradecimientos

- A mis profesores.
- A mis padres por el apoyo incondicional.
- A la universidad pública por el sistema educativo que me brindó.
- A mi hermano por la broma que le conté y a Google- Machine Learning: Solving Problems Big, Small, and Prickly precisamente el día que me inspiraron en idear el tema de esta tesis.
- Al profesor José Arturo Olvera López pues siempre facilitó el apoyo para el desarrollo de este trabajo.

Esta investigación fue realizada gracias al apoyo del Consejo de Ciencia y Tecnología del Estado de Puebla

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN Y MARCO TEÓRICO

CAPÍTULO 1

En este capítulo se presenta la introducción con los objetivos de este trabajo de investigación. En particular, las áreas de interés en este trabajo de tesis son referentes a machine learning, deep learning y seguimiento de objetos por lo que algunos conceptos relacionados al respecto se describen en este capítulo.

1.1 Introducción

La detección, clasificación de objetos y el seguimiento de regiones en imágenes son problemas vigentes en las ciencias de la computación. Actualmente, este tipo de problemas pueden solucionarse mediante enfoques de aprendizaje profundo, específicamente con redes convolucionales y algoritmos de seguimiento. Comúnmente en herramientas libres de deep learning como Opencv [24] o Caffe [43] existen implementaciones de modelos entrenados de redes convolucionales y algoritmos de seguimiento que pueden ser utilizados por separado, es decir, la clasificación esta apartada del seguimiento.

Por lo tanto el caso de estudio en este trabajo plantea combinar enfoques de modelos pre-entrenados de redes convolucionales para clasificar objetos y un algoritmo de seguimiento para crear un aplicativo teniendo en cuenta lo siguiente:

Objetivo general

Crear una aplicación de apoyo en vigilancia que reconocerá objetos mediante deep learning además de seguir regiones para detectar acciones no comunes en espacios cotidianos.

Objetivos específicos

- Investigar y analizar dos modelos existentes de aprendizaje profundo para la detección de objetos.

- Llevar a cabo seguimiento y detección mediante técnicas basadas en deep learning y algoritmos de seguimiento.

- Implementar un algoritmo de decisión que permitirá mostrar o no una alerta respecto a la distancia entre objeto y persona.

- Analizar archivos de videos de espacios con condiciones reales que sean aplicables a este proyecto.

A continuación se detalla un poco más los temas que aborda este trabajo.

1.2 Detección y clasificación en imágenes

En el campo de la visión por computadora existen temas que son muy estudiados como son la clasificación, la detección y el seguimiento de imágenes. La detección es segmentar la ubicación de un objeto en una imagen. La Figura 1 muestra un ejemplo de la detección.

Figura 1. El recuadro azul se denomina bounding box.

La clasificación es asignar un nombre de clase en el área de detección de una imagen. En la Figura 2 se clasifica la detección.

Figura 2. El área dentro del bounding box es “Persona”.

Para la solución de la clasificación y la detección en imágenes los investigadores han propuesto diferentes algoritmos para realizar esas tareas. Precisamente en la actualidad es el deep learning que son modelos con múltiples capas de procesamiento con el fin de aprender representaciones de los datos con múltiples niveles de abstracción [1] que presenta buenos resultados matemáticos por lo que se utiliza para segmentar y clasificar. Como

ejemplo están las redes convolucionales que se componen de elementos muy generales en la solución de problemas [22].

En los diferentes enfoques que se han propuesto se proponen arquitecturas nuevas que mejoran la capacidad de detectar y clasificar. Por ejemplo existen arquitecturas que son consideradas estados del arte como Alexnet [14], R-cnn [16], VGG [15]. Existen propuestas de nuevas arquitecturas que responden a ciertas optimizaciones, por ejemplo, Google propuso la arquitectura Mobilenet [35] que es rápida para dispositivos móviles.

1.3 Seguimiento de regiones en imágenes

Consiste en seguir una región a través del flujo de imágenes dentro de un video a través de algoritmos matemáticos. Los métodos más usados son los filtros de correlación. Actualmente grupos proponen enfoques para la investigación de algoritmos de seguimiento como VOT (Visual Object Tracking) [58] y MOT (Multiple Object Tracking) [9]. La Figura 3 muestra un ejemplo gráfico del concepto de seguimiento.

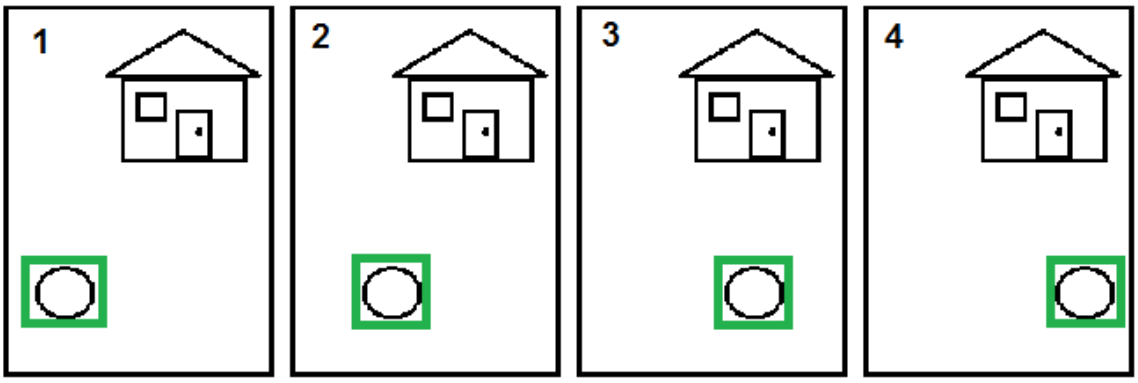


Figura 3. Se propone seguir una pelota dentro de un video con 4 imágenes o frames, entonces el algoritmo de seguimiento deberá mostrar la posición de la pelota a lo largo del video.

1.4 Marco teórico

Un modelo clasificador es una función que permite decidir que elementos de un conjunto están relacionados para pertenecer o no a una clase [20]. El aprendizaje automático o machine learning a través de sus enfoques permite realizar clasificación. Sin embargo para tareas más específicas como clasificación de objetos en imágenes el deep learning arroja mejores resultados.

1.5 Machine learning

Es el campo de la ciencia de la computación que tiene como objetivo estudiar y obtener modelos de “aprendizaje” matemáticos a través de datos, se denomina aprendizaje pues es conocimiento útil, representable, medible y

comparable en un sentido matemático. Para obtener dichos modelos se utiliza la probabilidad, algebra de matrices, etc [3]. Algunas de las técnicas más aplicadas son naive bayes (una técnica que aplica el teorema de bayes), k-nearest neighbor o knn (un algoritmo que compara los vecinos más cercanos), k-means una técnica que agrupa datos, redes neuronales que son aproximadores a funciones y árboles de decisiones los cuales constituyen una regla de ganancia, en general estos métodos son los más utilizados [36].

A continuación se expone el perceptrón simple y además otros conceptos que en general son la base en el campo del aprendizaje profundo.

1.5.1 Perceptrón simple

El perceptrón toma una entrada o lista de entradas y realiza una transformación matemática para obtener una salida [2]. Se define con la fórmula:

$$y = f(W^t x) \tag{1}$$

Donde x son las entradas (vector o un array multi-dimensional), W^t es la matriz de pesos que se ajustan durante el entrenamiento (W^t servirá después para obtener y). $f()$ es una función de activación y finalmente y es la salida. La operación matemática en $W^t x$ es la multiplicación de matrices. W^t representa la transpuesta de la matrix W que permite realizar la multiplicación con x. El número de entradas en el perceptrón es n.

Actualmente el perceptrón también se le conoce como neurona y se representa como la Figura 4.

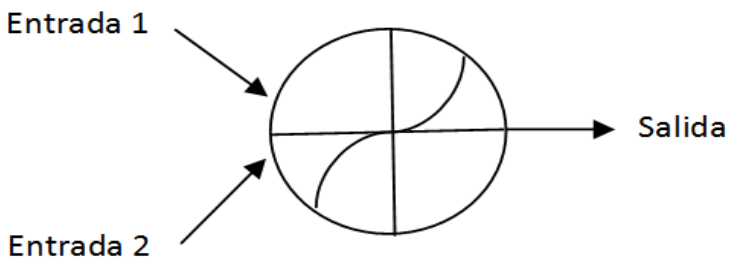


Figura 4. Perceptrón simple, 2 entradas aplican una combinación matemática para obtener una salida.

Los orígenes del perceptrón datan de 1940 cuando Warren McCulloch y Walter Pitt describieron como podría trabajar una neurona [5], después la primera implementación fue el perceptrón de Rosenblatt en 1950.

1.5.2 Función de activación

La función de activación se encarga de devolver una salida a partir de un valor de entrada y mantiene un conjunto de valores en un rango [18] [19]. En la Figura 5 se observa esta relación. Además en la Tabla 1 se presentan las funciones de activación más importantes en machine learning.

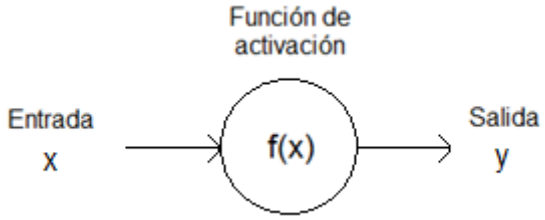


Figura 5. Representación gráfica de la función de activación.

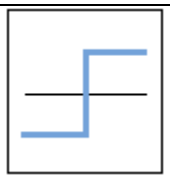
Nombre	**Relación de entrada x y salida y**	**Gráfica**
Hard limit	$y = 0 \quad x < 0$ $y = 1 \quad x \geq 0$	
Symmetrical Hard Limit	$y = -1 \quad x < 0$ $y = 1 \quad x \geq 0$	
Symetric Saturating Linear	$y = -1 \quad x < -1$ $y = x \quad -1 \leq x \leq 1$ $y = 1 \quad x > 1$	
Hyperbolic Tangent Sigmoid	$y = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	
Positive Linear	$y = 0 \quad x < 0$ $y = x \quad 0 \leq x$	

Tabla 1. Funciones de activación en machine learning.

1.5.3 Sesgo del perceptrón

En la forma $y = f(W^t x)$ se le puede agregar un bias el cual es un parámetro que sirve para minimizar la función de costo [17].

$$y = f(W^t x + b) \tag{2}$$

Donde W^t son los pesos, x las entradas, b el bias y f la función de activación que obtendrán una salida y.

1.5.4 Perceptrón con 1 salida y 1 bias

Dentro de un perceptrón simple con 2 entradas x_1 y x_2 existirán 2 pesos representados W_1 y W_2 que se combinarán en una sumatoria junto con el bias, esta sumatoria se le aplicará una función de activación para obtener una salida. A continuación se coloca un ejemplo gráfico en la Figura 6.

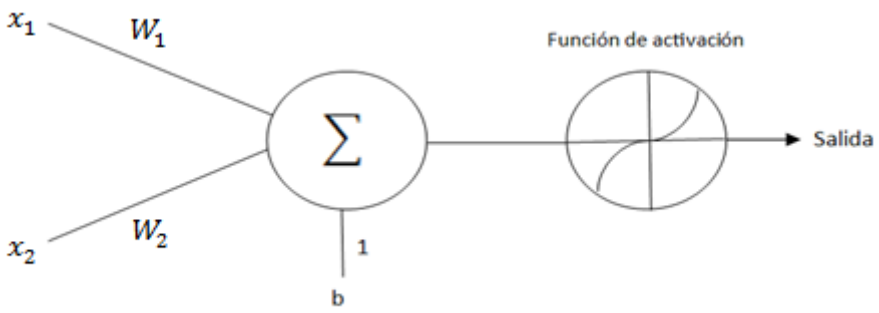


Figura 6. A partir de 2 entradas x_1, x_2 el perceptrón proporciona una salida mediante la función de activación.

La fórmula de este ejemplo es:

$$Salida = f(W_1 x_1 + W_2 x_2 + b) \tag{3}$$

Finalmente supongamos que el bias fuera multiplicado por 0 en tal caso la arquitectura quedaría de la forma $y = f(W^t x)$.

1.5.5 Arquitectura con 2 neuronas

A continuación colocamos la Figura 7 que es una arquitectura con 2 neuronas y 2 salidas con la forma $y = f(W^t x)$. Cada entrada debe combinarse con cada neurona lo que conlleva a crear más pesos. W_1 y W_3 para la salida 1, además W_2 y W_4 para la salida 2.

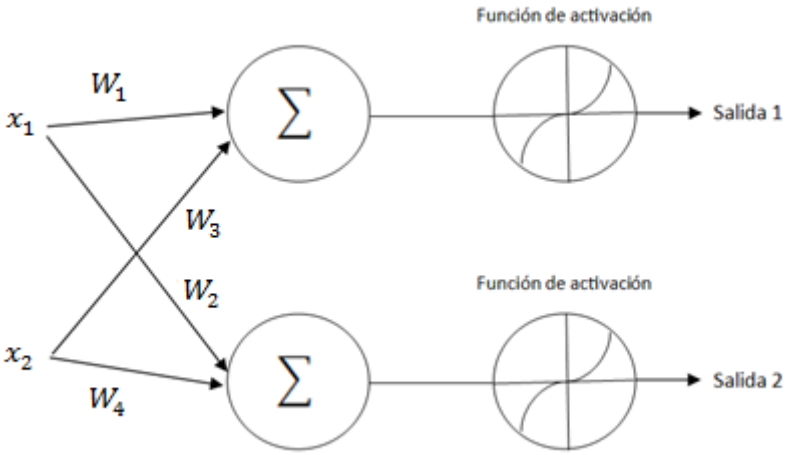


Figura 7. A partir de 2 entradas x_1, x_2 dos perceptrones proporcionan dos salidas mediante sus funciones de activación.

La fórmula de este ejemplo es:

$$\begin{bmatrix} Salida\ 1 \\ Salida\ 2 \end{bmatrix} = \begin{bmatrix} f(W_1 x_1 + W_3 x_2) \\ f(W_2 x_1 + W_4 x_2) \end{bmatrix} \quad (4)$$

1.5.6 Redes neuronales

Las redes neuronales están compuestas de perceptrones [5] (también se le puede nombrar neuronas o nodos), cumplen estar completamente conectadas y presentan capas de neuronas. En la Figura 8 podemos observar un ejemplo de una arquitectura de red neuronal:

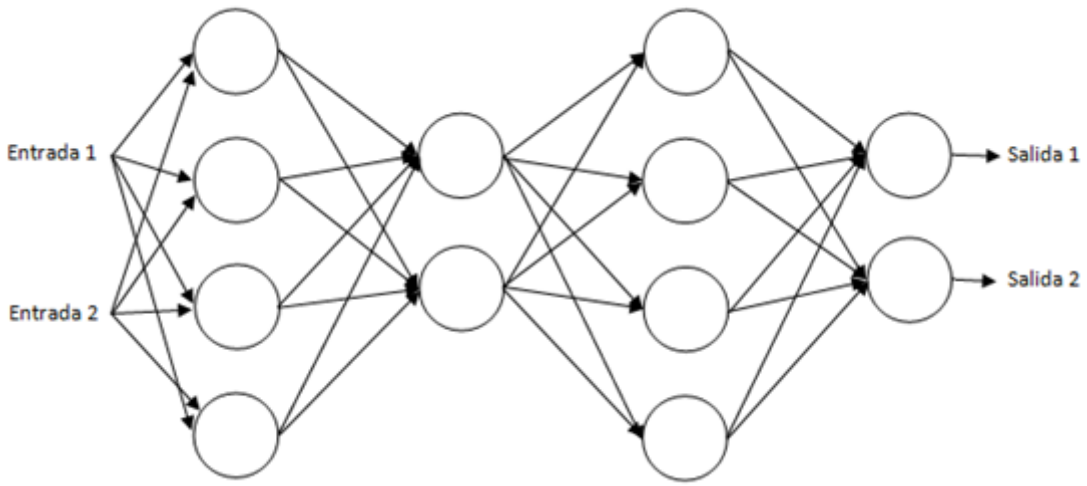


Figura 8. Una red neural tiene entradas, nodos en la misma jerarquía denominados capas, pesos y salidas.

Estas redes también cumplen el terminó de feed forward networks [5] ya que no tiene bucles y la entrada fluye a través de la red en una sola dirección.

Los parámetros aquí son los pesos (weights) los cuales se actualizarán respecto a la entrada.

El objetivo es obtener una red optima con una buena capacidad de clasificar por medio de técnicas que ofrece el machine learning. Para esto, se necesita un conjunto de entradas x con sus clasificaciones y en donde el aprendizaje de los pesos (denominado entrenamiento o training) las entradas x pasan en la red capa por capa. En cada capa, la entrada de la capa anterior es transformada de acuerdo a las propiedades de la capa. La salida final es la predicción o clasificación.

También para actualizar los pesos se utiliza la función de perdida que mide que tan alejada esta la predicción del valor correcto de y junto con un enfoque basado en derivadas llamado gradiente descendente. Otro algoritmo fundamental es backpropagation. Las redes neuronales se definen también como aproximadores universales a funciones [36].

Para entender más la estructura de las redes neuronales completamente conectadas a continuación se desarrolla un apartado del tema.

1.5.7 Redes neuronales completamente conectadas

Consiste en una serie de capas completamente conectadas. Una capa completamente conectada es una función de $\mathbb{R}^m$ a $\mathbb{R}^n$ [5]. Cada dimensión de salida depende de cada dimensión de entrada. La forma matemática de una red neuronal completamente conectada es la siguiente. Definamos $x \in \mathbb{R}$ representa la entrada a una red completamente conectada y definamos $y_i \in \mathbb{R}$ ser la i-esima salida de una red completamente conectada. Entonces $y_i \in \mathbb{R}$ es calculada como:

$$y = f(W_1 x_1 + \cdots + W_m x_m) \tag{5}$$

Aquí f es una función no lineal, y w_i son los parámetros para aprender (pesos) en la red. La salida completa es entonces:

$$y = \begin{pmatrix} f(W_{1,1} x_1 + \cdots + W_{1,m} x_m) \\ \vdots \\ f(W_{n,1} x_1 + \cdots + W_{n,m} x_m) \end{pmatrix} \tag{6}$$

Hay tres tipos de capas: capa de entrada, capa oculta y capa de salida. En la Figura 9 se puede observar estas capas.

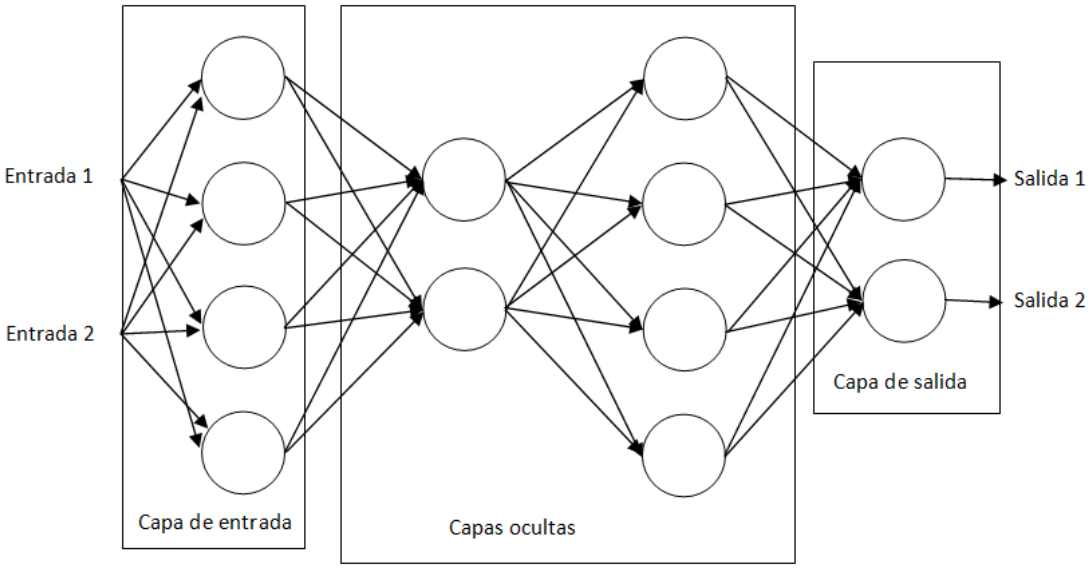


Figura 9. Distintos tipos de capas de una red neuronal.

Si hay un número mayor a una capa oculta la literatura marca que se vuelve una red profunda. El número de capas ocultas puede ser n.

1.5.8 Conceptos de machine learning

Para extraer características en redes neuronales o perceptrones existen conceptos que a continuación se enlistan además se puede consultar el Anexo A para ver bases de datos:

- **Entrenamiento supervisado**: En las redes convolucionales y perceptrón se entrena de una manera supervisada, es decir, las clasificaciones de los ejemplos en el entrenamiento se conocen y mediante algoritmos de entrenamiento la red aprenderá características para evaluar nuevos ejemplos.
- **Función de perdida (loss function)**: Mide el error que se produce de la salida que se predice (clasificación) con respecto a la salida esperada (etiqueta original) [10].
- **Gradiente descendente**: Calcula la pérdida y actualiza los pesos después de una iteración. Su fórmula es:

$$Weights = Weights - lerning\ rate * loss \quad (7)$$

  Donde $Weights$ son los pesos, el $lerning\ rate$ es un valor híper parámetro y $loss$ la perdida.

- **Época**: Una época es el ciclo donde se actualizan los pesos respecto a las operaciones internas como el gradiente descendente.
- **Algoritmo Backpropagation**: Retro propaga el error que se obtiene hacia capas anteriores con el objetivo de minimizar el error.
- **Inicialización de pesos**: Se propone inicializar de manera aleatoria los pesos o con optimizadores de pesos como la inicialización Glorot.

- **Overfitting**: Es el problema de obtener un buen desempeño para clasificar en la etapa de entrenamiento pero en la etapa de validación se obtiene resultados bajos.
- **Gráfica del error**: Permite visualizar cómo se comporta el error a través de las épocas.

1.5.9 Modelo en machine learning

El objetivo del perceptrón o de las redes neuronales es obtener pesos ajustados que permitan clasificar. Los pasos generales en un modelo machine learning con neuronas son los siguientes:

- Recabar datos del problema los cuales deben estar procesados, es decir, datos sin ruido, seleccionadas las mejores características, seleccionar una parte de los datos para hacer evaluación y validación. Además estos datos deben contar con sus clasificaciones correspondientes.
- Diseñar una arquitectura que puede ser desde el perceptrón unitario, capas de perceptrones, redes neuronales. También se debe elegir una función de activación.
- Los pesos se actualizan en épocas en un conjunto de al menos el 80% de los datos del conjunto con algoritmos como el gradiente descendente, la función de perdida, backpropagation. Este proceso se denomina entrenamiento.
- Al terminar el entrenamiento se evalúan los pesos con una porción de datos restante de 10% para la evaluación. Si el porcentaje de clasificación es alto se procede a la siguiente etapa. Si no, se puede volver a re-entrenar dando más épocas, seleccionando una función de activación diferente o eligiendo otra arquitectura.
- La siguiente etapa es la validación donde la otra porción del 10% de datos se evalúa, también se propone evaluar todo el conjunto de datos para ver cómo se comporta. Finalmente si se obtiene un porcentaje de clasificación con exactitud superior al 80% entonces ese modelo es correcto y puede clasificar.
- Finalmente se almacenan los pesos para después cargarlos y poder clasificar o re-entrenar.

Sin embargo tener una buena comprensión del problema para entrenar un modelo, es decir, estudiar el problema, procesar el ruido, elegir un conjunto específico de datos y elegir alguna de las técnicas que mejor resuelvan el problema no garantiza crear un buen modelo. Para resolver algunos de estos problemas surge el deep learning que es el caso de estudio en este trabajo. Históricamente sucedieron ciertos acontecimientos e investigaciones que dieron paso al deep learning. A continuación se da una referencia.

1.6 AI Winter

Este término denomina el periodo a lo largo de la vida de los tópicos de ingeniería de datos (machine learning, visión por computadora,deep learning e inteligencia artificial) en el cual se ha perdido el interés de estudiar, investigar y desarrollar estos temas ya que hay factores que lo impiden. Por ejemplo, en sus inicios en 1950 era complicado hacer un perceptrón multicapa además el hardware tenía un procesamiento pequeño y por tal motivo se perdió el interés hasta que en los años 80 se desarrolló el algoritmo backpropagation que permitió hacer posible la clasificación en capas, sin embargo en este periodo se prefirió realizar otros enfoques más simples pues todavía era poco el procesamiento del hardware.

El enfoque sobrevivió pero con poca investigación por parte de los investigadores, hasta que en los años 90 Yann LeCun propuso las redes convolucionales, un enfoque práctico pero muy robusto que se colocó como un caso de estudio.

Finalmente este tópico fue muy relevante hasta el 2012 con Alex Krizhevsky pues él creó una red convolucional denominada Alexnet la cual utilizó para la competencia ILSVRC2012 de visión por computadora donde obtuvo los mejores resultados en la clasificación [41]. Esto provocó que las redes convolucionales tomaran gran relevancia. Actualmente este tema tiene una gran comunidad de investigadores y desarrolladores, además se utiliza en la industria. En la siguiente sección se explica más a detalle el deep learning.

1.7 Deep learning

Por décadas, construir un sistema de machine learning (aprendizaje automático) requería una considerable ingeniería y habilidad para construir un extractor de características que transformara los datos crudos en transformaciones internas adecuadas, además se debía de utilizar alguno de los clasificadores existentes. En gran parte por estas 2 cuestiones los investigadores plantearon crear métodos de aprendizaje con propósitos generales denominados deep learning.

El deep learning o aprendizaje profundo es el enfoque de las técnicas que permiten crear modelos computacionales compuestos de múltiples capas de procesamiento, técnicas que tienen el fin de aprender representaciones de los datos con múltiples niveles de abstracción [1].

Los métodos de deep learning son de representación del aprendizaje, es decir, son un conjunto de métodos que permite a un sistema ser alimentado con datos crudos y automáticamente descubrir las representaciones necesarias para la detección o la clasificación con múltiples niveles de representación, obtenidos gracias a la simple composición de módulos no lineales los cuales

transformaran la representación en niveles más abstractos. Con la composición de suficientes de estas transformaciones se puede aprender funciones complejas.

También se necesita un gran conjunto de datos para aprender buenas características por lo tanto los conjuntos de entrenamiento son por lo regular grandes (por ejemplo se puede entrenar con todas las imágenes existentes en la web respecto al problema que se quiere resolver). Algunos ejemplos de tecnologías que utilizan deep learning se pueden consultar en el *Anexo B*.

Actualmente el deep learning es una rama de investigación muy activa, algunos de sus métodos son Convolutional neural networks y Recurrent neural networks. En este proyecto solo se propone trabajar con el método de las Covolutional neural networks o redes convolucionales pues han demostrado que pueden llevar a cabo de manera aceptable los procesos de extracción de características y la clasificación de imágenes.

1.8 Redes convolucionales

Son estructuras de redes neuronales que emplean como una de sus operaciones la convolución con la finalidad de extraer características [2]. En la literatura se les llama convnets, redes convolucionales, redes conv o CNN por convolutional neural networks.

CNN busca características de bajo nivel tales como curvas y bordes para después construir conceptos más abstractos a través de una serie de capas convolucionales [22].

Algo fundamental es que las convnets reducen el número de operaciones y además son redes feed-forward, es decir, la información fluye hacia delante dentro de la arquitectura hasta llegar al final sin presentar bucles. Las redes conv tienen elementos específicos que a continuación se explican.

1.8.1 Campos receptivos locales y convolución en 2D

Un campo receptivo local consiste en subdividir la imagen en cuadrantes específicos que serán entradas específicas a la red. La Figura 10 muestra un ejemplo de esto.

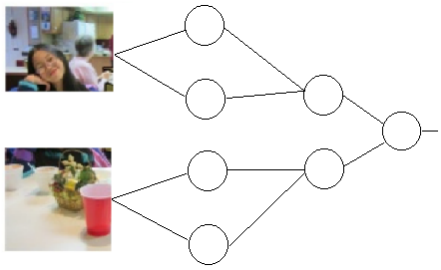


Figura 10. Secciones de imágenes entran a capas específicas.

Ahora cuando se trabaja con 2 dimensiones de entrada tales como una matriz X_{n1*n2} y una matriz filtro (kernel convolucional) W_{m1*m2}, donde $m1 \leq n1$ y $m2 \leq n2$, entonces la matriz

$$Y = X * W \tag{8}$$

es el resultado para la convolución 2D de $X\ con\ W$ [3]. El símbolo $*$ representa la operación de convolución y cada elemento en la matriz Y se define:

$$Y[i,j] = \sum_{k_1=-\infty}^{+\infty} \sum_{k_2=-\infty}^{+\infty} X[i-k_1, j-k_2] W[k_1, k_2] \tag{9}$$

La operación entre los elementos de X y W es el producto punto. Además la convolución vista en el apartado anterior también puede expresarse con un bias y expresarse de la siguiente manera:

$$\mathrm{Y} = \mathrm{X} * \mathrm{W} + \mathrm{b} \tag{10}$$

La convolución 2D permite reducir el número de operaciones que la conforman dependiendo la combinación en sus parámetros. Hay 2 conceptos relacionados con la convolución en 2D el primero es el Stride y el segundo es el Padding. A continuación se explican.

1.8.2 Stride

Es el paso que recorre el kernel en el campo receptivo para realizar la convolución comenzando por la parte superior izquierda [2] [23], es decir, en la posición (1,1). Este valor puede ser 1 o menor a la dimensión horizontal de X. Cada nueva posición para la convolución está a la misma distancia vertical y horizontal respecto al valor del Stride. Por ejemplo, en la Figura 11 se presentan diferentes valores de Stride en la matriz X.

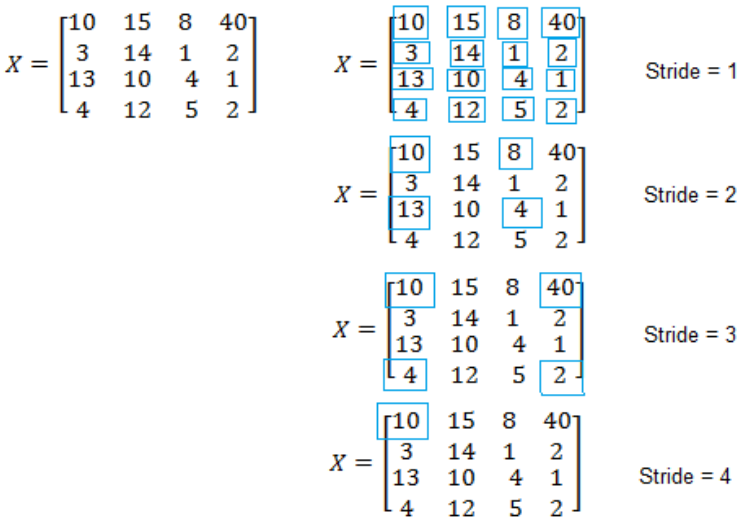


Figura 11. Los cuadros en azul son las posiciones de los Strides.

Un kernel puede entonces posicionarse en un recuadro azul y solo junto con sus elementos vecinos a la derecha hacer la convolución. Por ejemplo la matriz X, un kernel W de 2x2 y un Stride igual a 2 presenta convoluciones como en la Figura 12.

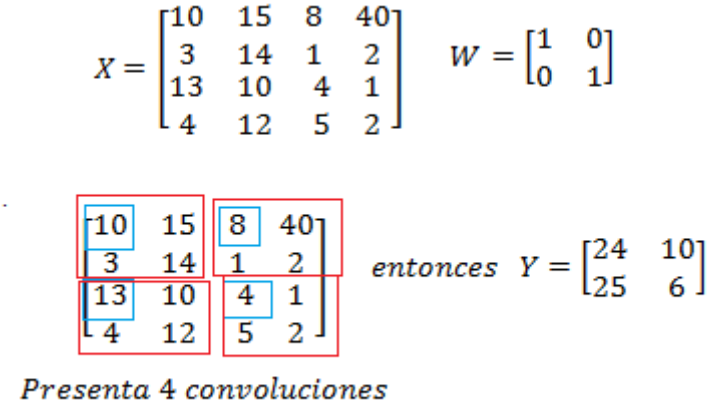


Figura 12. Los cuadros en azul son posiciones de Strides y los cuadros en rojo representan los lugares donde el kernel realizará la convolución.

Ahora si en la misma matriz X se utiliza un kernel W de 3x3 y el Stride de 2, el número de convoluciones se presenta en la Figura 13 sin embargo como las otras regiones no hay elementos suficientes no se hace convolución.

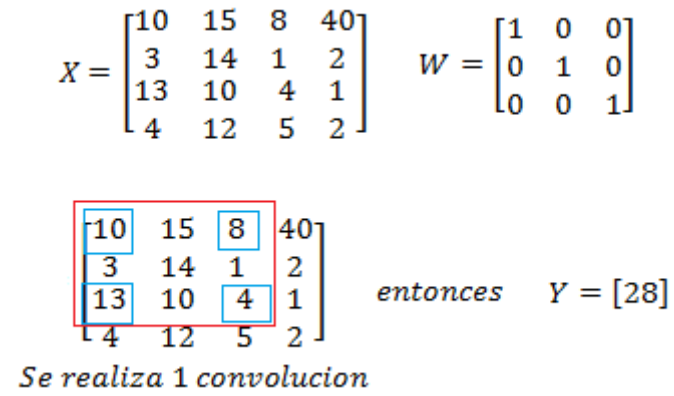


Figura 13. Los cuadros en azul son Strides sin embargo por el tamaño de kernel solo se permite una convolución que se representa con el cuadro rojo.

1.8.3 Padding

Es colocar una columna extra de ceros a la imagen original o feature map para obtener como resultado dimensiones diferentes en Y.

$$X = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 0 \\ 0 & 4 & 1 & 2 & 0 \\ 0 & 5 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{11}$$

Hay 3 tipos de padding p respecto a su valor donde $p \geq 0$.

- **Full (Completo):** El valor de p puede tener un valor igual al valor de los elementos horizontales del campo receptivo y como resultado las dimensiones en Y seran más grandes que el campo receptivo.
- **Same (Igual):** El valor de p se calcula respecto al kernel para dar como resultado Y con las mismas dimensiones que el campo receptivo. Es padding más utilizado.
- **Valid (Valido):** El valor de p es igual a 0 entonces las dimensiones de Y serán menores.

1.8.4 Convolución en imágenes

Una imagen digital se representa como una matriz de valores lo que permite aplicar la convolución en 2D. La Figura 14 muestra modelos de color con sus dimensiones.

Figura 14. Dimensiones de imágenes en modelos de color diferentes.

Hay dos casos posibles de convolución de imágenes manejados en la literatura, el primer caso trata imágenes en escala de grises y el segundo trata imágenes a color. Para el primer caso se toma solamente la imagen con dimensiones mxn realizando el proceso de la convolución en 2D. Para el segundo caso se toma una imagen de colores es decir $mxnxc$ donde c son las dimensiones de color o canales, en este caso se realiza el proceso para cada canal como si fuera mxn y los resultados de las convoluciones se suman.

1.8.5 Kernel convolucional y funciones de activación

Los kernels son los pesos que aprende la red [22], además un kernel tiene la misma profundidad que el campo receptivo al que se le aplica y su resultado es la convolución con solo una dimensión. La literatura propone kernels con dimensiones 1x1, 3x3, 5x5. También un conjunto de kernels forman un filtro [21].

Ahora bien en deep learning se propone utilizar funciones no lineales para devolver una salida, la literatura utiliza mucho ReLU y para realizar clasificación en forma de probabilidades se utiliza la función Softmax. A continuación en la Tabla 2 se presentan algunas de las funciones de activación más utilizadas.

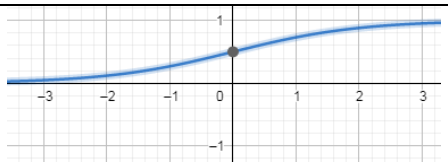
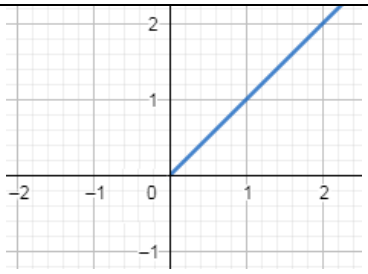
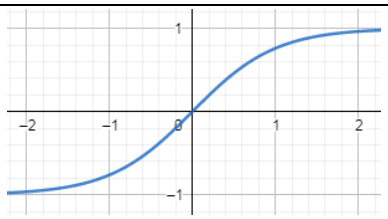
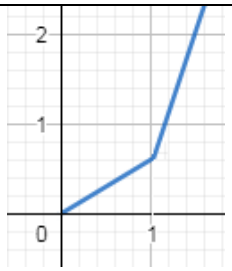
Nombre	Función	Grafica
Sigmoide	$f(x) = \frac{1}{1 + e^{-x}}$	
ReLU Unidad lineal rectificada	$f(x) = \max(0, x)$	
Tanh Tangente Hyperbolico Sigmoide	$f(x) = \tanh(x)$ $f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$	
Leaky ReLU ReLU con fugas	$f(x) = \max(\alpha x, x)$ Donde α es un número flotante pequeño y positivo	
Softmax	$f(x)_i = \frac{e^{x_i}}{\sum_i e^{x_i}}$	Para $f(x)$ se obtiene un número real R donde $0 < R < 1$

Tabla 2. Funciones de activación que se usan en aprendizaje profundo.

1.8.6 Feature map

El siguiente paso es obtener H que es el mapa de características (feature maps) [3] resultado de aplicar una función de activación sobre Y.

$$\mathrm{H} = f(\mathrm{Y}) \tag{12}$$

Cada elemento de Y se activa con la misma función de activación f.

$$H = f(Y) = \begin{bmatrix} f(y_{1,1}) & f(y_{1,2}) & \cdots & f(y_{1,n}) \\ f(y_{2,1}) & f(y_{2,2}) & \cdots & f(y_{2,n}) \\ \vdots & \vdots & \ddots & \vdots \\ f(y_{m,1}) & f(y_{m,2}) & \cdots & f(y_{m,n}) \end{bmatrix} \tag{13}$$

1.8.7 Pooling

En conv nets reduce la dimensión del mapa de características a través de seleccionar regiones y construir con esos datos una nueva matriz. Además al realizar esto se agrega invariancia local lo que significa que valores altos tienen más impacto y decrece el número de los elementos lo que es benéfico para el cálculo de operaciones.

Hay 3 tipos de pooling: maxpooling para los valores más altos, minpooling para los valores más bajos y averangepooling para el promedio de cada región. El más usado es maxpooling. Por ejemplo, si se selecciona 4 regiones para generar una matriz pooling 2x2 se puede ver el resultado en la Figura 15.

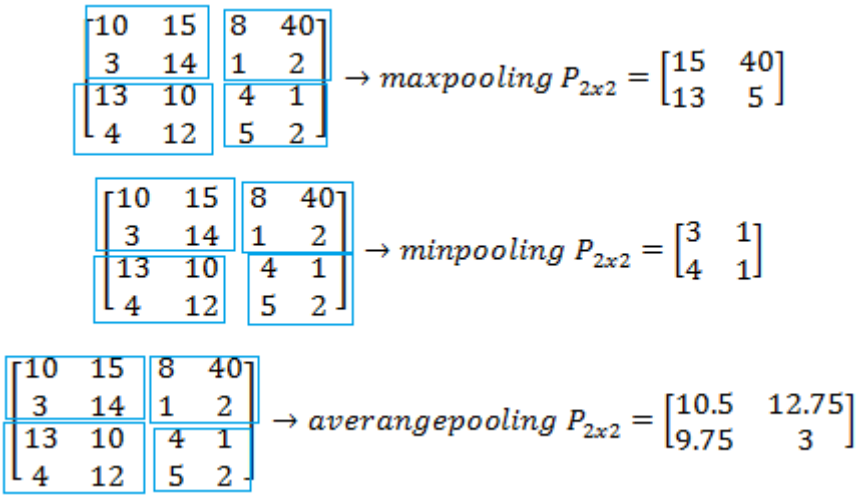


Figura 15. Los cuadros en azul son las regiones para crear el pooling.

El resultado del pooling puede ser la salida de la capa convolucional la cual se usa como entrada para otra capa convolucional o capa completamente conectada.

1.8.8 Capa completamente conectada y flatten layer

En la capa completamente conectada todas las salidas de la capa i se conectan a todas las entradas de la siguiente capa i+1, se le denomina también fully conected layer como en la Figura 16.

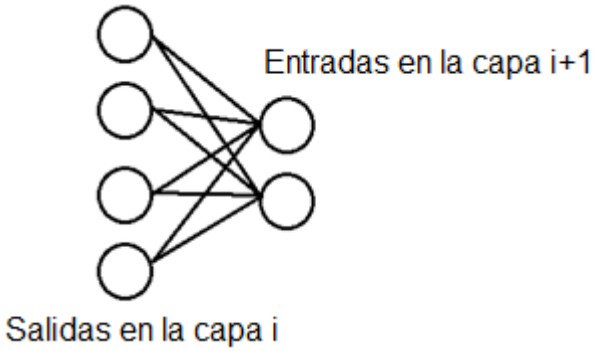


Figura 16. Capa completamente conectada.

Para la capa flatten layer es un tipo de capa que redimensiona la entrada a un array unidimensional, es decir, el primer elemento de entrada pasa a ser el primero en la salida flatten y el segundo elemento sería el segundo elemento del array y asi sucesivamente. A continuación se observa el proceso en la Figura 17.

$$Entrada = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, Salida = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$$

Figura 17. Proceso de la Flatten layer.

1.8.9 Dropout

Es un enfoque de regularización para prevenir overfitting. En redes neuronales durante la etapa de entrenamiento algunas neuronas de las capas más altas se “borran” (se evitan ciertas salidas de neuronas a otras neuronas) aleatoriamente en cada iteración con la probabilidad p_{drop} la cual se utiliza en la siguiente formula.

$$p_{keep} = 1 - p_{drop} \tag{14}$$

Donde p_{keep} es la probabilidad de las neuronas que tienen salida en la capa. p_{drop} es un numero entre 0 y 1 (generalmente es .5). La Figura 18 muestra una representación gráfica del dropout.

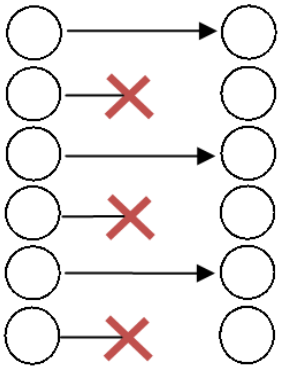


Figura 18. Dropout evita que algunas salidas se vuelvan entradas.

El efecto del dropout fuerza a la red a aprender una representación redundante de los datos. Es decir, la red no podrá relacionar sobre una activación de algún conjunto de neuronas ocultas especificas ya que estas pudieron ser apagadas durante el entrenamiento por lo tanto la red es forzada a aprender patrones de datos más generales y robustos. Se debe de tomar en cuenta que cuando se “borran” las entradas de neuronas entonces se deben re-escalar los pesos. En la etapa de evaluación todas las neuronas estarán activadas y contribuirán para la salida a la siguiente capa.

1.8.10 Backpropagation o retro propagación

Es una regla generalizada para aprender los pesos de redes neuronales. Supongamos que $f(\theta, x)$ es una función que representa una red profunda completamente conectada. Aquí x son las entradas a la red y θ son los pesos para aprender. Entonces el algoritmo backpropagation consiste en calcular $\frac{\partial f}{\partial \theta}$. Además permite minimizar el error de la función de costo C (error) a través de calcular derivadas parciales de los pesos $\frac{\partial C}{\partial w}$ y los bias $\frac{\partial C}{\partial w}$ en conjunto de un algoritmo que a continuación se describe.

1. Entrada: Activa neuronas de la primera capa.

2. Propagación: para cada capa l=2, 3,.., L (capa final) calcular.

$$z^l = w^l a^{l-1} + b^l \tag{15}$$

$$a^l = \sigma(z^l) \tag{16}$$

Donde $z^l = w^l a^{l-1} + b^l$ representa z^l la suma vector para los pesos ponderados, w^l representa los pesos de la capa, b^l representa los bias de la capa, a^l representa la salidas de activación de la capa, es decir, $\sigma(z^l)$. Entonces a^{l-1} representa las salidas de la capa anterior.

3. Calculo de errores en la capa de salida, se calcula el vector

$$\delta^L = \nabla_a C \odot \sigma'(z^L) \quad (17)$$

Donde $\nabla_a C$ es el error de la capa de salida que se aplica con el producto de hadamarv $\odot$con $\sigma'(z^L)$.

4. Retro propagación del error: Para cada l=L-1, L-2,…,2 calcular:

$$\delta^l = (w^{l+1})^T \delta^{l+1} \odot \sigma'(z^L) \quad (18)$$

5. Salida: El gradiente de la función de costo C viene dado por:

$$\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_k \quad (19)$$

$$\frac{\partial C}{\partial b^l_j} = \delta^l_k \quad (20)$$

El algoritmo de retro propagación se utiliza en la regla delta o en el gradiente estocástico que más adelante se explican.

1.9 Capa de convolución en CNN

En la Figura 19 se observa de manera gráfica como se realiza el proceso de una capa convolucional o capa conv. Cada kernel respecto a la entrada producirá un mapa de características. El número de kernels en una capa convolucional puede ser un número pequeño, de cientos o miles.

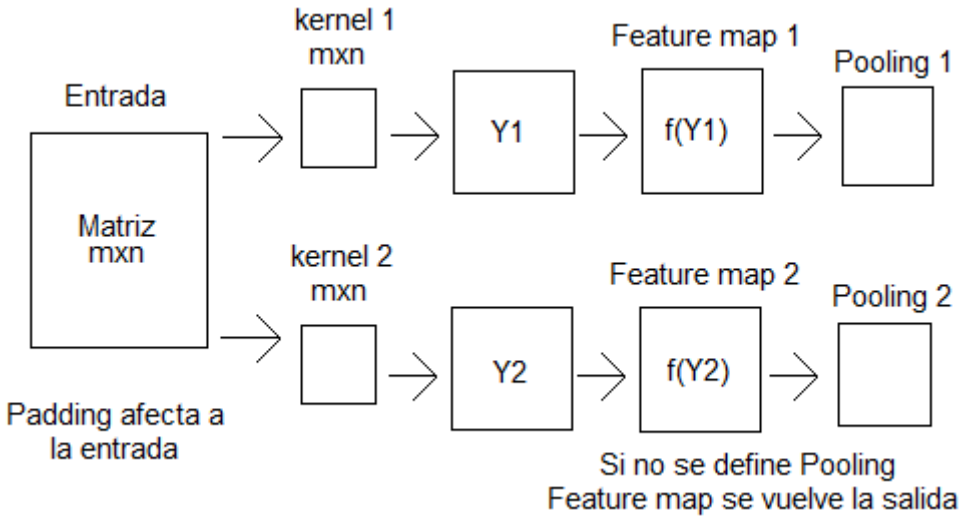


Figura 19. Flujo de datos de capa CNN.

. A continuación se colocan las fórmulas para medir parámetros y salidas.

- El número de parámetros de kernels en la capa se calcula con la siguiente formula:

$$kernels_{parámetros} = forma_{kernel} * numero_{kernels} \quad (21)$$

Donde $forma\_kernel$ representa el valor de $mxnxc$ y $numero\_kernels$ el número de kernels de la capa.

- El número de salidas que se obtiene de una capa convolucional se calcula como vectores con la siguiente operación:

$$o = \left\lfloor \frac{I + 2P - K}{S} \right\rfloor + 1 \tag{22}$$

Donde I es el tamaño entrada, K el tamaño del kernel, P es el padding usado y S es el Stride.

1.10 Arquitectura Convolutional Neural Network

Una red CNN está compuesta de diferentes capas, las principales son capas convolucionales, capas completamente conectadas, funciones de activación, optimizadores, métricas e híper parámetros. Estos elementos se colocan seguidos unos de otros, feed forward y sin bucles, por ejemplo la Figura 20. Las arquitecturas CNN son muy variadas. Para tareas de clasificación las capas altas o últimas de representación amplifican aspectos de la entrada que son importantes para la discriminación y además suprimen variaciones irrelevantes.

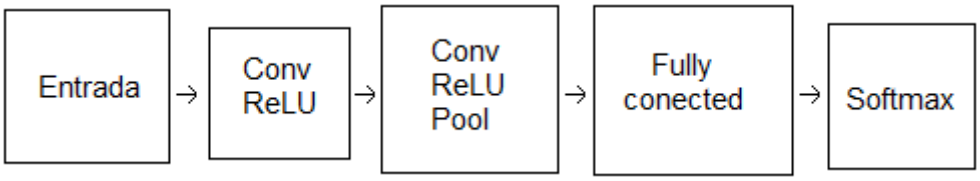


Figura 20. Arquitectura básica CNN.

1.10.1 Conceptos de CNN

A continuación se listan los pasos para realizar un modelo deep learning CNN, además se puede consultar en el Anexo A las bases de datos que son utilizadas para realizar pruebas y modelos deep learning.

- **Recolección de datos**

Recolectar datos es el proceso por el cual se obtienen los datos que se usan para entrenar el modelo.

- **Procesamiento de datos**

Se debe realizar un proceso de tratamiento a los datos como reducción de ruido, redimensionamiento, conversión de dimensiones, etc.

- **Diseño de la arquitectura**

En general puede tener los elementos que se definieron para la arquitectura CNN con diferentes capas conv, diferentes capas completamente conectadas, un número variado de kernels, valores diferentes para cada híper parámetro, métricas, optimizadores y con una función de pérdida. Por lo tanto se parte de diseños simples a más desarrollados.

- **Entrenamiento**

El entrenamiento en las redes convolucionales es supervisado lo que significa que se utilizan los datos y sus clasificaciones. Se utiliza el algoritmo backpropagation de inicio a fin, batches de ejemplos, una función de pérdida, optimizadores, métricas, dropout e híper parámetros en un número de épocas para ajustar el valor de los kernels. Esta etapa utiliza mucho tiempo por el número de operaciones que se deben de realizar. Por tal motivo es muy común el uso de gpu's que realizan un mayor número de cálculos en menor tiempo. El entrenamiento se puede repetir múltiples ocasiones para obtener mejores resultados.

- **Evaluación**

Este paso se realiza al terminó del proceso de entrenamiento. En esta etapa se evalúan ejemplos que no se utilizaron en la parte del entrenamiento con el modelo resultado del entrenamiento.

- **Validación**

En esta etapa se utilizan ejemplos que no se utilizaron ni en el entrenamiento ni en la evaluación para evaluar como clasifica el modelo.

Si el resultado de la etapa de entrenamiento supera las pruebas de evaluación y validación es un modelo terminado.

- **Guardar modelo**

Un modelo de redes convolucionales se guarda al terminar el entrenamiento para poder cargarlo después. También se puede ir guardando por partes cada vez que termina una iteración ya que así se podrá retomar el entrenamiento donde se quedó si sucediera un error.

- **Cargar modelo**

El modelo se carga para re-entrenarse o hacer predicciones.

Crear un modelo CNN es un proceso complejo, requiere de tiempo, estudio y pruebas. A continuación se exponen conceptos que se requieren en CNN.

1.10.2 Inicialización de pesos

Los kernels deben de inicializarse de tal manera que al hacer los cálculos se pueda facilitar la minimización del error. Por tal razón es común inicializar de forma aleatoria aunque actualmente se propone otra inicialización

más óptima denominada inicialización Xavier Glorot [13] que consiste en aproximar un balance de los gradientes a través de las diferentes capas.

1.10.3 Loss function o función de perdida

Mide el error que se produce de la salida que se predice (clasificación) con respecto a la salida esperada (etiqueta original) [10]. Es una métrica que ayuda a comprender si el aprendizaje va en la dirección correcta [11] resultado de los cálculos internos dentro de la red y se representa $\mathcal{L}$ [5].Hay dos problemas en la clasificación supervisada que definirán que función de perdida utilizar.

- El primero es el problema de clasificación que busca a través de un sistema de machine learning asignar una etiqueta discreta (0/1 o 0,…, n), un ejemplo de función a utilizar es la entropía cruzada.
- El segundo es el problema de regresión donde un sistema de machine learning adjunta un valor real a un elemento, en ejemplo de función a utilizar es L^2.

1.10.4 Optimizadores

Son algoritmos que ayudan a determinar la dirección del cambio requerido en los pesos para reducir la función de pérdida. Y es también gracias a la función de pérdida que se estima el cambio en los pesos. Es por lo tanto una relación donde la pérdida se calcula primero. También para la optimización se deben tener en cuenta los siguientes conceptos.

- **Paso**: Es el cálculo de un ejemplo de entrenamiento de la entrada a la salida.
- **Iteración**: Es cuando la red actualiza los pesos después de procesar todo el batch de ejemplos.

A continuación se explican algunos de los optimizadores más utilizados.

1.10.4.1 Gradiente descendente o regla delta

Supongamos que f es una función que depende sobre algunos pesos W. Entonces ∇W denota la dirección de cambio en W que debería incrementar a f. Esto significa que tomando pasos en la dirección opuesta se debería acercar a el mínimo de f. La idea del gradiente descendente es encontrar el mínimo de funciones por repetidamente seguir el gradiente negativo. Algorítmicamente esta regla se expresa de la siguiente manera:

$$W = W - \alpha \nabla W \tag{23}$$

Donde α es un tamaño de paso y dicta cuanto peso se le da al nuevo gradiente ∇W. ∇W se expresa de la siguiente manera:

$$\nabla W = \frac{\partial \mathcal{L}}{\partial W} \tag{24}$$

∇W representa el cálculo de la dirección que máximamente cambia la perdida $\mathcal{L}$.

1.10.4.2 Stochastic Gradient Descent (SGD)

SGD calcula la perdida y actualiza los pesos después de una iteración en un batch o en cada ejemplo individualmente, es un algoritmo rápido. A continuación se exponen detalles de estos casos.

- En el caso de batch se promedia la perdida para todos los ejemplos en el pedazo y se actualizan los pesos al final de la iteración.
- En el caso de ejemplos individuales se genera una curva de perdida con mucho ruido.

La fórmula de la actualización de los pesos es la siguiente.

$$Weights = Weights - lerning\ rate * loss \tag{25}$$

Donde $Weights$ son los pesos, el $lerning\ rate$ es un híper parámetro y $loss$ la perdida. Además este método es más exitoso cuando se entrena con datos suavizados y en batches. Se propone que los batches contengan un número de ejemplos base 2.

1.10.4.3 Adaptative Moment Estimation (Adam)

Este optimizador calcula una taza de aprendizaje adaptativa para cada parámetro. Este define momentum y varianza del gradiente de la perdida y permite un efecto combinado para actualizar los parámetros de pesos. La varianza y el momentum permiten suavizar la curva de aprendizaje y efectivamente mejorar el proceso de aprendizaje. La fórmula matemática es.

$$Weigths = Weights - (momentum\ and\ variance\ combined) \tag{26}$$

Donde $Weigths$ son los pesos.

1.10.5 Métricas de rendimiento

Este tipo de métricas permiten cuantificar el desempeño de un modelo. Existen diversas métricas que responden a casos específicos de clasificación o regresión. Para consultar más métricas hay un gran catálogo en [12]. A continuación se exponen algunas.

Accuracy: Es el porcentaje de acierto respecto a la entrada. Su fórmula es [48]:

$$Accuracy = \left(\frac{Entradas\ clasificadas\ correctamente}{Numero\ total\ de\ entradas}\right) * 100\% \qquad (27)$$

Mean Absolute Error: Calcula la media del error absoluto entre las etiquetas y las predicciones. Se utiliza en problemas de regresión. Su fórmula es la siguiente [49]:

$$Mean\ Absolute\ Error = \frac{\sum_{i=1}^{N}|y_i - x_i|}{N} \qquad (28)$$

Donde x son las etiquetas e y las predicciones. Además N es el número de entradas.

Mean Squared Error: Calcula la media del error cuadrático entre las etiquetas y las predicciones. Se utiliza en problemas de regresión. Su fórmula es la siguiente [50]:

$$Mean\ Squared\ Error = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2 \qquad (29)$$

Donde y son las etiquetas e $\hat{y}$ las predicciones. Además N es el número de las etiquetas.

Categorical Cross entropy: Calcula la métrica de entropía cruzada entre las etiquetas y las predicciones. Se utiliza en problemas de clasificación. Su fórmula es la siguiente [51]:

$$Categorical\ Cross\ entropy = -\sum_{i=1}^{N} y_i * log\hat{y}_i \qquad (30)$$

Donde y son las etiquetas e $\hat{y}$ las predicciones. Además N es el número de las etiquetas.

Binary Cross entropy: Calcula la métrica de entropía cruzada entre las etiquetas y las predicciones. Se utiliza en problemas de clasificación. Su fórmula es la siguiente [52]:

$$Binary\ Cross\ entropy = -(ylog(\hat{y}) + (1-y)\log(1-\hat{y})) \qquad (31)$$

Donde y son las etiquetas e $\hat{y}$ las predicciones.

Pearson R^2: Es una métrica de regresión que mide el conjunto de fluctuaciones de las predicciones y etiquetas de sus medias normalizadas por sus respectivos rangos de fluctuaciones [5].

$$R^2 = \frac{cov(x,y)}{\sigma(x)\sigma(y)} \qquad (32)$$

Donde x son las etiquetas, y las predicciones además cov es la covarianza y σ la desviación estándar.

Root Mean Squared Error (RMSE): Es una métrica de regresión que mide la cantidad absoluta del error entre la predicción y las cantidades verdaderas [5].

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(x_i - y_i)^2}{N}} \tag{33}$$

Donde N es el número de ejemplos de entrada, x las etiquetas e y las predicciones.

1.11 Interpretación visual de resultados

Es de gran ayuda ver el comportamiento de las clasificaciones por medio de gráficas. A continuación se exponen algunas de estas.

1.11.1 Gráfica del error

En una gráfica se coloca el comportamiento del error a través de las iteraciones. En la Figura 21 se coloca un ejemplo.

Figura 21. La gráfica del error indica las clasificaciones de los ejemplos erróneos que ocurren en cada iteración.

1.11.2 Curva de característica operativa del receptor o curva ROC

Una curva ROC es un gráfico que muestra el rendimiento de un modelo de clasificación binario en todas las iteraciones de clasificación [57]. Esta curva representa dos parámetros:

- Tasa de verdaderos positivos (TPR o True Positive Rate)

$$TPR = \frac{VP}{VP + FN} \tag{34}$$

- Tasa de falsos positivos (FPR o False Positive Rate)

$$FPR = \frac{FN}{FN + VP} \tag{35}$$

Donde VP son los verdaderos positivos y FN los falsos negativos. En la Figura 22 se muestra como luce una curva ROC.

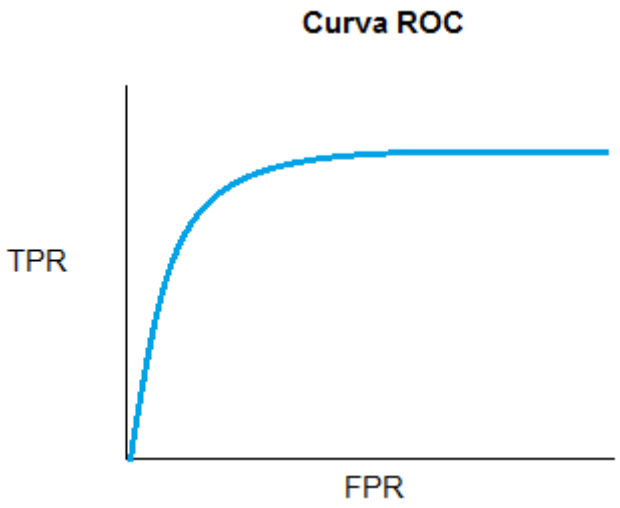


Figura 22. La curva ROC representa Tasa de verdaderos positivo frente a Tasa de falsos positivos en diferentes umbrales.

1.11.3 Área bajo la curva ROC o AUC

AUC mide toda el área bidimensional por debajo de la curva ROC [57]. Proporciona una medición agregada del rendimiento en todos los umbrales o iteraciones de clasificación posibles. Además es invariable con respecto a la escala y mide qué tan bien se clasifican las predicciones en lugar de sus valores absolutos. La Figura 23 muestra AUC.

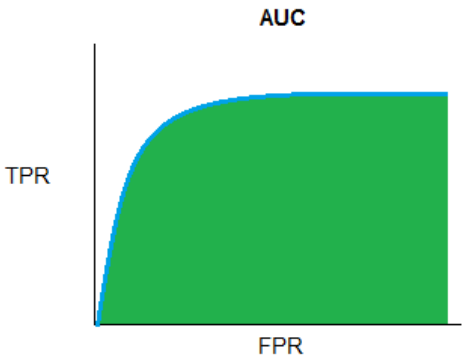


Figura 23. El AUC representa la probabilidad de que un ejemplo aleatorio positivo se posicione a la derecha de un ejemplo aleatorio negativo.

1.11.4 Matriz de confusión

Para un problema con k clases, la matriz de confusión es una matriz kxk. Donde la (i,j)-esima celda representa el número de puntos etiquetados como clase i con la etiqueta correcta de la clase j. La Figura 24 presenta un ejemplo de matriz de confusión.

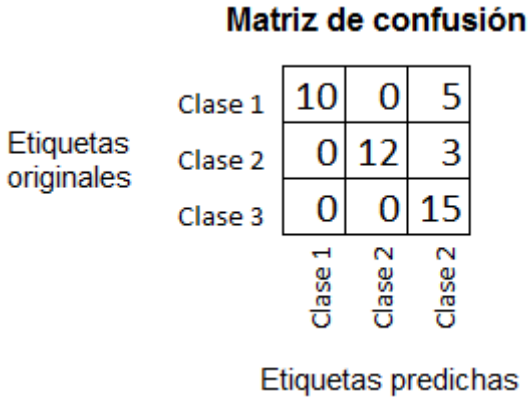


Figura 24. Una matriz de confusión indica como el conjunto de ejemplos de una clase en particular fueron clasificados, es decir, muestra sus ejemplos clasificados correctamente y los que se asignaron a otra clase.

1.12 Híper parámetros en redes convolucionales

Son parámetros que el programador de la red decidirá qué valor tendrán. A continuación se exponen algunos ejemplos:

Learning rate: La taza de aprendizaje dicta la cantidad de importancia a darle a cada paso de gradiente descendente.

Minibatching: Para grandes conjuntos de datos no es factible calcular los gradientes en el conjunto de datos completo. Se propone seleccionar un pedazo de los datos (típicamente entre 50 a 500) y calcular el gradiente en esos ejemplos, a esto se le denomina minibatching.

Época: Una época es el paso completo del algoritmo del gradiente descendente sobre los datos x. Mas particularmente muchos pasos de gradientes descendentes son requeridos para ver todos los datos dados en un tamaño del minibatch. Por ejemplo, supongamos que un conjunto de datos tiene 1000 ejemplos y en el entrenamiento se usa un minibatch de 50. Entonces cada época consistirá de 20 actualizaciones (o iteraciones) del gradiente descendente. Cada época del entrenamiento incrementa la cantidad de conocimiento útil que el modelo ha ganado. Matemáticamente esto consiste en reducciones al valor de la función de perdida en el conjunto de entrenamiento.

Momentum: Permite a la red aprender más rápidamente cuando la pendiente de la derivada de la función de criterio es pequeña. Se combina con el gradiente descendente [36]. El momentum se representa:

$$\alpha \Delta W(m-1) \tag{36}$$

Donde α es un número menor a 1. La fórmula combinada es la siguiente:

$$W(m+1) = W(m) + \Delta W(m) + \alpha \Delta W(m-1) \tag{37}$$

Donde m representa la actualización actual de los pesos, es decir, el momentum toma valor de la actualización previa.

1.13 Seguimiento de objetos

El seguimiento de reconocimiento de imágenes es el proceso de detectar la posición de una imagen referencia dentro de una imagen escena a través de algoritmos [25] [26]. En este trabajo se propone trabajar con los filtros de correlación en múltiples canales ya que son bastante populares en los algoritmos de tracking [27]. La Tabla 3 muestra un ejemplo donde la imagen de referencia es el rostro de la mujer.

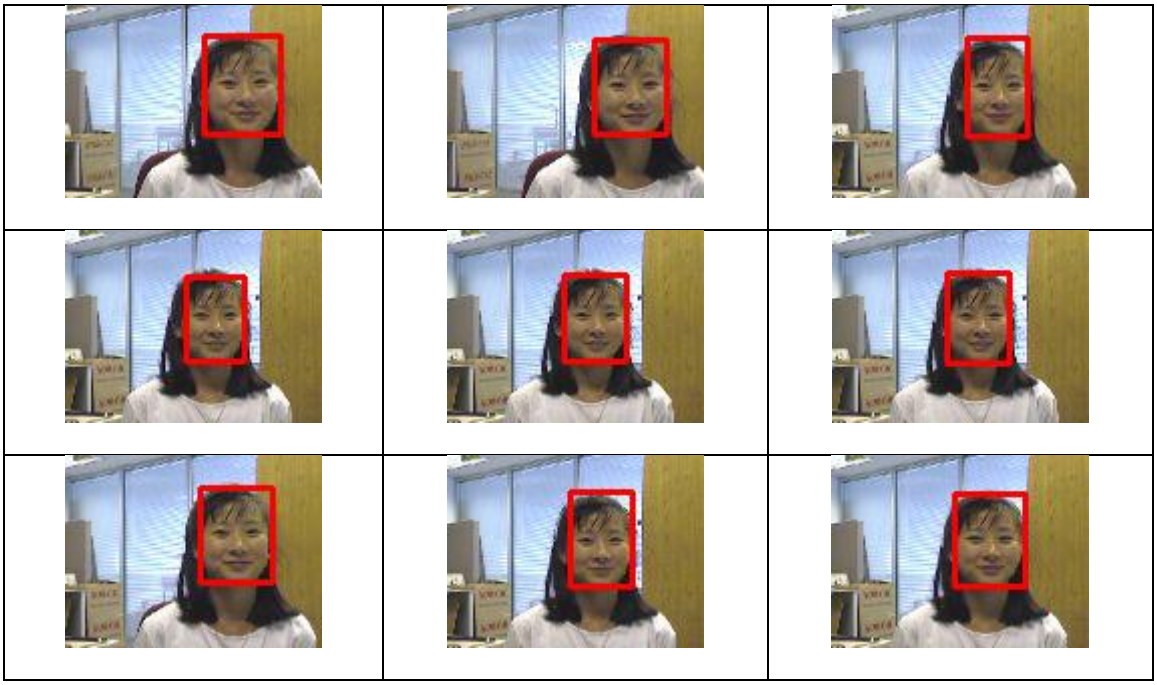


Tabla 3. Los cuadros rojos son las regiones que se siguen a través de los frames.

Actualmente con las nuevas computadoras, cámaras más especializadas y la necesidad de evaluar videos automáticamente se ha creado un interés en los algoritmos de seguimiento [53]. El uso del seguimiento de objetos se utiliza en los siguientes campos:

- Detección de objetos automáticamente.
- Vigilancia automática, que consiste en monitorear una escena para detectar actividades sospechosas.
- Interacción humano computadora, reconocimiento de gestos, seguimiento de ojos etc.
- Monitoreo del tráfico, es decir, estadísticas en tiempo real del tráfico para dirigir el flujo del tráfico.
- Navegación automática de vehículos, es decir capacidades de video vigilancia basadas en planeación de rutas y evitar obstáculos.

Hay tres puntos en el análisis de videos: detección de movimiento de objetos, seguimiento de los objetos frame a frame, y el análisis de comportamiento en el seguimiento de objetos. Es por tal motivo que se han creado algoritmos seguidores para dar respuesta a los problemas mencionados, más concretamente con algoritmos seguidores y tomando en cuenta el uso final para crear un seguidor.

Entonces un seguidor asigna etiquetas consistentes a los objetos seguidos en los diferentes frames que componen el video. Adicionalmente el seguidor puede dar información como la orientación, área o forma del objeto. Ahora, los objetos seguidos pueden ser complejos debido a los siguientes factores:

- Perdida de información por la proyección del mundo 3d a una imagen 2d.
- Imágenes con ruido.
- Comportamientos complejos de los objetos.
- Parcial y completa oclusión.
- Formas de objetos complejos.
- Cambios de iluminación.
- Requerimientos en tiempo real.

En general el adecuado funcionamiento de un seguidor depende de la apariencia del objeto, formas del objeto, numero de objetos, movimientos del objeto, movimiento de la cámara y las condiciones de iluminación.

1.13.1 Representación de objetos

En un escenario de seguimiento un objeto puede definirse como algo que es de interés para un análisis más adelante donde los objetos pueden ser representados por sus apariencias y formas [53]. A continuación se exponen las representaciones más comunes de objetos.

- Puntos: El objeto es representado por un único punto central o por un conjunto de puntos. Esta representación es aceptable para seguir objetos que ocupan regiones pequeñas en una imagen.
- Formas geométricas primitivas: La forma del objeto es representada mediante elipses, rectángulos, etc. Es más fácil para representar objetos no rígidos.
- Silueta del objeto y contorno: La representación de contorno representa el perímetro de objeto, la región dentro del contorno se llama silueta del objeto. Es aceptable para seguir formas no rígidas.
- Modelos de forma articulados: Los objetos articulados están formados por partes del cuerpo que se mantienen juntas con articulaciones. Se pueden modelar las partes con cilindros, elipses, etc.
- Modelos esqueléticos: Pueden ser extraídos por aplicar transformación de ejes medios a la silueta del objeto. Este modelo puede ser usado para modelar objetos articulados y objetos rígidos.

1.13.2 Representación de apariencia

Existen numerosos caminos para representar las características de apariencia del objeto, se debe tomar en cuenta que las representaciones de objeto pueden ser combinadas con las representaciones de apariencia para seguir. Algunas representaciones de apariencia son:

- Densidad de probabilidad de la apariencia del objeto: La densidad de probabilidad del objeto puede ser gaussiana, mezcla de gaussianas o no paramétrica como histogramas.
- Plantillas: Se forman usando formas geométricas simples o siluetas. Sin embargo solo codifican la apariencia generada por una sola vista. Entonces las plantillas son solamente útiles cuando las posiciones no varían considerablemente durante el seguimiento.
- Modelos de apariencia activa: Son generados por simultáneamente modelar la forma del objeto y apariencia creando puntos de referencia que pueden residir en el perímetro del objeto o alternativamente residir dentro de la región del objeto. Para cada punto de referencia un vector de apariencia es guardado en la forma de color, textura o magnitud del gradiente. Además requieren una fase de entrenamiento donde forma y apariencia es aprendida de un conjunto de ejemplos usando el análisis de componentes principales.
- Modelos de apariencia multivista: Estos modelos codifican diferentes vistas de un objeto. Un enfoque para representar las diferentes vistas del objeto es generar un sub espacio de las vistas dadas. Por ejemplo el análisis de componentes principales (PCA) y el análisis de componentes independientes (ICA) han sido usados para forma y representación de apariencia.
- Otro enfoque para aprender diferentes vistas de un objeto es entrenar un conjunto de clasificadores por ejemplo máquinas de soporte vectorial.

En genera hay una fuerte relación entre la representación del objeto y el algoritmo de seguimiento.

1.13.3 Selección de características para seguir

Seleccionar las características correctas juega un rol crítico en el seguimiento, en general la más deseable propiedad de una característica visual es la unicidad tal que el objeto pueda ser fácilmente distinguido en el espacio de características. La selección de características es cercanamente relacionada a la representación del objeto. Por ejemplo, el color es usado como característica base para representaciones de apariencia para histogramas, mientras que para representaciones base contorno se usan bordes de objetos como característica. Los detalles de características visuales más comunes son:

- Color: En procesamiento de imágenes el espacio de color RGB (red, green, blue) es comúnmente usado para representar el color. También se puede utilizar el espacio de color HSV (hue, saturation, value), sin embargo no hay una última palabra para decidir cuál espacio de color es más eficiente.
- Bordes: Los bordes de objetos usualmente generan fuertes cambios en la intensidad de imágenes, la detección de bordes es usada para detectar esos cambios. Una propiedad importante de los bordes es que ellos son menos sensitivos a cambios de la iluminación comparada con las características de color. Algoritmos que siguen los bordes de los objetos usualmente usan bordes como la característica representativa.
- Flujo óptico: Es un campo de vectores de desplazamiento los cuales definen la traslación de cada pixel en una región. Esto es calculado usando la restricción de brillo la cual asume la constancia de brillo de pixeles correspondientes en frames consecutivos. El flujo óptico es comúnmente usado como característica basadas en movimiento como segmentación o seguimiento.
- Textura: Es una medida de la variación de intensidad de una superficie la cual cuantifica propiedades tales como suavidad y regularidad. La textura necesita un paso de procesamiento para generar los descriptores. Además las características de textura son menos sensitivas a los cambios de iluminación que las características de color.

El usuario debe elegir que característica se adapta mejor para su algoritmo, sin embargo la selección automática de características ha recibido atención significativa en la comunidad del reconocimiento de patrones. Los métodos de selección automática de características se dividen métodos de filtros y métodos de envoltura. Los métodos de filtro intentan seleccionar las características basándose en un criterio general, por ejemplo las características deberían ser no correlacionadas. Los métodos de envoltura seleccionan las características basadas en la utilidad de las características en un dominio específico de problema, por ejemplo la clasificación de la ejecución usando un subconjunto de características.

1.13.4 Detección de objetos

Todo método de seguimiento requiere un mecanismo de detección de objeto para cada frame o cuando el objeto aparece por primera vez en un frame [53]. Sin embargo, algunos métodos de detección de objetos hacen el uso de información temporal calculada de una secuencia de frames para reducir el número de falsas detecciones. Esta información temporal es usualmente en la forma de frames diferenciados, los cuales alertan cuando cambian las regiones en frames consecutivos. Dadas las regiones del objeto en la imagen, es entonces tarea del seguidor ejecutar la correspondencia del objeto desde un frame al siguiente frame generando el seguimiento. Algunos detectores de objetos son los siguientes:

- Detector de puntos: Son usados para encontrar puntos de interés en imágenes las cuales tienen una textura expresiva en sus respectivas locaciones. Los puntos de interés han sido ampliamente usados en el contexto de problemas de movimiento y seguimiento. Una cualidad deseable de un punto de interés es la invariancia a cambios de iluminación y puntos de vista de la cámara.
- Substracción de fondo: La detección de objetos puede ser realizada por construir una representación de la escena llamado modelo de fondo y el cual encuentra desviaciones para cada frame entrante donde algún cambio significativo en una región de imagen significa un objeto moviéndose.
- Segmentación: Los algoritmos de segmentación particionan la imagen a regiones perceptualmente similares. Cada algoritmo de segmentación direcciona dos problemas, el criterio de una buena partición y el método para alcanzar una partición eficiente.
- Aprendizaje supervisado: La detección de objetos puede ser ejecutada por aprender diferentes vistas de objetos automáticamente de un conjunto de ejemplos por un mecanismo de aprendizaje supervisado.

1.13.5 Categorías de seguimiento

Un seguidor de objetos debe generar la trayectoria de un objeto sobre el tiempo mediante la localización de su posición en cada frame del video. El seguidor de objetos puede también proveer la región completa en la imagen que es ocupada por el objeto en cada instante del tiempo. La tarea de detectar el objeto y establecer correspondencia entre las instancias del objeto a través de los frames puede ser realizada separada o junta. En el caso separado, la posible región de objetos en cada frame es obtenida por medida de un algoritmo de detección de objetos y entonces el seguidor da correspondencia a los objetos a través de los frames. Para el caso de juntos, la región del objeto y la correspondencia es juntamente estimada por iterativamente actualizar la región de la ubicación del objeto y la región de información obtenida de los frames previos. A continuación se listan algunas categorías de seguimiento:

- Puntos seguidores: Objetos detectados en frames consecutivos son representados por puntos y la asociación de puntos está basado en el estado del objeto previo el cual puede incluir la posición y el movimiento del objeto.
- Seguimiento por núcleos: El núcleo hace referencia a la forma del objeto y su apariencia. Por ejemplo el núcleo puede ser una plantilla rectangular o en una forma elíptica con un histograma asociado. Los objetos son seguidos por calcular el movimiento del núcleo en frames consecutivos.
- Seguimiento de siluetas: El seguimiento es ejecutado por estimar las regiones del objeto en cada frame. Estos métodos usan la forma de la densidad de apariencia y modelos de forma los cuales se usan en mapas de bordes.

CAPÍTULO 2
TRABAJO RELACIONADO

En este capítulo se exponen algunas aplicaciones que combinan seguimiento de imagenes con deep learning. Generalmente las aplicaciones que más auge tienen en la actualidad son las que se utilizan en la vigilancia, seguimiento de objetivos y en la identificación facial. También existen otras aplicaciones como la conducción automática y además existen aplicaciones para la salud o para procesamiento del lenguaje.

2.1 Clasificadores que fusionan seguimiento con clasificación

Es importante mencionar que existen trabajos de investigación que utilizan los resultados de la clasificación junto con el seguimiento para crear aplicaciones robustas como por ejemplo la detección y seguimiento de personas con drones o la conducción automática de vehículos, los cuales son temas que tienen un amplio campo de investigación en la vigilancia de personas. A continuación se exponen algunos de estos ejemplos:

2.2 Seguidores de personas

Una arquitectura denominada Deep Sort [8] que a través de deep learning y filtros de Kalman asigna un número a las personas que caminan sobre una calle, este número debe ser constante resultado de hacer seguimiento. Este trabajo surgió para el enfoque MOT de Alemania [9]. En la Figura 25 podemos ver un ejemplo.

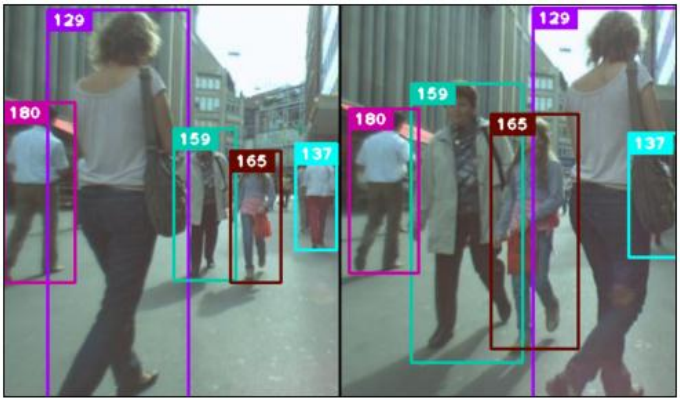


Figura 25. Deep Sort sigue personas en la base MOT.

En [28] se propone un enfoque de red neuronal recurrente que explota el historial de las locaciones tan bien como las características visuales distintivas aprendidas por las redes neuronales. El trabajo estudió la capacidad de regresión de LSTM [60] (Long Short-Term Memory) en el dominio temporal, y propuso concatenar características visuales de alto nivel producidas por las redes convolucionales con información de región. Se usa la regresión para la predicción directa de las locaciones del seguimiento en la capa convolucional y en la unidad recurrente. En la Figura 26 podemos ver un ejemplo.

Figura 26. Ejemplo donde el cuadro verde representa el resultado del trabajo, en azul el resultado de YOLO y el cuadro en rojo es ground truth.

El trabajo en [29] plantea un enfoque que incorpora una arquitectura deep learning con un framework en línea AdaBoost para seguir objetos en fondos complejos. Se propone un autoencoder de eliminación de ruido apilado (Stacked Denoising Autoencoder SDAE [61]) que es usado para aprender descriptores de características multi-nivel de un conjunto de imágenes auxiliares. Cada capa de SDAE representa un espació de características diferentes y después es transformado a un clasificador de red neuronal profunda para discriminar objetos/fondo por añadir una capa de clasificación. Lo siguiente es utilizar el framework de selección de características en-línea con el conjunto del clasificador DNN el cual es actualizado para robustamente distinguir entre el objetivo y el fondo. En la Figura 27 podemos ver un ejemplo.

Figura 27. El bounding box en amarillo es el resultado, los demás son otros seguidores.

En [30] se aborda una nueva arquitectura de red neuronal profunda que puede resolver el problema de asociación de datos con un numero variable de seguidores y detecciones incluyendo falsos positivos. La red consiste en dos partes un codificador y un decodificador. El codificador es la red completamente conectada con múltiples capas que toma bounding boxes de detecciones e historiales de seguimientos como entradas. Las salidas del codificador son secuencialmente alimentadas dentro del decodificador el cual se compone de una red bi-direccional LTSM (Long Short-Term Memory) con una capa de proyección. La salida final de red es una matriz de asociación que refleja puntos que coinciden entre seguimientos y clasificaciones. En la Figura 28 podemos ver un ejemplo.

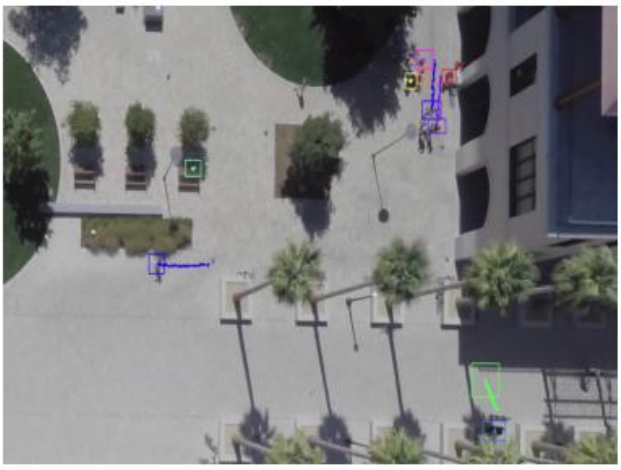


Figura 28. Imagen capturada con un Dron para hacer seguimiento aéreo.

El trabajo en [32] propone un enfoque de seguimiento de dos etapas multi-objeto para seguir peatones en escenarios complejos. Lo primero es generar segmentos parciales de seguimiento de alta confidencia (tracklets) usando un detector robusto de peatones para después asociarlos usando descriptores y rutas potenciales entre tracklets. Es un trabajo acerca de la video vigilancia pues actualmente es un tema que ha tomado gran interés en el seguimiento de peatones. En la Figura 29 podemos ver un ejemplo.

Figura 29. Los cuadros en amarillo y magenta son tracklets y los cuadros en rojo residuos.

En [33] se presenta un enfoque de seguimiento y detección de objetos que introduce un aumento en el algoritmo de resta de fondo que usa seguimiento de característica de esquina como señal, un seguimiento de características y un algoritmo que agrupa los resultados de la resta del fondo. El seguimiento de objetos y la detección tiene áreas de aplicación como los sistemas de transportes inteligentes que mejoran la operación y seguridad del transporte [34]. En la Figura 30 podemos ver un ejemplo.

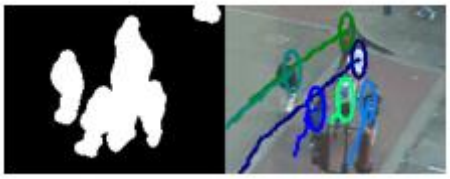


Figura 30. Detección y seguimiento de peatones.

2.3 Seguidores de autos

La conducción automática permite al automóvil manejarse por sí mismo mediante la percepción del ambiente y de la ejecución de una acción sensible en tiempo real [54]. También procesan una gran cantidad de datos de sus sensores que son internos y externos como cámaras o sensores de movimiento. A continuación se explican algunos de estos ejemplos.

El trabajo en [31] trata acerca de un prototipo que reconoce transeúntes y sigue, predice localización de vehículos en un ambiente urbano. Para la medición se utilizó un sensor de laser multicapa que se colocó en el parachoques del auto. La clasificación de los transeúntes se realiza a través de algoritmos de conocimiento a priori y heurísticas lo que permite seguirlos y clasificarlos incluso a largas distancias. Utiliza la ganancia de filtros de kalman y puntos de referencia de objetos para realizar el seguimiento. En la Figura 31 podemos ver un ejemplo.

Figura 31. Segmentos asociados de video seguidos por distancia donde el sensor muestra un numero de capas escaneadas en diferentes colores.

[55] propone como un experimento la creación de un automóvil autónomo que recorre París. Utiliza la detección de objetos mediante varios sensores instalados en el auto además de contar con un conductor todo el

tiempo. Para la detección siempre se conoce su posición en todo momento, cuenta con diversos sensores como cámaras, antena GNS, sensor LiDar multicapa, sensor de odometría, unidad inercial y sensores mono LiDar. En la Figura 32 se observa este vehículo.

Figura 32. Vehículo autónomo Navya.

El trabajo en [56] propone un autobús-tren autónomo implementado por la empresa china CRRC la empresa más grande de material ferroviario en el mundo para brindar transporte y evitar la congestión automovilística que tiene China en algunas de sus ciudades. Además es un medio de transporte que funciona con electricidad y su trayectoria se aplica con sensores LiDar y GPS que se ubican en la parte frontal y laterales los cuales se guían con unas líneas que están en el centro de la carretera y otras líneas laterales que indican la proximidad con otros vehículos. En la figura 33 se observa al vehículo en una de sus estaciones.

Figura 33. Vehículo autónomo.

ENFOQUE DEEP TRACK VISION

CAPÍTULO 3

En este capítulo se presenta un enfoque para desarrollar el sistema que se propone donde lo primero es utilizar un modelo pre-entrenado llamado Mobilenet SSD que clasifica las personas en la imagen y otro modelo pre-entrenado denominado Inception que detecta objetos en una región especifica. Al tener una persona y un objeto en una región específica se comienza a realizar seguimiento (tracking) a la persona mediante el algoritmo de seguimiento CRST creando una relación persona-objeto y finalmente mediante una condición de distancia se propone mostrar el mensaje de alerta. A continuación se expone más a fondo estos temas y la razón de porque utilizarlos para este proyecto, es decir, este capítulo expone el algoritmo DTV, las regiones que estudia para generar una respuesta y las tecnologías que utiliza.

3.1 Descripción general del algoritmo Deep Track Vision

Olvidar objetos es una situación muy común, específicamente cuando se trata de objetos con cierto valor como celulares, tablets, libros etc. Es un problema que puede pasar en cualquier momento, sin embargo se puede proponer un espacio libre de olvidos a través de aplicaciones tecnológicas que es la aplicación que considera este trabajo de tesis. En particular se presenta una solución con la cual es posible evitar olvidar objetos gracias a una alerta que se muestra en una pantalla. El algoritmo denominado Deep Track Vision o DTV une deep learning y el seguimiento de objetos para crear una aplicación y para probar este mismo se propone un modo de utilizarlo ya que es muy específico en las regiones que se manejan, es decir, es un espacio controlado para tratar de evitar olvidar objetos en una región específica.

La idea general propone mostrar un mensaje de alerta en una pantalla dependiendo de la evaluación de la situación donde hay una persona y que además dejó algún objeto sobre una superficie para así tener registro del objeto.

Para llevar a cabo el proceso descrito anteriormente se propone el uso de modelos pre-entrenados deep learning Caffe como Mobilenet-SSD para ubicar la clase persona e Inception para ubicar las clases de objetos variados de imágenes y el algoritmo seguidor de regiones de imágenes CRS-DFC (Anexo B) o CRST que actúan cada uno como una caja negra ya que solo se utilizan sus resultados contenidos en librerías. Estas decisiones se tomaron a través de búsqueda y recomendaciones de la página de deep learning pymagesearch [6].

Ahora bien un modelo pre-entrenado en deep learning es un archivo compuesto por pesos ya entrenados en alguna base de datos y una arquitectura definida que se puede utilizar para hacer evaluaciones. Hay modelos pre-entrenados en los diferentes frameworks como Tensorflow [7] [4], Keras [42], Opencv [24], Caffe [43], Theano [44], Pythorch [45]. Sin embargo este proyecto utiliza como base el lenguaje python del software Anaconda junto con las bibliotecas Opencv en su versión 4.0, numpy [46], math y os. Cabe aclarar que Caffe es un framework de deep learning que permite definir, crear y testear modelos de aprendizaje profundo aunque en este trabajo para clasificar con sus modelos solo se utiliza Opencv que tiene soporte para realizar esta acción.

Para el caso de los algoritmos seguidores estos permiten seguir una región especifica dentro de una imagen más grande a través de todos los frames que pueda contener un video, por lo cual estos mismos han adquirido gran relevancia en temas como la vigilancia o la conducción automática lo que ha permitido que exista un campo de investigación respecto a estos temas por ejemplo los filtros gaussianos, filtros de kalman y filtros por dimensiones.

Precisamente es el caso del algoritmo CRST el cual es un filtro por canales dimensionales y que se utiliza para este proyecto pero solo se utilizan sus resultados ya que Opencv permite utilizar este seguidor solo con algunos métodos actuando también como caja negra pues solo da resultados.

A continuación se detalla por separado partes específicas que se unen para crear el algoritmo.

3.2 Persona

El modelo pre-entrenado de *Mobilenet SSD* (Anexo B) para Caffe es propuesto en github en la dirección de internet [39]. Este modelo identifica entre 21 clases sin embargo se descartan 20 para solo clasificar la clase persona junto con su confidencia, es decir, que tanto la red predice que es una persona. A continuación se observa en la Tabla 5 el resultado al evaluar el modelo en python sobre imágenes lo cual es un bounding box de la clase persona.

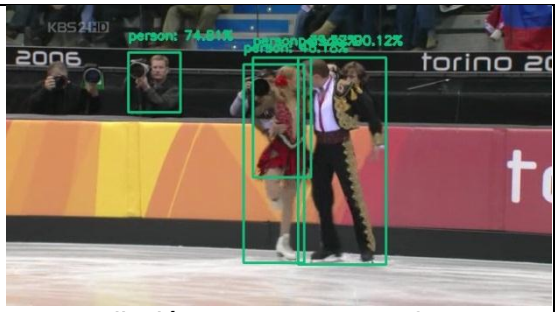
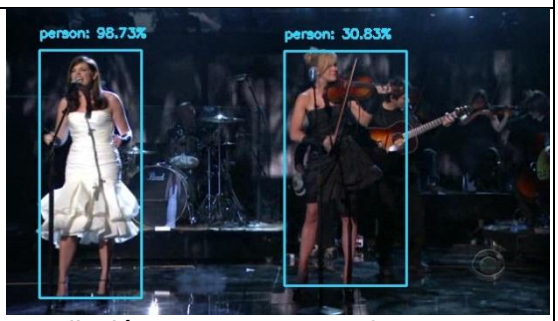
	 Predicción 4 personas, conf 74%
	 Predicción 2 personas, conf 98% y 30%
	 Predicción 1 persona, conf 98%

Tabla 4. Clasificaciones persona hechas con Mobilenet-SSD, junto con coincidencia conf.

3.3 Objetos variados

El Modelo *zoo Caffe googlenet* es un modelo pre-entrenado basado en Inception (Anexo B), el número de clases que puede reconocer es 1000 además muestra la confidencia que expone cuanto la red está segura de la clasificación del objeto, para este trabajo se propone reconocer el cambio entre si un objeto aparece en una región. Se puede descargar su modelo y su archivo de protocolo desde la página de Opencv [40]. Para toda la imagen se da solo una clasificación, en la Tabla 6 podemos ver ejemplos.

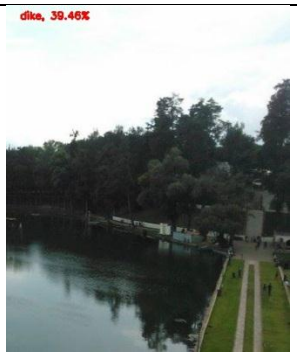
 Sala de baño	 Predicción lavamanos, conf 65%
 Computadora	 Predicción computadora, conf 65%
 Lago	 Predicción dique, conf 39%
 Concierto	 Predicción escenario, conf 28%

Tabla 5. Muestras evaluadas con Inception, junto con su coincidencia conf.

La razón de usar este modelo es porque en todo momento asigna una clasificación a cada imagen que es evaluada aunque no corresponda a una clasificación correcta. Esto se utiliza para crear el vector de inicialización y comparar los resultados para la región de Inception. La Tabla 7 muestra algunos ejemplos.

 Señal de salida	 Predicción reloj, conf 33%
 Planta girasol	 Predicción abeja, conf 57%
 Cepillo dental	 Predicción destornillador, conf 35%
 Sala	 Predicción teatro, conf 78%

Tabla 6. Algunos ejemplos clasificados erróneamente por Inception, junto con su coincidencia conf.

3.4 Algoritmo de seguimiento CRST

Opencv permite utilizar métodos específicos de algoritmos seguidores en conjuntos de imágenes. Para este proyecto se propone utilizar el algoritmo seguidor CRST que es la implementación de CSR-DFC (Anexo B). En este trabajo se utiliza este algoritmo para seguir y saber la posición del objetivo persona en un conjunto de imágenes. Para utilizarlo en este trabajo se propone

seleccionar el área de la persona como la región inicial del algoritmo seguidor, después con métodos de Opencv se actualiza la región del objetivo para cada nueva imagen obteniendo un nuevo bounding box del objetivo. Además con cada nueva región se propone ubicar, guardar y actualizar el punto central de la región CRST con el fin de saber su posición. La Tabla 7 muestra una colección de frames consecutivos que siguen la región de la cara de una mujer con un rectángulo.

Tabla 7. Frames que son seguidos con CRST.

3.5 Distancia

La métrica que se eligió para determinar la distancia y así poder saber si se alejaba la persona del objeto entre los pixeles es la distancia euclidiana. Los puntos que se toman son el punto central de la región CRST-persona y el punto central de la región Inception. A continuación se presenta la fórmula.

$$distancia = \sqrt{(x_{inception} - x_{crst})^2 + (y_{inception} - y_{crst})^2} \qquad (37)$$

3.6 Regiones del algoritmo DTV

La Figura 34 representa cómo son las regiones que estudia este proyecto sobre una imagen o frame individual con sus dimensiones, la región negra es clasificada por el modelo de Mobilenet SSD para persona además del espacio donde también se presentará CRST (se denomina región Mobilenet SSD), y la región azul es clasificada por Inception para objetos (denominada también como la región Inception).

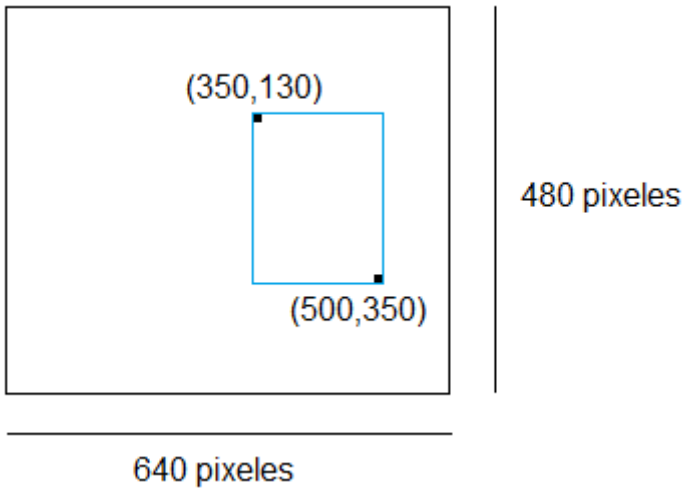


Figura 34. Las regiones para clasificar, la región azul se clasifica con Inception, la región negra se clasifica con Mobilenet-SSD.

3.7 Algoritmo Deep Track Vision o DTV

DTV es un algoritmo que se creó con el objetivo de evitar olvidar objetos en regiones bien definidas que se proponen en esta tesis. Por lo tanto el algoritmo que se propone debe responder a ciertas cuestiones como por ejemplo saber cuándo un objeto se presenta en una región, clasificar objetos variados, realizar seguimiento, saber que hay una persona y tomar una decisión para mostrar un mensaje. La idea principal para lograr todo esto fue utilizar la clasificación para identificar clases donde la respuesta fue utilizar deep learning que clasifica muy bien los objetos, sin embargo el trabajo se topó con que este solo clasificaba para ciertos frames por lo que se propuso utilizar la tecnología del seguimiento para todos los frames, es decir, utilizar un seguidor que no perdiera el objetivo en ninguno de los frames. Por eso se decidió utilizar deep learning y algoritmos de seguimiento pues daban una buena respuesta a la clasificación y al seguimiento. Ahora para decidir cómo mostrar el mensaje se propone utilizar la distancia euclidiana pues esta se facilitó en utilizar los pixeles centrales entre la región rígida de Inception (la región pequeña) con la región del seguidor de la persona CRST y mostrar el mensaje sobre la imagen al superar un límite de distancia definido. Entonces el programa propuesto utiliza estas tecnologías para realizar una respuesta además de programación extra para realizar la funcionalidad. A continuación se muestra la ventana principal en la Figura 35 que contiene diferentes zonas para interactuar con el proyecto como son pruebas, aplicativos, parámetros e información.

Figura 35. Ventana principal del algoritmo DTV.

De manera general este proyecto aplica el algoritmo DTV en donde primero se configuran los parámetros y después se presiona alguno de los botones que inician con la palabra Por, sin embargo esto se puede omitir ya que existen parámetros por default. Al presionar entonces un botón saldrá una ventana para seleccionar la carpeta donde se encuentre el video, una carpeta con los frames que representan un video o directamente ejecutar con una cámara, en alguno de los casos que se oprima el botón el algoritmo DTV comenzará primero inicializándose con los datos de los parámetros y emergerá una pantalla con las clasificaciones de persona Mobilenet SSD y la región estática Inception. Si se superan los límites que se definieron en los parámetros se mostrarán los resultados sobre la pantalla. A continuación se muestra el algoritmo en la Figura 36.

```
Algoritmo Deep Track Vision
1: function DTV(lobjeto,lpersona,linicialización,ldist)
2:    Carga de modelos deep learning
3:    Crear vector de inicialización de tipo strings
4:    for i=0, linicialización, 1 do
5:       Analizar con Inception
6:    end for
7:    if Evaluacion es diferente a todas then
8:       Se anexa al vector
9:    end if
10:   Crear el seguidor
11:   Crear banderas
12:   while Terminar frames o q do
13:      Leer Frame
14:      Redimensionar Frame
15:      if redmobilenet=0 then
16:         Mobilenet evalua frame
17:         if persona=1 then
18:            Aumentar contpersona
19:            if contpersona=lpersona then
20:               Banderas aumentan en 1, act=1, banpersona=1
21:            end if
22:         end if
23:      end if
24:      if act=1 then
25:         Se anexa seguidor, actualizartrack=1, persona=1, redmobilenet=1
26:      end if
27:      if actualizartrack=1 then
28:         Se actualiza el seguidor
29:      end if
30:      cam=frame
31:      if redinception=0 then
32:         Seleccionar región de inception
33:         Inception evalúa el frame
34:         La evaluación se compara con el vector de inicialización
35:         if evaluación de Inception != vector de inicialización then
36:            Aumentar limiteobjetos
37:         end if
38:         if limiteobjetos=lobjetos then
39:            banderacosa=1, redinception=1, cosa=1
40:         end if
41:      end if
42:      if persona=1 and cosa=1 then
43:         Dibujar línea entre los centros del seguidor e inception
44:         Calcular la distancia euclidiana entre los centros
45:         if distancia > ldist then
46:            Mostrar mensaje de olvido
47:         else
48:            Mostrar mensaje de todo en orden
49:         end if
50:      end if
51:   end while
52: end function
```

Figura 36. Algoritmo DTV.

En general el algoritmo se lleva a cabo por la acción de un botón y parámetros que están en la ventana principal, la cual también permite modificar los parámetros y realizar algunas pruebas.

Entonces el algoritmo propuesto lo primero que realiza es inicializar un vector de clasificaciones sobre la región de Inception en un número definido de iteraciones definidas en la ventana principal, esto con el motivo de tener un vector el cual servirá como referencia de comparación en las siguientes etapas del algoritmo. (Lineas 2-9 del algoritmo).

Luego se inicia un ciclo que clasifica personas y objetos, el cual es la parte más importante del algoritmo pues dentro de este ciclo se utilizarán las clasificaciones, se creará el seguidor y se tomará la decisión de mostrar el mensaje según la evaluación de distancia. Este ciclo termina cuando se acaban los frames o por una orden de salida. (Lineas 12-51 del algoritmo).

Después para crear el seguidor primero se evalúa la parte de clasificar personas o la región de Mobilenet SSD y cada vez que se detecte una persona se acercara al límite que se definió en la ventana principal, que al llegar al límite se creará el seguidor y se desactivará el código para seguir clasificando personas, además se activará una bandera para indicar que se encontró una persona. También en adelante para cada nuevo frame se actualizará el seguidor. (Lineas 15-29 del algoritmo).

Ahora la parte del código que se utiliza en la región de Inception para clasificar objetos variados primero obtendrá la clasificación y después la comparará con todos los elementos del vector de inicialización y cada vez que encuentre una clasificación de objeto diferente se acercará a su límite definido en la ventana principal que al superarse se desactivará la clasificación con Inception y se activará la bandera de objeto. (Lineas 31-41 del algoritmo).

Finalmente cuando las banderas del objeto y la persona están activadas se comenzará a medir la distancia euclidiana entre los puntos centrales del seguidor y la región Inception, cuando la distancia que se definió en la ventana principal se supera se mostrará en la imagen el mensaje de alerta. (Lineas 42-50 del algoritmo).

En resumen lo que se propone en esta tesis es mostrar que a través de encontrar una persona, crear el seguidor CRST y además se haya detectado que el objeto cambio a través de los frames en la región de Inception, se debe crear una relación del punto central de CRST y el punto central de la región de Inception con la distancia euclidiana e indicar el mensaje dependiendo del límite de distancia que se definió en la ventana principal. Por ejemplo la Figura 37 muestra resultados.

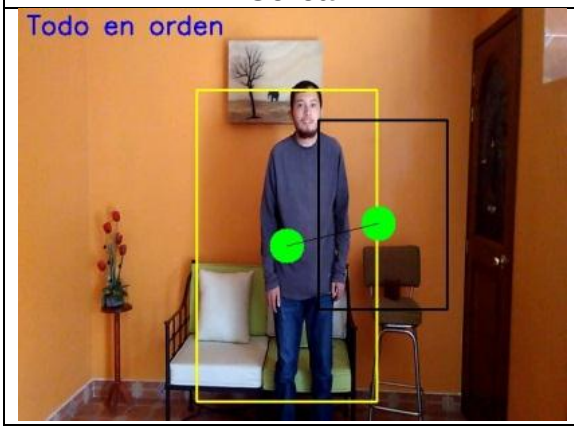
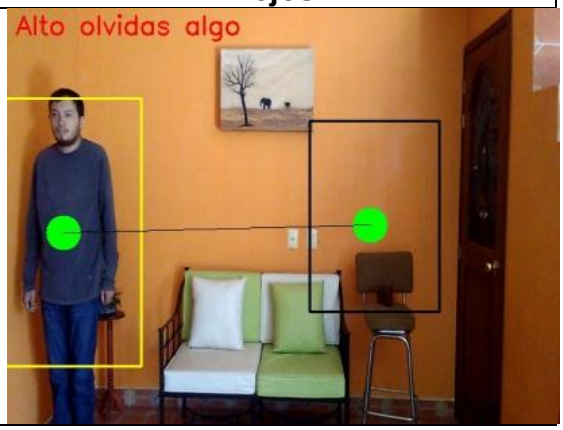
Cerca	Lejos
	

Figura 37. Resultados de aplicar DTV.

A continuación se describe la manera en que se aplica el algoritmo descrito anteriormente.

3.8 Fases para ejecutar el algoritmo y equipo de hardware

Para experimentar la propuesta presentada, se llevaron a cabo las siguientes fases:

- Se comienza a grabar con cámara una región cercana con una mesa o un lugar para colocar objetos y una silla durante 1.30 segundos, después entra la persona con su objeto, coloca el objeto en la región pequeña de Inception y la persona se sienta en la silla durante 30 segundos.
- Después la cámara se aleja para mostrar una región un poco más amplia de la escena de al menos 2.5 metros de alejamiento y entonces la persona se levanta, esta acción dura 1.30 segundos.
- La persona se aleja un poco con un paso lateral en un segundo y regresa a la silla igualmente con un paso lateral pero no se sienta, esta acción dura 2 segundos.
- Después la persona se aleja aún más que la primera vez con dos pasos laterales y finalmente regresa para tomar su objeto y salir del espacio de grabación, esta acción dura 4 segundos. La Tabla 8 muestra algunos frames del modo de uso.

El equipo utilizado para llevar a cabo las fases fue una laptop de 64 bits con una memoria ram de 8 GB. Además de ejecutarse en un entorno virtual de Anaconda.

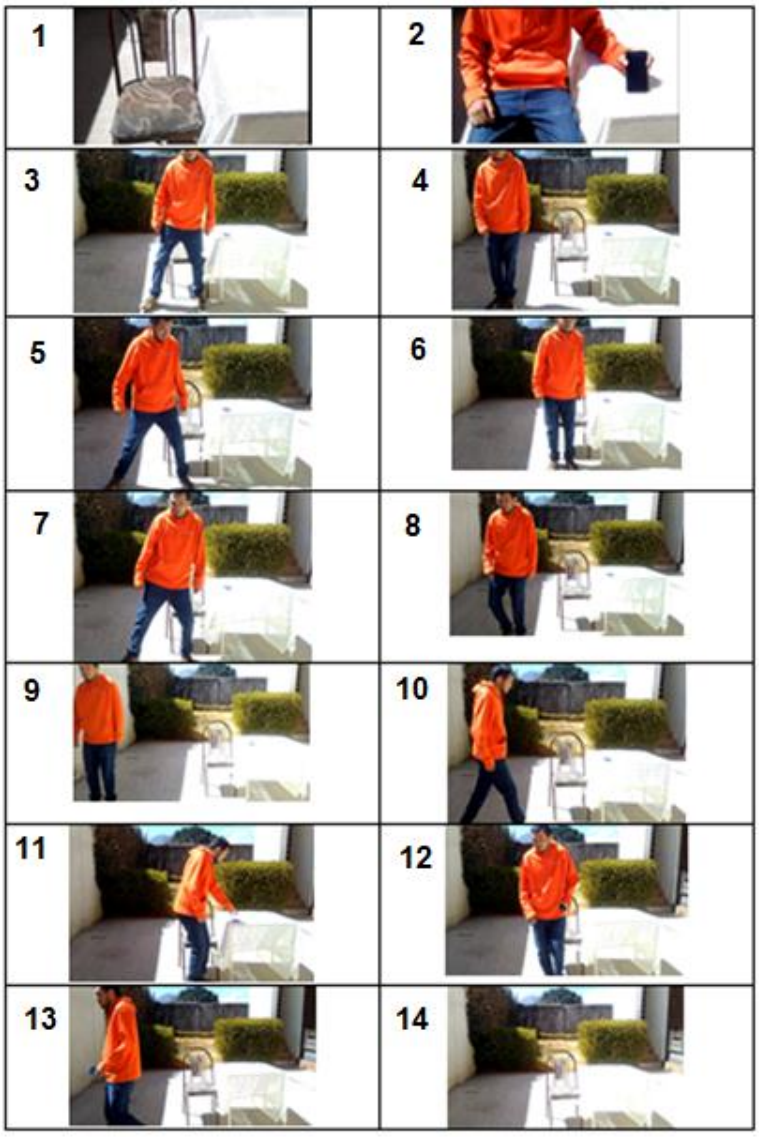


Tabla 8. Indicaciones de la manera de aplicar el algoritmo.

Entonces el algoritmo debe dar la respuesta a una observación sobre las fases para probar el algoritmo, es decir, ubicar una persona en una región y seguirla además de ubicar que hay un objeto nuevo en otra región, y después decidir si se muestra el mensaje de alerta.

A continuación se muestran algunas pruebas siguiendo las fases que propusieron para este trabajo.

3. 9 Resultados obtenidos

Mediante una serie de videos, se obtuvieron resultados los cuales se describen en esta sección, los videos fueron grabados con un celular con dimensiones de video 480x640 y aproximadamente cada video tiene de 650 a 700 frames con una duración de aproximadamente 20 segundos. Estos videos se inicializaron con los parámetros que propone por defecto el proyecto. Los videos se encuentran en [47].

A continuación en las Figuras 38, 39 y 40 se exponen algunos ejemplos evaluados en donde se muestra 1 persona la cual se mueve y se ven los resultados que a continuación se muestran. Primero se muestran los frames iniciales, después se muestran frames donde ya se está siguiendo la región de la persona y además la región de Inception ya detectó un cambio de objeto, finalmente se muestra el escenario cuando se cumple que el seguidor y la región de Inception están alejados.

a) Clasificación obtenida: broom (escoba)

b) Clasificación obtenida: couch (sofá)

Figura 38. Ejemplos clasificados con deep learning.

Aquí en la Figura 38 se puede observar como los primeros frames son evaluados por los modelos deep learning y estos obtienen una clasificación, la figura a) muestra la clasificación de la persona junto con un bounding box que selecciona su región además para la región de Inception también se obtiene una clasificación la cual es “broom” (escoba), para el caso de la imagen b) se obtiene la región segmentada de la persona clasificada por Mobilenet SSD y también la región de Inception que tiene asignada la clasificación de “couch” (sofá).

a) Traslape menor al 10% del bounding box entre la persona y el objeto de interés

b) Traslape mayor al 50% del bounding box entre la persona y el objeto de interés

Figura 39. Pruebas de distancia entre la persona y el objeto de interés.

En esta Figura 39 se puede observar la siguiente etapa en el algoritmo que es el punto de estudio de este trabajo el cual es alertar cuando la persona se aleja del objeto. Aquí se observa como la persona es seguida con un seguidor en color amarillo y que existe una distancia entre el punto central de la región Inception y el centro del seguidor amarillo. La imagen a y b representan el escenario donde la persona no esta tan alejada y por lo tanto se debe mostrar el mensaje de “Todo en orden” sobre la imagen.

a) Frame lejos

b) Frame lejos

Figura 40. Comparativa entre las regiones de interés con ausencia de traslape y respuesta del clasificador.

En la Figura 40 se puede observar el caso cuando la persona se aleja de la región Inception entonces sobre la imagen se coloca la oración “Alto olvidas algo” para indicar que se olvida el objeto pues la persona está alejada del objeto de interés.

Como se observa en los ejemplos evaluados, en los primeros frames se puede comprobar como clasifica las diferentes regiones de Mobilenet SSD e Inception. Además respecto al punto de referencia que es el centro de Inception muestra los resultados cerca y lejos con sus respectivos mensajes.

En la Figura 41 también se observan que los ejemplos cuando ya detectaron a la persona y al objeto entonces ahora solo se sigue con la métrica de la distancia euclidiana para decidir que mensajes mostrar.

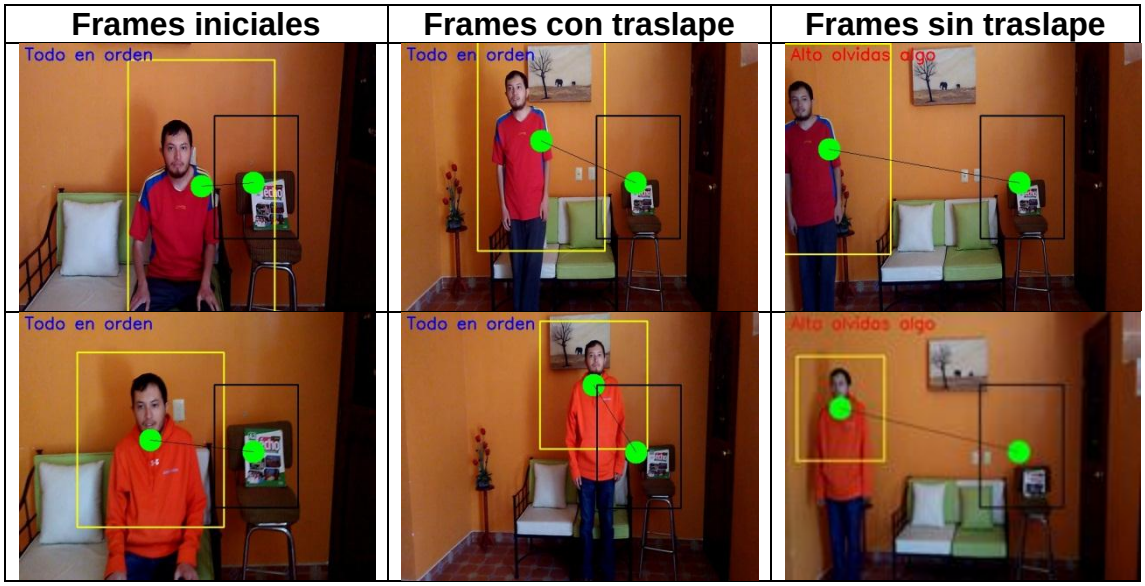


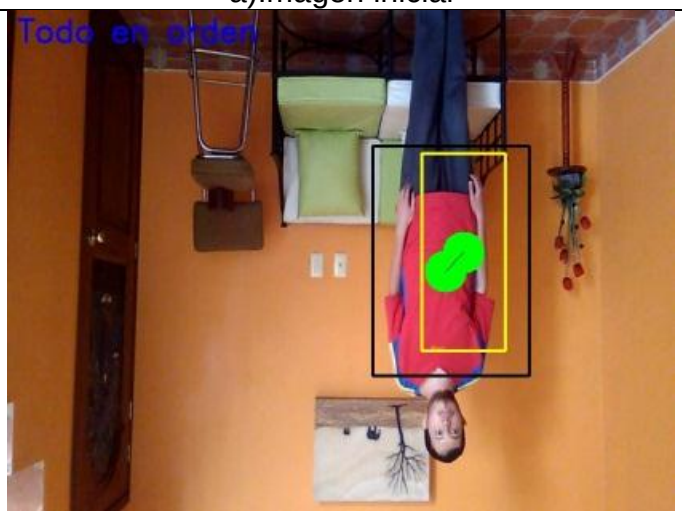
Figura 41. Pruebas de DTV con algunos videos.

En general se observa en estos ejemplos que el algoritmo es correcto para seguir en situaciones donde se aplica las fases de uso, cerca del centro de la región Inception muestra el mensaje de “Todo en orden” y lejos superando el límite de pixeles definido en la ventana principal del algoritmo muestra el mensaje “Alto olvidas algo”, logrando una respuesta satisfactoria de indicación de olvido de algún objeto. Además se puede observar que en los frames iniciales se realizan clasificaciones. Por lo tanto si se aplica el algoritmo en una región que cumpla la fase de uso permitirá que el algoritmo funcione correctamente y muestre sobre las imágenes los mensajes de alerta.

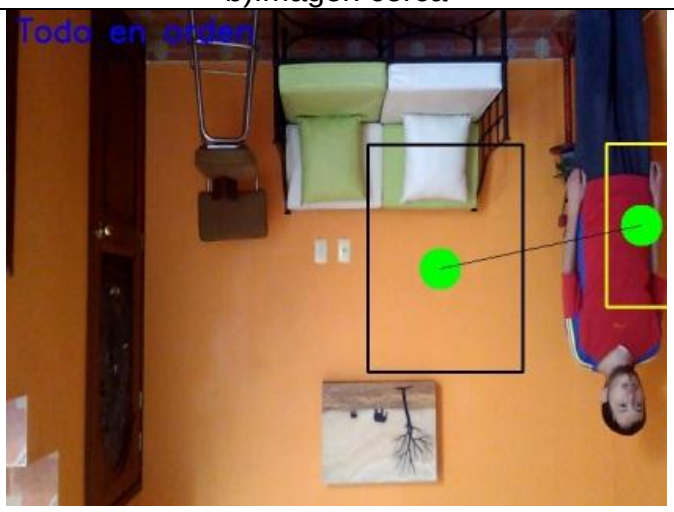
3.9.1 Pruebas en orden inverso de video

Se propone cargar videos 180 grados invertidos para verificar el comportamiento del algoritmo en esta situación. A continuación se muestran los resultados obtenidos para el video en la Figura 42.

a)Imagen inicial



b)Imagen cerca



c)Imagen lejos

Figura 42. Ejemplos evaluados de una manera invertida 180 grados.

Como se puede ver en la Figura 42 en su imagen inicial se detectan las clasificaciones, después en la imagen cerca se pueden ver los resultados que indican todo en orden en la parte invertida y finalmente en la imagen lejos se

puede ver también el mensaje de “Todo en orden” sin embargo esto es un error. A continuación en la Figura 43 se presentan más ejemplos evaluados.

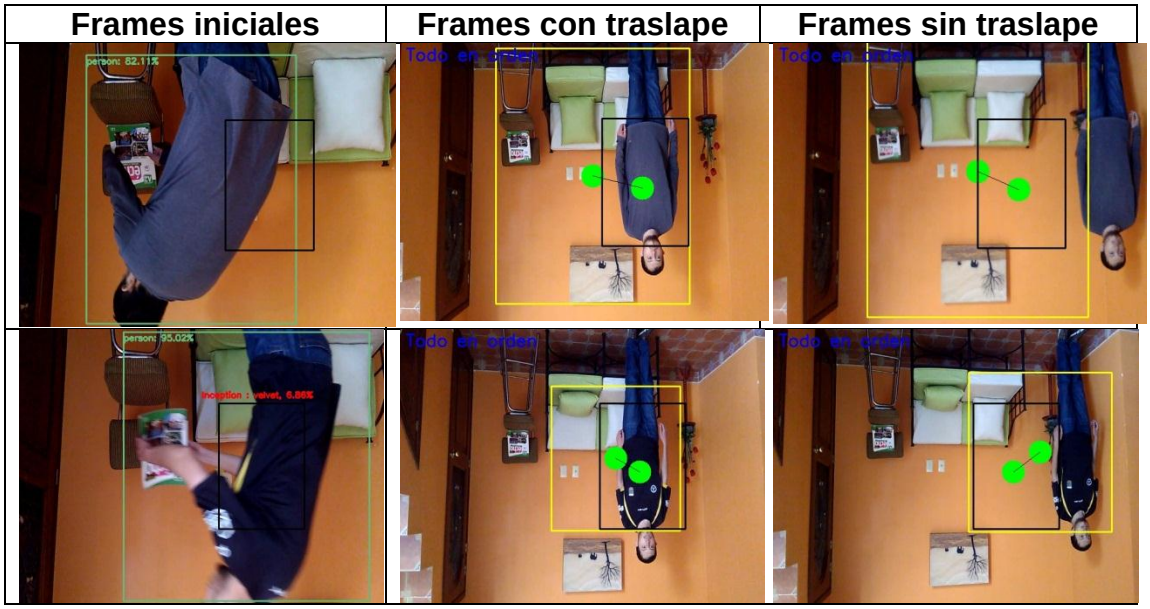


Figura 43. Ejemplos evaluados con DTV invertidos.

Estas pruebas invertidas muestran como resultado en los mensajes utilizando la fase de uso invertida 180 grados que todo estaba en orden lo cual es incorrecto. Clasificó personas invertidas y creó el seguidor CRST sin embargo la región de Inception se activó porque la persona paso directamente sobre esta región además en un ejemplo se perdió el seguimiento. Por lo tanto el algoritmo es sensible a tomar ejemplos invertidos 180 grados.

3.9.2 Pruebas con desenfoque

Se realiza con un kernel 5x5 y la función blur de Opencv. En general no modifica mucho en los resultados, es decir, sigue reconociendo los objetos y las personas de una forma óptima sin embargo en un caso se perdió el seguimiento, en la Figura 44 podemos ver este ejemplo.

a) Imagen inicial

b) Imagen cerca

c) Imagen lejos

Figura 44. DTV aplicado en frames con blur.

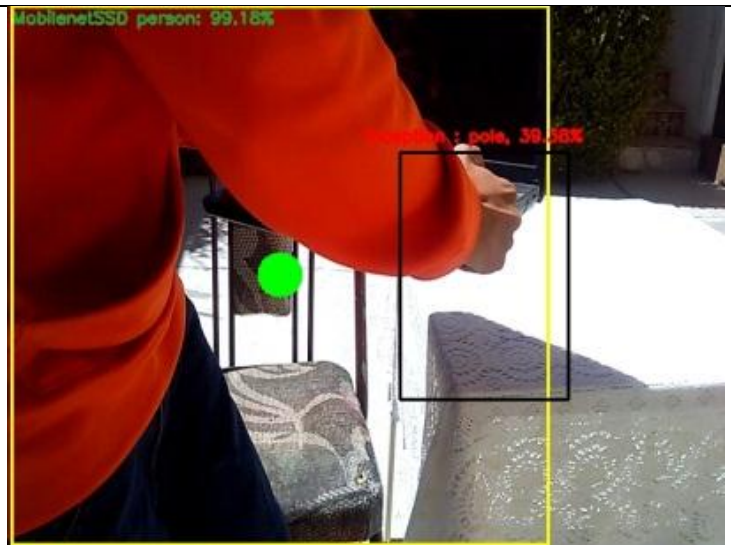
La Figura 44 es un ejemplo donde se pierde la capacidad de seguir el objeto, se clasifica bien desde el inicio en las diferentes regiones sin embargo el error de dejar de seguir es un error muy poco frecuente. A continuación se muestran más ejemplos en la Figura 45.

Figura 45. Ejemplos evaluados con DTV con desenfoque.

El desenfoque prácticamente no afecto las pruebas que se realizaron sobre los ejemplos, se clasificó bien a la persona y la región Inception, solo se presentó un caso donde el seguimiento se perdió.

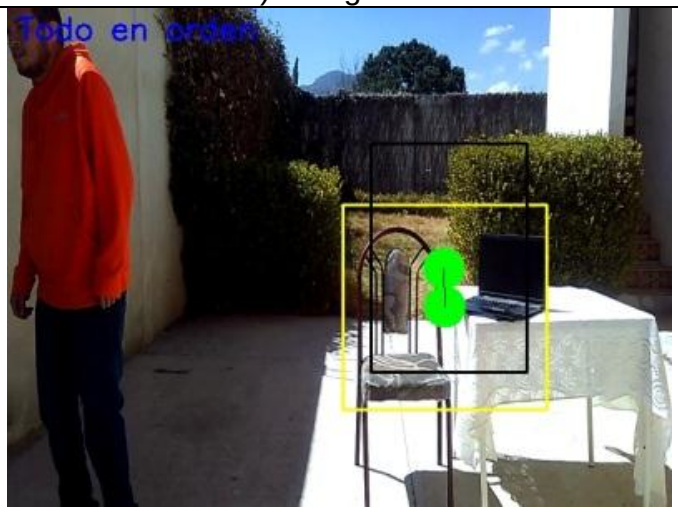
3.9.3 Pruebas en exterior

Las pruebas en exterior muestran algunos problemas, en algun caso el seguimiento se perdió como se muestra a continuación, además de que era mas fácil que la región de Inception se detectaran otras clases. En la Figura 46 podemos ver un ejemplo.



a) Imagen inicial

b) Imagen cerca



c) Imagen lejos

Figura 46. DTV aplicado a ejemplos en el exterior.

En la Figura 46 por las variaciones del clima e iluminación fue más fácil perder el seguimiento lo cual es incorrecto para la aplicación del algoritmo. A

continuación en la Figura 47 se muestran más ejemplos que si fueron seguidos y clasificados correctamente.

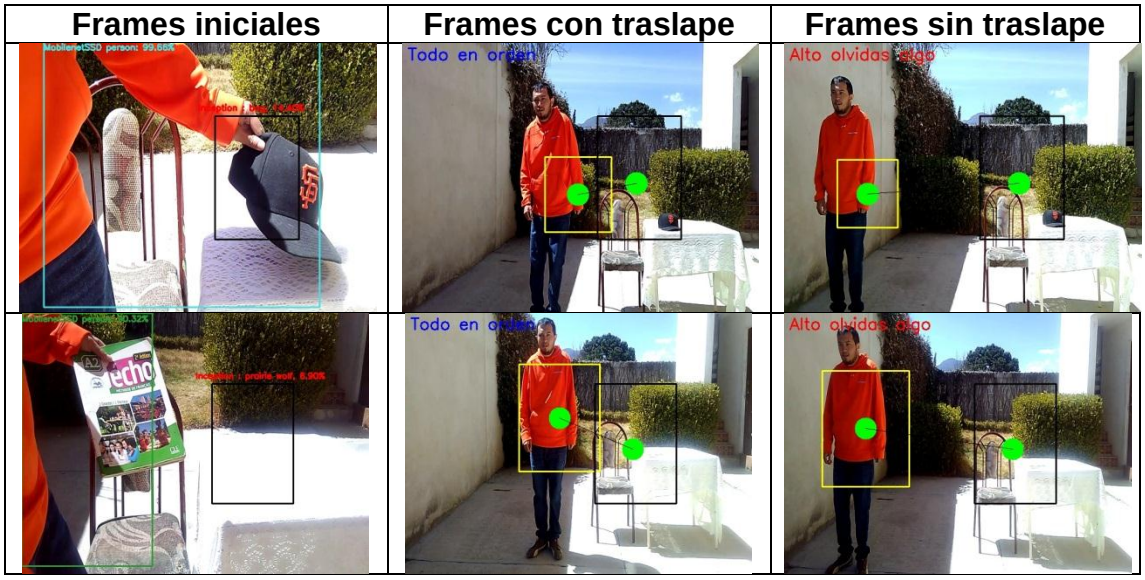


Figura 47. Ejemplos evaluados con DTV en exterior.

Finalmente las pruebas al espacio abierto permitieron ver que la clasificación de la persona es correcta sin embargo la región de Inception clasificaba objetos más diferentes por el movimiento del mantel donde se colocaban por el aire y rápidamente se desactivaba. Por lo tanto las variaciones de iluminación y el aire hacen que se clasifiquen objetos diferentes más rápido lo que es incorrecto.

3.9.4 Pruebas con 2 personas

Se propone realizar pruebas cuando hay más de un individuo y existan en el video 2 personas, a continuación se muestran algunos frames con resultados en la Figura 48.

a) Imagen inicial

b) Imagen cerca

c) Imagen lejos

Figura 48. DTV aplicado con 2 personas.

Como se observa en este ejemplo de la Figura 48 en los frames iniciales se clasifican las regiones de persona de una forma correcta sin embargo por el número de personas a ubicar se seleccionó un resultado aleatorio donde el resultado para el seguidor fue la segunda persona (la más alejada) como la principal a seguir lo cual es incorrecto. A continuación se muestran más ejemplos en la Figura 49 en donde se varía el número de persona lo que propicia que se seleccione la primera persona (la que está más cerca) de la región Inception.

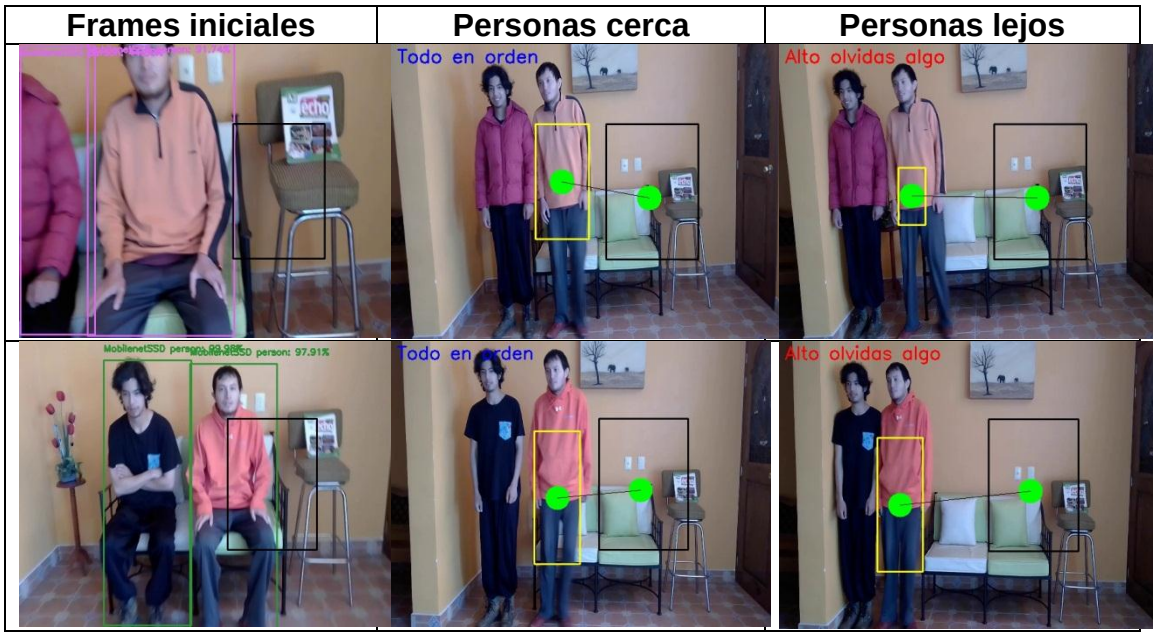


Figura 49. Ejemplos evaluados con DTV para 2 personas.

Esta prueba fue más difícil pues para que se detectaran la clasificación de persona se debe definir un número de parámetro límite de persona aleatorio que detecte a la persona más cercana y además debe ser un número de parámetro alto, es decir, el clasificador detectará a la persona principal que puede ser la primera o la segunda según el número que se definió en la ventana principal y además el parámetro debe ser alto para que la región que se definió para el seguidor sea la correcta para la región de la persona. Esta prueba es por lo tanto sensible al parámetro de límite de persona en valores altos aleatorios.

3.9.5 Pruebas con parámetros diferentes

El mayor cambio notorio en las pruebas es el cambio de distancia límite por lo tanto se propone cambiar este valor. En esta prueba se toma el valor de distancia límite en 400 en la ventana principal para visualizar como afecta el comportamiento en el algoritmo. A continuación en la Figura 50 se muestran algunos ejemplos.

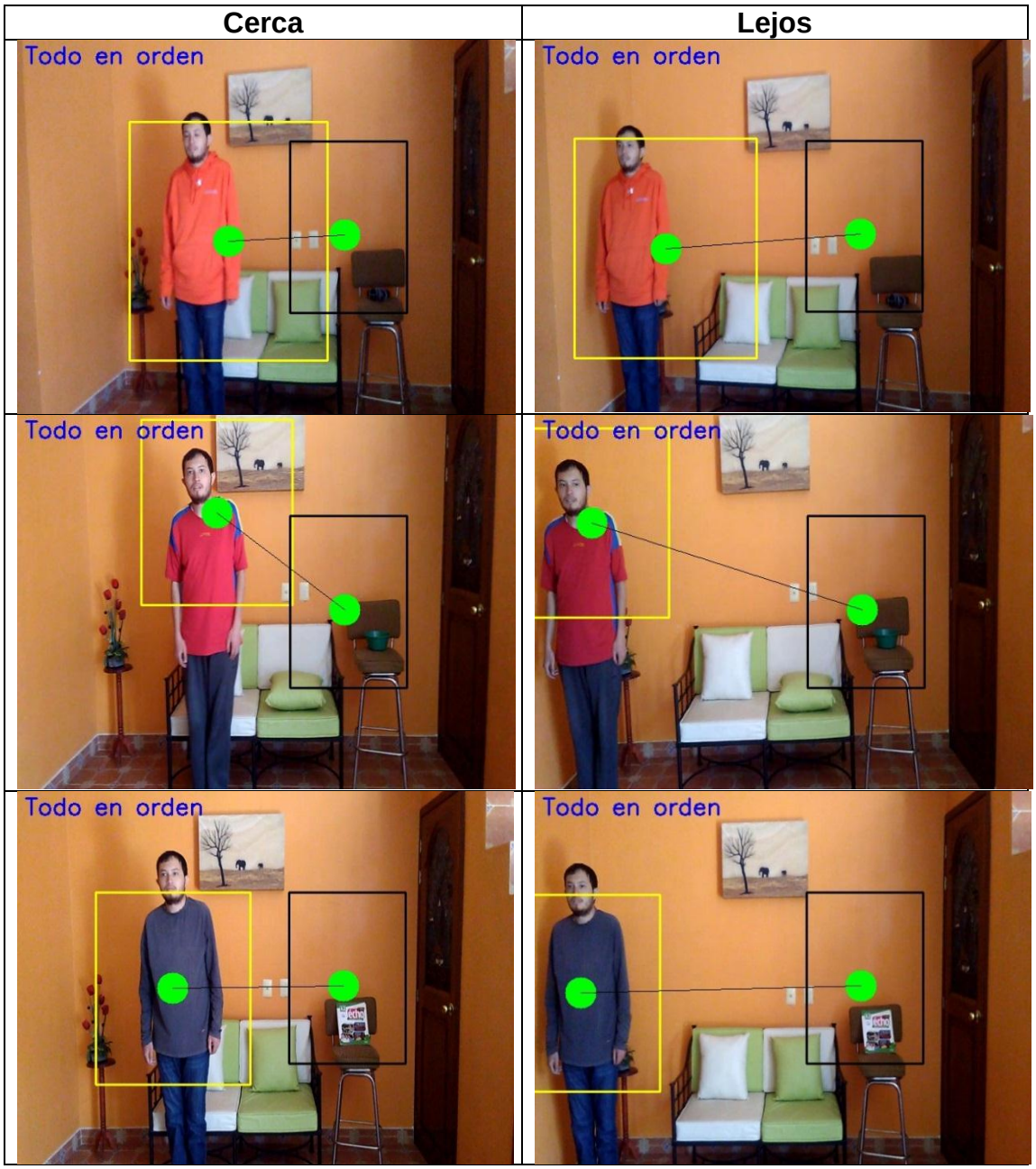


Figura 50. Ejemplos evaluados con un límite de distancia alto en DTV.

Como se observa en los resultados de la Figura 50 ya no se muestra el mensaje de olvido a distancias alejadas de la región de Inception y la región del seguidor solo se muestra el mensaje de “Todo en orden” para las distancias cerca y lejos lo cual demuestra que el valor de distancia limite puede cambiarse y afectar de manera negativa al proyecto pues no cumple la opción de mostrar el mensaje de olvido.

CONCLUSIONES

CAPÍTULO 4

En este trabajo de tesis se propuso un algoritmo que permite clasificar la relación de pertenencia entre dos objetos a través de deep learning y un algoritmo de seguimiento con información de alerta en una pantalla. Para la parte de deep learning se utilizaron modelos pre-entrenados de deep learning los cuales fueron los modelos Mobilenet SSD para ubicar la clase persona y el modelo Inception para clasificar clases de diversos objetos. Para la parte de seguimiento se decidió utilizar el algoritmo CRST. Todo esto en conjunto para generar la aplicación que se propone en esta tesis. Es decir, estas tecnologías actúan en conjunto primero en ubicar una persona y con esto inicializar el seguidor CRST para todos los frames nuevos además detectar que ha cambiado un objeto en una región definida denominada la región Inception y que finalmente se utilizan sus puntos centrales de CRST con la región Inception calculando la distancia euclidiana para mostrar el mensaje de alerta según se supere un límite definido.

Como trabajo futuro se propone que el algoritmo distinga en la secuencia de los frames a 2 personas y decidir a quién le pertenece el objeto que se propone sea el individuo más cercano.

ANEXO A

Bases de datos y benchmarks

A.1 Bases de datos para deep learning

En la Tabla 9 se describen las principales bases de datos para la investigación en deep learning.

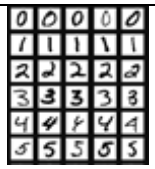
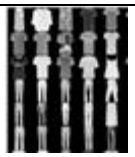
Dataset	Clases	Año de actividad	Ejemplo	Número de ejemplos
PASCAL Visual Clases Object	20	Inició 2005 - Finalizó 2012		11, 530
Imagenet	1000	Inició 2010 - Finalizó 2017		14, 197, 122
VQA Visual Question Answering	91	Inició 2015 - Continua		265, 016
Microsoft COCO: Common Objects in context	91	Inició 2015 - Continua		330, 000
MNIST	10	Inició 1998- Continua		70, 000
Cifar-10	10	Inició 1998- Continua		60, 000
FASHION MNIST	10	Inició 2017 - Continua		70, 000

Tabla 9. Información de bases de datos para aprendizaje profundo.

A.2 Benchmarks de seguimiento

En la Tabla 10 se presenta algunos benchmarks que investigan temas de seguimiento.

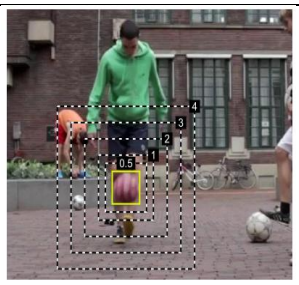
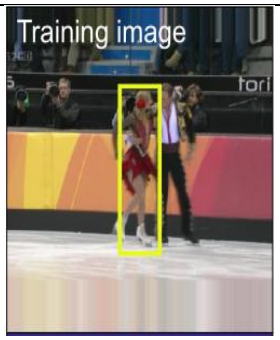
Benchmark	Descripción	Ejemplo
VOT Visual Object Tracking	Compara seguidores tanto como plataformas comunes para discutir la evaluación y avances hechos en el campo del seguimiento visual. Contiene 38 trackers y 60 datasets [58].	
Object tracking benchmark 100	Es una base de datos con posiciones de objetos y extensiones para seguimiento, además se introduce la secuencia de atributos para el análisis de ejecución. Integra 29 algoritmos de seguimiento y 100 secuencias en un framework [59].	
Multiple Object Tracking Benchmarck	Provee una base de datos con el objetivo de hacer tracking a múltiples personas caminando. Contiene 14 videos y un toolkit [9].	

Tabla 10. Información de benchmarks que investigan el seguimiento de objetos.

ANEXO B

Arquitecturas deep learning y CSR-DFC

B.1 Mobilenet

Está basada en una arquitectura simplificada que usa convoluciones separadas en profundidad para construir redes neuronales profundas pequeñas. Además utiliza 2 capas características, la primera es la convolución en profundidad que tiene la función de filtrar y la segunda capa es la convolución puntiaguda que sirve para combinar lo cual permite generar características. También este modelo presenta dos híper parámetros el primero α el cual se denomina multiplicador de ancho cuya función es adelgazar uniformemente cada capa, es decir, para un número de canales de entrada M y el número de canales de salida N se transforman en αM y αN. El segundo parámetro es el multiplicador de resolución ρ el cual se aplica a la imagen de entrada lo que subsecuentemente afecta a las representaciones internas para que sean reducidas.

Este modelo tiene un interés en la eficiencia y simplicidad pues se pensó para utilizarse en sistemas embebidos y dispositivos móviles. Es un trabajo desarrollado por empleados de Google presentado en 2017 [35].

La estructura general del modelo Mobilenet está construida en convoluciones separables en profundidad excepto por la primera capa la cual es una capa de convolución completa. Todas las capas están seguidas de una batchnorm y una función ReLU excepto por la capa final que es una capa de la función softmax para la clasificación. La Figura 51 presenta la arquitectura de Mobilenet.

Arquitectura de Mobilenet

Tipo/Stride	Forma del filtro	Tamaño de entrada
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

Figura 51. Arquitectura de Mobilenet.

B.1.1 Convolución separable en profundidad

Es la combinación de la convolución en profundidad y la convolución 1x1 (convolución puntiaguda). Donde la convolución en profundidad con un filtro por canal de entrada (profundidad de entrada) puede ser escrita como:

$$G_{k,l,m} = \sum_{i,j} K_{i,j,m} \cdot F_{k+i-1,l+j-1,m} \tag{38}$$

Donde K es el kernel convolucional en profundidad de tamaño $D_k \; x \; D_K \; x \; M$ donde el m_{esimo} filtro en K es aplicado a el m_{esimo} canal en F para producir el m_{esimo} canal de la salida filtrada del mapa de características G. Se debe de tomar en cuenta que D_k es la dimensión espacial del kernel asumido para ser cuadrado, m es el número de entradas. La Figura 52 muestra la diferencia entre la convolución normal y la convolución en profundidad.

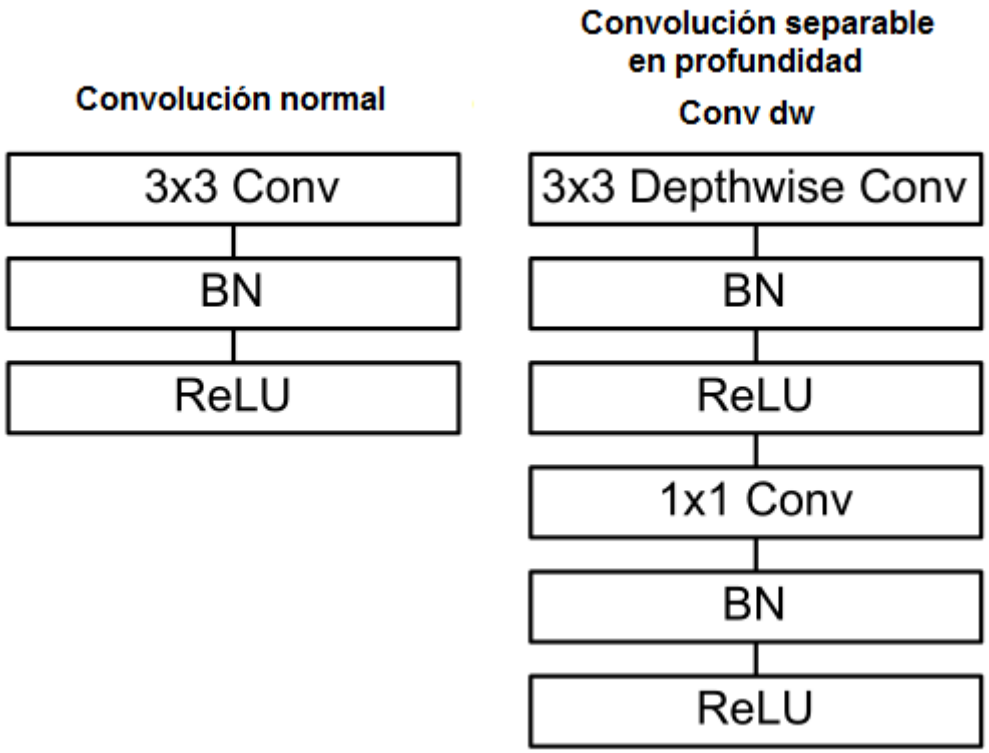


Figura 52. Diferencias entre convoluciones.

B.2 SSD

El enfoque Single Shot Multibox Detector (SSD) permite detectar objetos en imágenes usando redes neuronales profundas [37]. Esto se realiza discretizando el espacio de salida de los bounding boxes sobre diferentes escalas de localización en los mapas de características, es decir, se usan las salidas convolucionales de bounding boxes en múltiples escalas de los mapas de características en las últimas capas de la red lo que permite modelar el espacio de las posibles formas de las bounding boxes resultado. Por lo tanto en el entrenamiento se debe usar las regiones bounding box seleccionadas cuidadosamente de los objetos. La Figura 53 muestra la selección de la clase gato por default que se utilizara para el mapa de características.

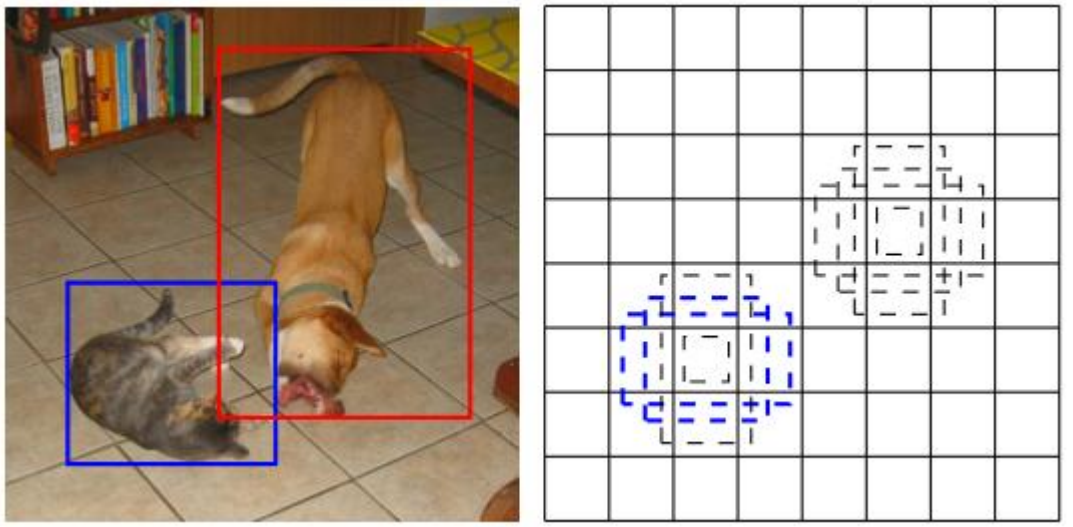


Figura 53. Caja delimitadora de gato default para el entrenamiento.

La arquitectura general está basada primero en una red base VGG16, utiliza el gradiente descendente simple, híper parámetros como batchsize, taza de aprendizaje, momentum, además se aplica la función de perdida y el algoritmo de backpropagation de extremo a extremo.

B.3 CSR-DCF o CRST

El nombre completo es Filtro de correlación discriminativo con canal y fidelidad espacial (por sus siglas en ingles). Es un algoritmo de seguimiento de objetos en imágenes con buenos resultados en tiempo real. En general el CSR-DFC puede verse como un mapa de fidelidad espacial que restringe el filtro de correlación a partes adecuadas para hacer seguimiento (tracking), mejorando la localización dentro de una región grande de búsqueda y además actúa para formas irregulares [27].

En general el mapa de fidelidad espacial adapta el filtro de soporte a la parte del objeto adecuado para hacer seguimiento, además este mismo resuelve los problemas del cambio circular adaptando una búsqueda arbitraria del tamaño de la región y también considera las limitaciones relacionadas a la suposición en forma de rectángulo.

Entonces los filtros de correlación en múltiples canales son bastante populares en los algoritmos de tracking. A continuación se definen. Dado un conjunto N_c y características de canales $f = \{f_d\}\, d = 1{:}\, N_c$ y ventanas de objetivos correspondientes (filtros) $h = \{h_d\}\, d = 1{:}\, N_c$, la posición del objeto es estimada como la ubicación de la respuesta de máxima correlación $\hat{g}(h)$,

$$\hat{g} = \sum_{d=1}^{N_c} f_d \star h_d \tag{39}$$

El símbolo $\star$ representa la correlacion circular entre $f_d \in R^{C_w \, x \, C_h}$ y $h_d \in R^{C_w \, x \, C_h}$, donde C_w y C_h es la región de entrenamiento/búsqueda expresada en ancho y alto. Ahora el filtro óptimo de correlación h es estimado por minimización:

$$\varepsilon(h) = \sum_{d=1}^{N_c} \|\hat{g}(h) - g\|^2 + \lambda \|h\|^2 \qquad (40)$$

Donde g es la salida esperada, además $g \in R^{C_w \, x \, C_h}$ que es típicamente una función gaussiana en 2D centrada en la ubicación del objetivo. Debe de considerarse que el trabajo CSR-DFC mejora puntos de los filtros de correlación espaciales que a continuación se definen.

El método consiste en optimizaciones, la primera optimización se refleja en la función de costo pues se considera cada canal independiente, es decir, cada filtro de canal es optimizado para rellenar las salidas separadamente. La función de costo es:

$$\varepsilon(h) = \sum_{d=1}^{N_c} \|f_d \star h_d - g\|^2 + \lambda \|h_d\|^2 \qquad (41)$$

Adicionalmente se agrega canales de pesos w = $\{\widehat{w}_d\}_{d=1:N_c}$ los cuales se considera factores de escalamiento basados en el poder discriminativo de cada canal de características. Esos pesos son llamados pesos de canales de confianza, además estos mismos son aplicados cuando la respuesta de correlación es calculada en el paso de localización del objeto:

$$\hat{g} = \sum_{d=1}^{N_c} f_d \star h_d \cdot \widehat{w}_d \qquad (42)$$

Los canales de fidelidad $\widehat{w}_d$ reflejan la importancia de cada canal en la etapa de localización, existen 2 tipos de canales en este método. El primero es el canal de fidelidad de aprendizaje $\widehat{w}_d^{(lrn)}$ el cual se calcula en la etapa del aprendizaje del filtro, el segundo es el canal de fidelidad de detección $\widehat{w}_d^{(det)}$ el cual es calculado en la etapa de localización. Cabe destacar que la fidelidad es estimada por la solución de mínimos cuadrados. Además los scores de los canales de fidelidad son usados para pesar las respuestas de filtros por canales en la localización. La fiabilidad de canales unidos $\widehat{w}_d$ en la etapa de localización del objeto se calcula como el producto de ambos canales:

$$\widehat{w}_d = \widehat{w}_d^{(lrn)} \cdot \widehat{w}_d^{(det)} \qquad (43)$$

Y normalizada $\sum_d \widehat{w}_d = 1$.

La siguiente optimización es respecto al canal con el que se trata, pongamos m $\in \{0,1\}$ sea un mapa espacial de fidelidad con elementos ya sea 0 ó 1, que identifica pixeles, los cuales deberían ser puestos en cero en el filtro aprendido. La restricción es definida por $h = m \cdot h$ (producto de Hadamard o elemento a elemento). Sin embargo, esta restricción no tiene una solución cerrada así que los autores de este algoritmo propusieron una solución iterativa respecto al trabajo de Kiani Galoogahi H. (2015) que permite derivar eficientemente el problema dando como resultado el uso de la transformada de Fourier. La optimización depende en usar una inversa de la transformada y otra transformada de Fourier lo que hace el algoritmo óptimo. También otro paso que se propone es utilizar una probabilidad a posteriori para hallar m.

B.4 Inception

La arquitectura Inception fue propuesto por investigadores de Google en el 2014 para el challenge ILSRVC2014 es una estructura que prioriza la profundidad y el ancho de la red, además mantiene los recursos computacionales constantes [38].

El modulo original de la arquitectura Inception está basada en encontrar como una estructura óptima a través de una red de visión convolucional puede ser aproximada y cubierta por componentes densos fácilmente disponibles. Es decir, la red utiliza redes convolucionales de tamaños 3x3, 5x5 y 1x1. Los cuales se combinan en una salida extra que es un filtro receptor (filter bank). A este cluster se le denomina módulo de Inception, la Figura 54 lo representa.

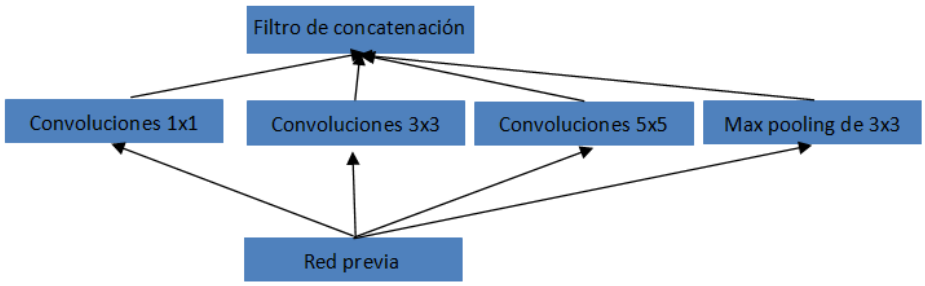


Figura 54. Módulo Inception.

Sin embargo el modulo original de Inception tiende a incrementar las salidas en las primeras fases, por tal motivo en la arquitectura Inception se propone aplicar reducción de dimensión y proyección donde los requerimientos computacionales podrían incrementar demasiado. A esto se le denomina Inception con reducción de dimensión. Esta estructura se logra gracias a convoluciones 1x1 que son usadas para calcular reducciones antes de calcular convoluciones de 3x3 o 5x5. A continuación se presenta un módulo Inception con reducción de dimensión en la Figura 55:

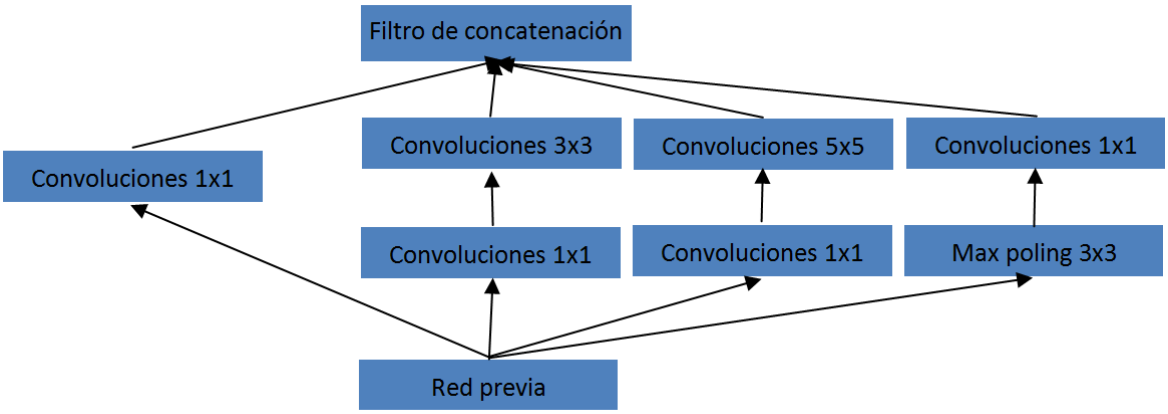


Figura 55. Módulo Inception con reducción de dimensión.

La arquitectura general está formada de módulos Inception con reducción de dimensión además presenta los siguientes elementos.

- Una capa pooling de promedio con filtros 5x5 y stride 3.
- Una convolución 1x1 con 128 filtros para reducción de dimensión y activación Relu.
- Una capa completamente conectada con 1024 unidades y Relu.
- Una capa dropout con 70% de salidas borradas.
- Una capa lineal con perdida softmax para clasificar.

BIBLIOGRAFÍA

[1] LeCun, Y., Bengio, Y., & Hinton, G.(2015).Deep learning. nature, 521(7553), p. 436-444.

[2] Abhinav Dadhich.(2018). Practical Computer Vision Extract insightful from images using Tensorflow, Keras and Opencv. Packt.

[3] Sebastian Raschka & Vahid Mirjalili.(2017). Python Machine Learning Machine Learning and Deep Learning with Python, scikit-learn, and Tensorflow. Packt.

[4] https://github.com/opencv/opencv/wiki/Deep-Learning-in-OpenCV. Sitio web de modelos pre-entrenados en Opencv. Recuperado el 15 de Enero del 2021.

[5] Bharath Ramsundar & Reza Bosagh Zadeh.(2018). Tensorflow for Deep Learning from linear regressionb to reinforcement learning. O'Reilly.

[6] http://pyimagesearch.com/. Blog donde se tratan proyectos de Machine Learning y Deep Learning. Recuperado el 15 de Enero del 2021.

[7] https://www.tensorflow.org/. Sitio web oficial de Tensorflow. Recuperado 2 de Febrero del 2021.

[8] Wojke, N., Bewley, A., & Paulus, D. (2017, September). Simple online and realtime tracking with a deep association metric. In 2017 IEEE international conference on image processing (ICIP). p. 3645-3649.

[9] https://motchallenge.net/. Sitio oficial del challenge MOT. Recuperado el 1 de Marzo del 2021.

[10] https://www.youtube.com/watch?v=DTajdV7S2qo. Video que explica la función de pérdida. Recuperado el 20 de Julio del 2021.

[11] Moolayil J.(2019). Deep Neural Networks A fast track aproach to modern deep learning with python. Apress.

[12] https://www.tensorflow.org/api_docs/python/tf/keras/metrics. Sitio que enlista métricas de Tensorflow. Recuperado el 24 de Julio del 2021.

[13] Glorot, X., & Bengio, Y. (2010, March). Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the thirteenth international conference on artificial intelligence and statistics. p. 249-256.

[14] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems. p. 1097-1105.

[15] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

[16] Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition. p. 580-587.

[17] https://towardsdatascience.com/understanding-and-reducing-bias-in-machine-learning-6565e23900ac. Sitio web que explica el bias en problemas de discriminación. Recuperado el 31 de Julio del 2021.

[18] https://www.diegocalvo.es/perceptron/. Página acerca del perceptrón. Recuperado el 31 de Julio del 2021.

[19] Sitio que explica la función de activación, es decir, una función de activación retorna un valor dependiendo de las entradas, además de ser muy útil en deep learning.https://www.diegocalvo.es/funcion-de-activacion-redes-neuronales/#:~:text=Definici%C3%B3n%20de%20funci%C3%B3n%20de%20activaci%C3%B3n,o%20(%2D1%2C1. Recuperado el 5 de Agosto del 2021.

[20] https://es.wikipedia.org/wiki/Curva_ROC. Página acerca de la curva ROC. Recuperado el 11 de Agosto del 2021.

[21] https://towardsdatascience.com/types-of-convolution-kernels-simplified-f040cb307c37. Página que aborda los tipos de kernels en la convolución. Recuperado el 6 de Agosto del 2021.

[22] https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks/. Página que explica convolución en deep learning. Recuperado el 7 de Agosto del 2021.

[23] https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks-Part-2/. Página que explica el Stride de deep learning. Recuperado el 10 de Agosto del 2021.

[24] https://opencv.org/. Página oficial de la librería de Opencv. Recuperado el 14 de Agosto del 2021.

[25] http://www.isoin.es/introduccion-en-tecnicas-de-tracking-o-seguimiento/. Página acerca del seguimiento de objetos en imágenes. Recuperado el 14 de Agosto del 2021.

[26] https://rpubs.com/desareca/Reconocimiento-Imagenes-Correlacion. Página sobre la correlación de imágenes. Recuperado el 14 de Agosto del 2021.

[27] Alan, L., Vojíř, T., Čehovin, L., Matas, J., & Kristan, M. (2018). Discriminative correlation filter tracker with channel and spatial reliability. International Journal of Computer Vision, 126(7), p. 671-688.

[28] Ning, G., Zhang, Z., Huang, C., Ren, X., Wang, H., Cai, C., & He, Z. (2017, May). Spatially supervised recurrent convolutional neural networks for visual object tracking. In 2017 IEEE international symposium on circuits and systems (ISCAS). p. 1-4.

[29] Zhou, X., Xie, L., Zhang, P., & Zhang, Y. (2014, October). An ensemble of deep neural networks for object tracking. In 2014 IEEE International Conference on Image Processing (ICIP). p. 843-847.

[30] Yoon, K., Kim, D. Y., Yoon, Y. C., & Jeon, M. (2019). Data association for multi-object tracking via deep neural networks. Sensors, 19(3), p. 559.

[31] Fuerstenberg, K. C., Dietmayer, K. C., & Willhoeft, V. (2002, June). Pedestrian recognition in urban traffic using a vehicle based multilayer laserscanner. In Intelligent Vehicle Symposium, 2002. IEEE (Vol. 1). p. 31-35.

[32] V. K. Singh, B. Wu and R. Nevatia, "Pedestrian Tracking by Associating Tracklets using Detection Residuals," 2008 IEEE Workshop on Motion and video Computing, Copper Mountain, CO, 2008, p. 1-8, doi: 10.1109/WMVC.2008.4544058.

[33] ZuWhan Kim, "Real time object tracking based on dynamic feature grouping with background subtraction," 2008 IEEE Conference on Computer Vision and Pattern Recognition, Anchorage, AK, 2008, p. 1-8, doi: 10.1109/CVPR.2008.4587551.

[34] https://es.wikipedia.org/wiki/Sistemas_inteligentes_de_transporte. Página acerca de los sistemas de transportes inteligentes. Recuperado el 29 de Agosto del 2021.

[35] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T. & Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.

[36] Duda R. et al. (2003). Pattern classification. WILEY-INTERCSIENCE.

[37] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016, October). Ssd: Single shot multibox detector. In European conference on computer vision. p. 21-37. Springer, Cham.

[38] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition. p. 1-9.

[39] https://github.com/chuanqi305/MobileNet-SSD. Página que contiene el modelo pre-entrenado de Mobilenet SSD. Recuperado el 4 de Agosto del 2021.

[40] https://docs.opencv.org/master/d5/de7/tutorial_dnn_googlenet.html. Página que contiene el modelo pre-entrenado de Inception. Recuperado el 4 de Octubre del 2021.

[41] http://image-net.org/challenges/LSVRC/2012/index. Esta página muestra la información del Challenge ILSVRC2012. Recuperado el 2 de Febrero del 2021.

[42] https://keras.io/. Página del framework Keras. Recuperado el 7 de Junio del 2021.

[43] https://caffe.berkeleyvision.org/. Página del framework Caffe. Recuperado el 7 de Junio del 2021.

[44] https://pypi.org/project/Theano/. Página de python del framework Theano. Recuperado el 7 de Junio del 2021.

[45] https://pytorch.org/. Página del framework de aprendizaje automático Pythorch. Recuperado el 7 de Junio del 2021.

[46] https://numpy.org/. Página oficial de numpy. Recuperado el 7 de Junio del 2021.

[47] Página oficial para este proyecto de tesis que se aloja en google drive pues permite alojar videos e información.drive.google.com/drive/folders/1jW2S-0FnAX_q5gmLe2rLpLgodLReuXs1?usp=sharing. Recuperado el 5 de Junio del 2021.

[48] https://www.researchgate.net/post/How_can_I_calculate_the_accuracy. Página que describe la fórmula de precisión o accuracy. Recuperado el 1 de Julio del 2021.

[49] https://es.wikipedia.org/wiki/Error_absoluto_medio. Página que describe la fórmula del error absoluto medio. Recuperado el 1 de Julio del 2021.

[50] https://es.wikipedia.org/wiki/Error_cuadr%C3%A1tico_medio. Página que describe la fórmula del error cuadrático medio. Recuperado el 1 de Julio del 2021.

[51] https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/loss-functions/categorical-crossentropy. Página que describe la fórmula de categorical crosentropy. Recuperado el 1 de Julio del 2021.

[52] https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html. Página que describe la fórmula de binary crosentropy. Recuperado el 1 de Julio del 2021.

[53] Challa S. et al. (2011). Fundamentals of object tracking. Cambridge University Press.

[54] Zhu H. et al. (2017). Overview of environment perception for intelligent vehicles. IEEE. Transactions on Intelligent Transportation Systems, 18(10), p. 2584-2601.

[55] https://www.youtube.com/watch?v=OXZ63W99tPo. Video que explica el experimento del carro autónomo Navya. Recuperado el 7 de Julio del 2021.

[56] https://www.youtube.com/watch?v=gSP56BiyPPc. Video que explica como es y cómo funciona un autobús autónomo chino. Recuperado el 7 de Julio del 2021.

[57] developers.google.com/machine-learning/crash-course/classification/roc-and-auc?hl=es-419. Recuperado el 4 de Agosto del 2021.

[58] https://www.votchallenge.net/. Página oficial del enfoque VOT. Recuperado el 7 de Julio del 2021.

[59] http://cvlab.hanyang.ac.kr/tracker_benchmark/datasets.html. Página del enfoque OTB. Recuperado el 7 de Julio del 2012.

[60] Sherstinsky, A. (2020). Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. Physica D: Nonlinear Phenomena, 404, 132306.

[61] Wang, E. K., Zhang, X., & Pan, L. (2019). Automatic classification of CAD ECG signals with SDAE and bidirectional long short-term network. IEEE Access, 7, 182873-182880.