UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

PLATAFORMA DE VISUALIZACIÓN DE INFRAESTRUCTURA ELÉCTRICA URBANA MEDIANTE INTELIGENCIA ARTIFICIAL

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELÉCTRICA

RODRIGO ALFREDO PÉREZ BARROS

PROFESOR GUÍA:
FERNANDO CASTILLO ARCE

MIEMBROS DE LA COMISIÓN:
FRANCISCO RIVERA SERRANO
ANDRÉS LARA CÓRDOVA

Este trabajo ha sido parcialmente financiado por:
FORCAST

SANTIAGO DE CHILE
2021

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL ELÉCTRICA
POR: **RODRIGO ALFREDO PÉREZ BARROS**
FECHA: 2021
PROF. GUÍA: FERNANDO CASTILLO ARCE

PLATAFORMA DE VISUALIZACIÓN DE INFRAESTRUCTURA ELÉCTRICA URBANA MEDIANTE INTELIGENCIA ARTIFICIAL

La infraestructura de la red eléctrica de distribución, dada su extensión y complejidad, es propensa a fallas generando interrupciones en el suministro eléctrico a sus clientes. Procesos de mantención preventiva se hacen necesarios de aplicar. Por otro lado, junto al plan estatal de llevar a la ciudad de Santiago a ser una *SmartCity*, se tiene una gran oportunidad de solucionar la problemática presente mediante tecnologías emergentes. Se propone la creación de un Sistema de Información Geográfica (SIG), donde el objetivo inicial, y que valida la solución, es el desarrollo de un modelo computacional que detecte eventos anómalos en la red eléctrica de distribución de forma automática. Mediante técnicas de procesamiento de imágenes y *deep learning*, se desarrolla una serie de módulos que conforman el modelo computacional. Los resultados, en términos de métricas, que entrega en detectar eventos de postes y catenarias son: Precision de 0.9286 y 0.954, Recall de 0.9785 y 0.8857, y F1 de 0.9529 y 0.9186, respectivamente. Se logra entonces desarrollar un modelo computacional que toma en cuenta las características de la infraestructura eléctrica chilena y permite obtener alertas preventivas de mantenciones, lo que genera a su vez un avance para el plan estatal de llevar a la capital a ser una *SmartCity*.

A mis amigos y cercanos,
que estuvieron en los momentos difíciles.

Gracias

TABLA DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE ILUSTRACIONES

Capítulo 1

Introducción

1.1. Motivación y Antecedentes del problema

La infraestructura de la red eléctrica de distribución, dada su extensión y complejidad, es propensa a fallas generando interrupciones en el suministro eléctrico a sus clientes. Se realizan regularmente inspecciones y mantenimientos en la red, actividad costosa debido a la logística y recursos humanos. En el año 2016 la empresa de distribución eléctrica Enel realizó más de 200.000 inspecciones [1], las cuales, sin embargo, no impidieron los cortes del 2017 que significaron en altas multas para la empresa. Los habitantes de las ciudades se ven afectados directamente con los cortes, teniendo consecuencias que pueden llegar a ser fatales. Debido a la falta de energía en los extensos cortes de 2017 en Santiago, dos electrodependientes fallecieron [2]. Desde fuentes internas de Enel distribución han indicado que solo pueden hacer monitoreo del 45 % de la infraestructura, quedando un 55 % sin visualizar. Además, los cortes afectan fuertemente a la imágen de la empresa y también implican considerables sanciones económicas. Enel fue multado durante los años 2017 y 2018 por un total de CLP 15.033 millones por las interrupciones en sus servicios [3], [4], las que podrıan haberse reducido considerablemente con un método de mantenimiento predictivo que hubiese alertado sectores frágiles de la red, permitiendo la aplicación de mejores planes de contingencia. A través del SERNAC, los reclamos en el año 2017 aumentaron más del 300 % que los que se realizaron durante el año 2016 [3].

El mantenimiento que se realiza a la red eléctrica sigue siendo reactivo ante los eventos y fallas, lo cual no es acorde al plan de la Agenda Digital 2020 que desea llevar a la ciudad de Santiago a ser una *SmartCity* [5], ya que la detección previa a los eventos que provocan fallas son claves para prevenirlo. El uso de la tecnología, junto a los avances en las aŕeas de la automatización de procesos y de la inteligencia artificial, son una alternativa real para lograr un mantenimiento del suministro eléctrico eficiente y adecuado para los objetivos de desarrollo de la ciudad.

El construir un *software* de visualización de la infraestructura eléctrica urbana representa una gran oportunidad para identificar mediante una herramienta potente y eficaz aquella infraestructura propensa a fallas. Esto significa una oportunidad comercial para el usuario de la herramienta, ya que permitiría anteponerse a problemas, reduciendo costos de mantención, reacción y multas por cortes. Además, también resulta ser un desafío atractivo para el país dado que involucraría desarrollo de *software* nacional en conjunto con un trabajo multidisci-

plinario entre distintas áreas profesionales; en particular, la de dos ingenierías: computacional, dedicada al desarrollo del visor de la infraestructura y el manejo correcto del flujo de datos de entrada y salida que ingresará o solicitará el cliente respectivamente; y la ingeniería eléctrica, que a partir de su diversidad de conocimiento en distintas materias (que es, por lo menos, la misión de enseñanza que tiene el departamento de Ingeniería Eléctrica de la Universidad de Chile [6]), enfocará su trabajo en la automatización de diversos procesos con el fin de tener un modelo predictor de fallas, además de planificar las distintas etapas y/o cuidados que deberá tener el producto desarrollado, gracias a sus conocimientos en el contexto en el que será aplicado: las redes eléctricas de distribución.

1.2. Solución

La solución que se plantea es una plataforma de Sistema de Información Geográfica (SIG) en la nube que analiza diferentes fuentes de videos e imágenes para alertar, advertir y geolocalizar los eventos riesgosos y defectuosos en redes de distribución eléctricas urbanas de baja y media tensión mediante procesos tecnológicos inteligentes. Esta información será usada por empresas eléctricas distribuidoras para mejorar el servicio a sus clientes y para prevenir pérdidas en sus activos. Los servicios que entregará este proyecto son los siguientes:

- Localización geográfica en Plataforma y Mapa de los eventos identificados
- *Fast-Track* de respuesta para problemas capturados por redes sociales (RRSS) - Dolencia de Enel Distribución
- Reportes automáticos
- Seguimiento histórico de problemas identificados
- Aprendizaje y generación de información en Mapa Geográfico de Hot-Spots y tendencias de eventos. El desarrollo tecnológico consiste en:
    - Algoritmos de reconocimiento e integración de imágenes
    - Algoritmos de pre-procesamiento y procesamiento de imágenes
    - Algoritmo de detección de objetos
    - Algoritmo experto de alerta ante reconocimiento
    - Reconocimiento acelerado ante información e imágenes de RRSS
    - Algoritmos de referencia y localización de eventos
    - Integración y desarrollo de una plataforma Sistema de Información Geográfica de integración de alertas ante eventualidades en la infraestructura eléctrica urbana
    - Análisis de tendencias en la plataforma

1.3. Objetivos

El principal desafio del proyecto, el que también será parte del **objetivo general** que se desarrollará en este informe bajo el contexto del trabajo de título, consiste en **desarrollar**

un modelo computacional que detecte eventos anómalos en la red eléctrica de distribución de forma automática. El proceso de identificación de elementos mediante técnicas de inteligencia artificial será desarrollado por FORCAST -entidad colaboradora del proyecto- que posee capacidades robustas en términos de telecomunicaciones, preprocesamiento y procesamiento de imágenes, manejo de datos y diseño de prototipos.

Para llevar a cabo el objetivo general del presente trabajo, se desglosan los siguientes **objetivos específicos**:

1. Detectar postes.
2. Detectar catenarias.
3. Clasificar catenarias a peligrosas o no.
4. Calcular altura de una catenaria.
5. Geolocalizar eventos.
6. Detectar postes defectuosos,

donde la hipótesis es que el rendimiento de los módulos que aplican *machine learning* presentan valores superiores a 0.9 en sus métricas de evaluación; a saber: *precision*, *recall* y F1.

1.4. Estructura del informe

El informe a continuación está organizado de la siguiente manera: en la siguiente seccion se detalla el **marco teórico** en el cual se basa el trabajo, dividiéndolo en dos partes: el procesamiento digital de imágenes, respondiendo a las preguntas sobre en qué consiste, qué conceptos frecuentes son necesarios conocer y qué ejemplos existen sobre este procesamiento; y el estado del arte, en el cual se revisan y analizan trabajos previos que serán útiles para el desarrollo de este trabajo. La sección que le sigue muestra la **metodología** a seguir, junto a los pasos previos al desarrollo del modelo computacional. Luego se muestra el **trabajo realizado** como partida para el presente proyecto, junto a la evaluación de los resultados con métricas utilizadas de forma estándar en el ámbito de la visión computacional, con el fin de obtener una evaluación objetiva. Finalmente, se presentarán las **conclusiones**, con una visión futura de trabajo en este proyecto de Plataforma de Visualización de Infraestructura Eléctrica Urbana Mediante Inteligencia Artificial.

Capítulo 2

Marco teórico

2.1. Procesamiento digital de Imágenes

Las imágenes digitales suelen ser ocupadas como una fuente de información para los modelos computacionales. Sin embargo, no siempre se logran los resultados esperados al ocupar toda la información original que se extraen de las imágenes. El procesamiento digital de imágenes busca solucionar el problema anterior, a través de diversos procesos que se aplican a una imagen de entrada, obteniendo una imagen de salida con atributos e información mejores para el propósito que se requiere [7].

Antes de detallar algunas técnicas de procesamiento digital de imágenes se hace necesario explicar cómo son las imágenes capturadas por una cámara digital. Teniendo de ejemplo una imagen de tamaño 1600×1200 a color regular, significa que el computador lee la imagen como un arreglo numérico de 1600 píxeles de ancho, 1200 píxeles de alto y a cada par numérico le corresponde 3 canales que representan cada uno de los colores primarios: rojo, verde y azul. Esta representación es conocida como RGB (por sus siglas en inglés), y la representación matricial de los canales puede ser vista en la Figura 2.1. El el caso de que la imagen capturada estuviese en escala de grises se tendría 1 solo canal y las mismas dimensiones de pixeles. Los valores de cada píxel de un arreglo tienen un valor entero entre 0 y 255 (también puede ser un valor real entre 0 y 1). El valor de cada píxel representa su intensidad lumínica dentro del canal, por lo tanto, en la escala de grises o escala de colores el píxel negro tiene valor (0) ó (0, 0, 0) mientras que el píxel blanco tiene el valor (255) o (255, 255, 255), respectivamente.

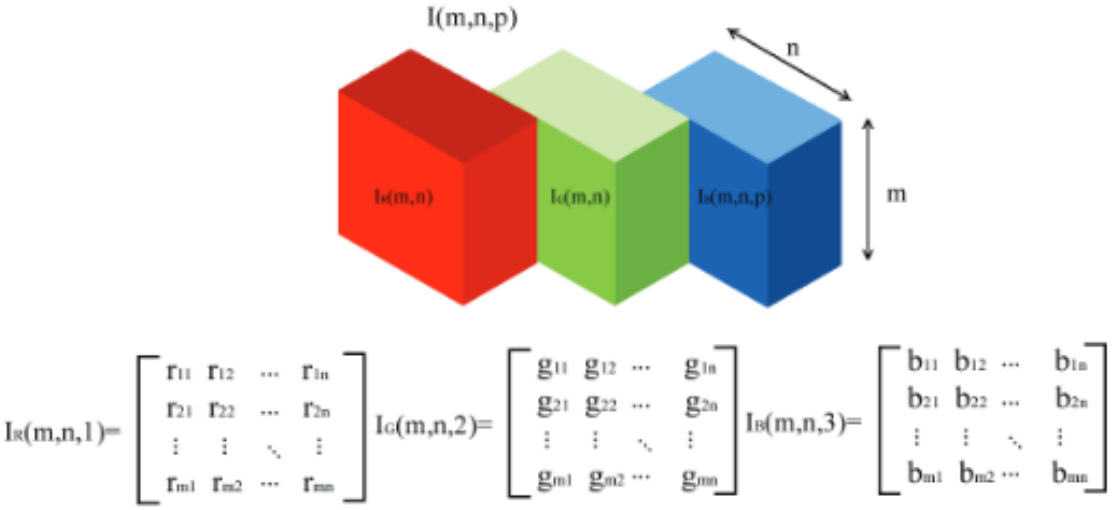


Figura 2.1: Planos de color RGB representados como tres matrices bidimensionales [7].

En el escenario real, el ojo humano puede percibir un objeto de un determinado color, el cual al ser capturado por una cámara se esperaría que su valor en pixeles sea acorde a lo observado. Por ejemplo, si se observa un objeto pintado rojo de manera uniforme, se esperaría que su valor en pixeles sea (255, 0, 0). Sin embargo, los valores realmente observados en la imagen tienen un nivel de error asociado que puede ser considerable dependiendo del ruido ocasionado por el funcionamiento de la cámara y/o las condiciones ambientales al momento de capturar la fotografía. Esto último dificulta el análisis computacional de imágenes; por ello es que se hace necesario realizar un pre-procesamiento de imágenes para prepararlas a la tarea que se quiera desarrollar.

2.1.1. Conceptos

Antes de estudiar algunas técnicas de procesamiento se deben conocer algunos conceptos frecuentes en las descripciones de éstas:

- **Filtro:** es una matriz (arreglo de 2 dimensiones) que por lo general se utiliza para una operación de convolución. Sin embargo, a veces se utiliza el término filtro o máscara como el resultado de la operación y el término *kernel* a la matriz. Dependiendo del filtro se tienen distintos resultados en la operación. Los operadores que los utilizan se dividen en: global, local y puntual, dependiendo de si el tamaño del filtro es igual al de la imagen, menor que la imagen o igual a un punto, respectivamente. Un ejemplo de filtro y operación se muestra en la Figura 2.2. Por lo general, los filtros poseen dimensiones de números impares, lo que permite tener 1 solo valor en su centro. Y dado que varias operaciones y comparaciones de filtros consideran el valor del centro, se facilitan tales procesamientos cuando éste es único.

- **Convolución:** es una operación que involucra un filtro con una imagen I, generando una nueva imagen. A cada punto de la imagen se le coloca el filtro con el elemento del centro en el punto. Luego se suma la multiplicación de los elementos que calcen

y el resultado es el valor del píxel de la nueva imagen. En la Figura 2.2 se muestra un ejemplo de convolución. La Ecuación (2.1) corresponde a la función de convolución aplicada a una imagen I con un filtro *kernel* K de tamaño $(2k+1) \times (2k+1)$, esto para cada píxel (x, y) de la imagen, donde $w_{i,j}$ son los elementos que tiene el *kernel* K.

$$(I * K)_{x,y} = \sum_{i=-k}^{+k} \sum_{j=-k}^{+k} w_{i,j} \cdot I(x+i, y+j) \tag{2.1}$$

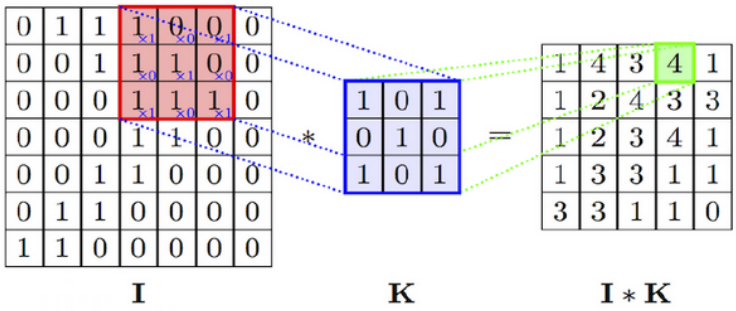


Figura 2.2: Ejemplo de una operación de convolución, donde I es la imagen, K es el filtro e I*K convolución [8].

Hay que destacar que en ciertos casos el filtro no opera sobre todos los píxeles de la imagen, como ,por ejemplo, cuando el centro del filtro pasa por los bordes de la imagen es imposible aplicar la operación de convolución. Existen diversas técnicas para tratar con estos casos límites y evitar pérdidas de información, sin embargo, no son relevantes para los tamaños de las imágenes que se utilizan en este trabajo.

- **Máscara:** se utiliza el término máscara para el filtro global, de tamaño igual al de la imagen, que por lo general disminuye la región de búsqueda. Por ejemplo, un filtro por color azul sobre una imagen genera una nueva imagen del mismo tamaño que contiene solamente los píxeles de color azul, disminuyendo la región de búsqueda. Esto queda claro si se observa la Figura 2.31 de más adelante. Existen también las **sub-máscaras**, que son de menor tamaño que la imagen y, por lo cual, se unen para generar una máscara completa.

A continuación se detallan algunas técnicas de procesamiento que se utilizan en el trabajo.

2.1.2. Ejemplos de procesamientos

2.1.2.1. Conversión a escala de grises

Como se explicó, la diferencia entre una imagen a color y su versión en escala de grises es que cada píxel posee 3 componentes (1 por cada canal de color) en lugar de solo 1 valor. Esto se traduce en que el costo computacional de operar imágenes RGB sea más alto que con imágenes en escala de grises, por lo que es común realizar la conversión de RGB a escala de grises previo a la aplicación de ciertas operaciones que no operan en función de la gama de colores. Hay distintos tipos de transformación a escala de grises, dentro de los cuales están

las Ecuaciones (2.2), (2.3), donde $I_{gris}(x, y)$ es el valor del píxel (x, y) en la imagen en escala de grises y $p_{azul}(x, y)$, $p_{rojo}(x, y)$, $p_{verde}(x, y)$ son los valores del píxel (x, y) en la imagen a color.

[9]:

$$I_{gris}(x, y) = \frac{1}{3}(p_{azul}(x, y) + p_{rojo}(x, y) + p_{verde}(x, y)) \tag{2.2}$$

[10], [11]:

$$I_{gris}(x, y) = (0.114 * p_{azul}(x, y) + 0.299 * p_{rojo}(x, y) + 0.587 * p_{verde}(x, y)) \tag{2.3}$$

Si bien todas las transformaciones anteriores generan una imagen en escala de grises, éstas producen contrastes diferentes que pueden ser útiles para determinadas aplicaciones. Un ejemplo de escala de grises se muestra en la Figura 2.3.

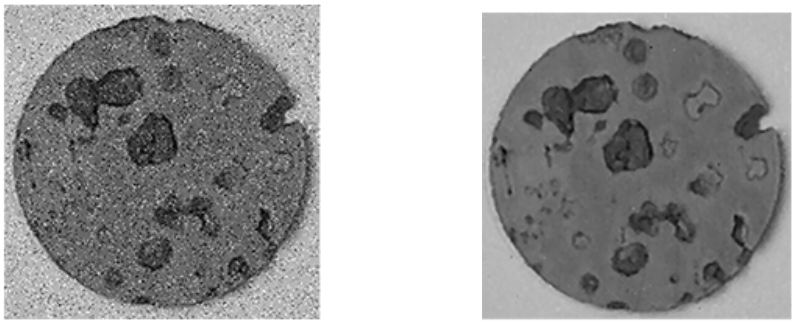
Figura 2.3: Ejemplo de una conversión a escala de grises [12].

2.1.2.2. Suavizado

El suavizado de la imagen busca, entre otras cosas, disminuir el ruido existente en ella. Se basa en que, en la práctica, los objetos ocupan varios píxeles en la imagen, donde se tiene que en la vecindad de cada uno de esos píxeles la distribución de color es similar, por lo que se asume que cualquier píxel con valores que no se correspondan con sus vecinos equivalen a ruido en la imagen. Por lo tanto, las operaciones de suavizado son, en su mayoría, filtros pasa-bajo que eliminan las altas frecuencias del ruido suavizado. El suavizado es una operación local y existen distintos tipos:

- **Filtro de media:** también llamado Filtro de caja normalizado. Se calcula el promedio en un entorno local para cada uno de los píxeles; esto es equivalente a aplicar una operación de convolución con filtros del tipo que se muestra en la Figura 2.4, donde la media se calcula en una cierta área. Esto reduce los cambios bruscos en los valores de los píxeles adyacentes. La Ecuación (2.4) corresponde a la fórmula del operador de un filtro de media de $(2k + 1) \times (2k + 1)$ a una imagen I [13].

$$media(I)(x, y) = \frac{1}{(2k + 1)^2} \cdot \sum_{i=-k}^{k} \sum_{j=-k}^{k} I(x + i, y + j) \tag{2.4}$$



a Imagen original b Imagen filtrada

Figura 2.4: Ejemplo de filtro de media [14].

- **Filtro de mediana:** se aplica una operación local parecida a la anterior que, en lugar de calcular la media, calcula la mediana de todos los píxeles en torno a su vecindad. Esto permite eliminar los puntos que tienen valores muy lejanos y conserva los valores comunes en la imagen, disminuyendo el contraste [13]. La fórmula del operador de un filtro de mediana de $(2k+1)\times(2k+1)$ a una imagen I esta representada por la Ecuación (2.5), mientras que un ejemplo de esta operación se visualiza en la Figura 2.5.

$$\text{median}(I)(x,y) = \text{median}\{p(i,j) \mid x-k<i<x+k; y-k<j<y+k\} \qquad (2.5)$$

Input					
1	4	0	1	3	1
2	2	4	2	2	3
1	0	1	0	1	0
1	2	1	0	2	2
2	5	3	1	2	5
1	1	4	2	3	0

Output					
1	4	0	1	3	1
2	1	1	1	1	3
1	1	1	1	2	0
1	1	1	1	1	2
2	2	2	2	2	5
1	1	4	2	3	0

Figura 2.5: Ejemplo de una operación con filtro de mediana [15].

- **Filtro gaussiano:** el filtro gaussiano busca lo mismo que el filtro de media. Sin embargo, pondera con distinto peso los píxeles adyacentes que el del centro, de tal forma que permita filtrar las frecuencias altas sin modificar tanto el contraste. Un ejemplo de una operación de convolución es aplicar el filtro de la Figura 2.6.

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

/273

Figura 2.6: Ejemplo de un filtro gaussiano [13].

- **Filtro bilateral:** si bien los filtros anteriores suavizan la imagen, también generan una pérdida de contraste cuando el filtro contiene el borde que separa 2 objetos, lo cual no permite detectar bien los bordes en procesos posteriores. Debido a esto aparece el filtro bilateral; sigue la misma idea del filtro gaussiano, donde los píxeles adyacentes tienen un peso distinto al del centro. Sin embargo, los pesos también dependen de la diferencia de intensidad entre el píxel del centro y los píxeles adyacentes. Con esto se logra eliminar el ruido y conservar los bordes de la imagen. Un ejemplo de aplicación de este filtro se observa en la Figura 2.7.

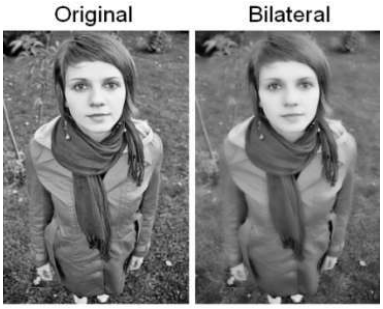


Figura 2.7: Ejemplo de un filtro bilateral [16]

2.1.2.3. Filtro por color

En ciertas ocasiones se quiere buscar en una imagen un objeto del cual solo se conocen sus colores. En estos casos se pueden buscar todos los píxeles de la imagen cuyos valores estén dentro del rango en torno a los colores correspondiente. Con esto se puede reducir el área de búsqueda del objeto, disminuyendo el costo computacional y las posibles detecciones falsas en un algoritmo de detección de objetos. De la misma forma, se puede eliminar de la imagen los píxeles cuyos valores estén fuera del rango del color deseado. En la Figura 2.8 se extrajo de la imagen original los píxeles correspondientes a, ya sea, al rango del color naranja o blanco.

a Imagen original

b Imagen filtrada

Figura 2.8: Ejemplo de un filtro por color, extrayendo los píxeles que están dentro del rango del naranja y blanco en la escala RGB [17].

2.1.2.4. Umbralización de Imágenes

La umbralización (*thresholding* en inglés) es una transformación aplicada a una imagen, usualmente de un valor por píxel como lo son las imágenes de grises, donde se define un umbral η para efectuar una discriminación de valores de píxeles [18]. Si se considera que el mínimo valor de un píxel en una imagen es u_{min} y el máximo es u_{max}, entonces η se debe

encontrar entre aquellos dos valores. Dependiendo del acabado que se requiera en la imagen, existen distintos tipos de umbralización:

- **Binarización:** los píxeles de la imagen pueden tener solamente dos valores; los píxeles de valor menor a η adquieren el valor u_{min} y los mayores a η toman el valor u_{max}. Generalmente $u_{min} = 0$ y $u_{max} = 255$, resultando la umbralización en una imagen en blanco y negro.
- **Operación de truncar:** los píxeles con valor mayor a η se mapean al valor del umbral y el resto conserva su valor.
- **Umbral a cero:** los píxeles con valor mayor al umbral quedan sin cambios y el resto se mapea a 0.

Estas umbralizaciones corresponden a operaciones de umbral fijo, lo que quiere decir que se utiliza el mismo valor de η para discriminar a todos los píxeles de la imagen. Cuando se tienen imágenes donde las condiciones de iluminación no son uniformes, el umbral fijo suele dejar fuera valores importantes, lo que significa una pérdida de información. Para solucionar este problema se requiere contar con **umbrales adaptativos** que, como el nombre sugiere, varían el valor de η de acuerdo a la región de la imagen que se está analizando, adaptándose a la generalidad del objeto que rodean. Algunos métodos de umbralización adaptativa son:

- **Umbral adaptativo con media:** el umbral η es igual a la media de los valores de los piíxeles que se encuentran en un área local.
- **Umbral adaptativo gaussiano:** el umbral se obtiene con una convolución entre un filtro gaussiano y el área local.

En la Figura 2.9 se observa la diferencia entre utilizar un umbral fijo y un umbral adaptativo en una imagen con iluminación no homogénea.

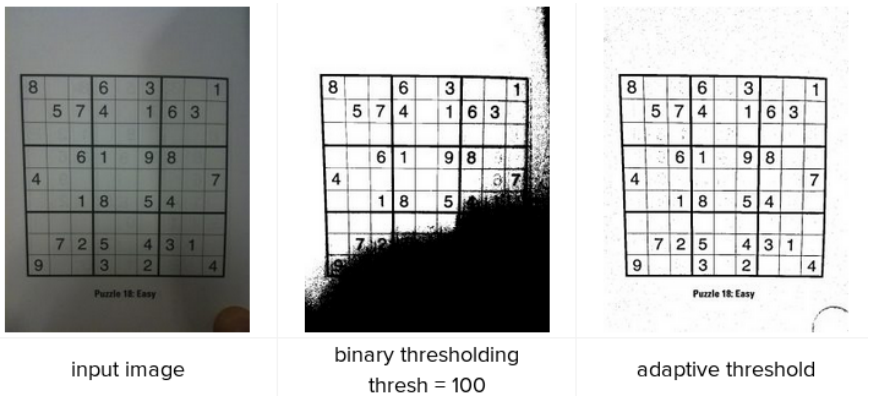


Figura 2.9: Diferencia de umbralización fija y adaptativa [19].

2.1.2.5. Mejora de contraste

Un método para realzar los detalles en una imagen es el ajuste de contraste. Esto permite observar con mayor detalle los elementos presentes y diferenciarlos del resto. Existen diversos ajustes al contraste que se pueden realizar, dependiendo del objetivo:

- **Transformación lineal al histograma [13]:** esta operación consiste en aplicar una transformación lineal sobre los píxeles de una imagen para que abarquen el rango completo de valores disponibles. A modo de ejemplo, si se tiene una imagen cuyo valor mínimo es $u_{min} = 30$ y su valor máximo es $u_{max} = 170$, se genera una transformación lineal de manera que el valor 30 sea mapeado a 0 (G_{min}) y 170 a 255 (G_{max}). De esta manera, los valores de los píxeles en la imagen utilizan todo el rango disponible. El efecto de esto es un realce de los colores, mejorando la vista de contornos y características de los objetos presentes. En resumen, la transformación de una imagen I en una nueva imagen de contraste ajustado I_{ca}, se obtiene modificando el valor de cada píxel u de acuerdo con la función $g(u)$, como se indica en la Ecuación (2.6a):

$$g(u) = b(u + a) \tag{2.6a}$$

$$a = -u_{min} \tag{2.6b}$$

$$b = \frac{G_{max} - G_{min}}{u_{max} - u_{min}} \tag{2.6c}$$

Los valores G_{min} y G_{max} pueden ser distintos de 0 y 255, respectivamente, dependiendo de los requerimientos del procesamiento de imágenes. En la Figura 2.10 se muestra un ejemplo de mejora de contraste con esta técnica.

a Imagen original

b Imagen con mejora de contraste mediante transformación lineal

Figura 2.10: Comparación de imagen con mejora de contraste por escalamiento lineal del histograma [original de los autores, 2020].

- **Ecualización del histograma [13]:** es una transformación que tiene como objetivo que todos los valores de píxeles aparezcan con la misma frecuencia en la imagen. Esto no es posible en general, debido a que valores iguales de píxeles en la imagen original I deben quedar en un mismo valor en la imagen nueva I_{new}. Por lo que la solución es una aproximación a esta problemática. A cada valor de píxel u de la imagen I se le asigna un valor $g(u)$, tal como se muestra en la Ecuación (2.7), donde c_I es la frecuencia relativa acumulada. Un ejemplo visual de esta transformación se encuentra en la Figura 2.11.

$$g(u) = c_I(u) \cdot G_{max} \tag{2.7}$$

Figura 2.11: Izquierda: imagen original. Derecha: imagen luego de la ecualización del histograma [13].

2.1.2.6. Detección de borde

La detección de borde se utiliza para segmentar objetos. Dentro de los distintos tipos de detector de borde se encuentra el siguiente:

- **Operador Canny [13]:** este método calcula las derivadas horizontal S_x y vertical S_y de la imagen, con el fin de obtener el módulo del gradiente a través de la Ecuación (2.8) y el ángulo ϕ del gradiente de la forma ($\phi = \arctan(S_y, S_x)$), el cual se redondea en múltiplos de $\pi/4$. Luego, se compara el valor del módulo del gradiente de cada píxel con el de sus vecinos en la dirección de ϕ. Si el píxel satisface que el módulo de su gradiente es el mayor en la dirección de ϕ y, además, es mayor que un umbral *High threshold*, entonces se marca como borde. Finalmente, se realiza un paso de seguimiento de borde, donde se parte siguiendo el trazo de los bordes ya marcados y si se encuentra un píxel adyacente que es máximo en su dirección y mayor a un umbral *Low threshold* < *High threshold*, entonces se marca como borde. Esto queda ejemplificado en la Figura 2.12.

$$|S_x| + |S_y| \approx ||gradiente||_1 \tag{2.8}$$

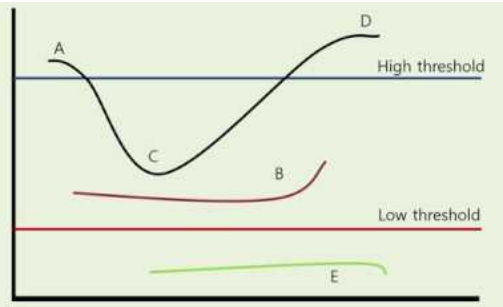


Figura 2.12: Los puntos A, D que están arriba de *High threshold* son marcados como borde. Los puntos B, E que están abajo de *High threshold* que no tienen conexión con ni un punto de A o D son descartados. Y los puntos C son considerados como borde por tener una conexión con A o D [20].

- **Algoritmo FAST [21]:** como los bordes en imágenes digitales se definen por tener píxeles con intensidades particulares a su alrededor [13], en donde hay alta variación de intensidad en torno a su vecindad, este algoritmo funciona de la siguiente manera:

  - Para cada píxel de la imagen se compara su intensidad I_p con la intensidad de los 16 vecinos que hay en un círculo de radio 3 píxeles que tienen intensidad I_i.
  - Se elige un umbral τ para comparar la intensidad de los píxeles. Se revisa qué tan brillante es el píxel central con respecto a sus vecinos, donde se tiene que si el píxel es notoriamente más brillante que sus vecinos, se cumple que $I_p < I_i - \tau$, mientras que si es notoriamente menos brillante, se satisface $I_p > I_i + \tau$.
  - Luego, si hay 12 píxeles contiguos en el círculo de vecinos que satisfacen que o son mucho más brillantes que el central o mucho más oscuros, entonces se considera que el punto central es una esquina.

  La Figura 2.13 muestra un ejemplo para el algoritmo FAST, aplicada en una área de la imagen que se presenta.

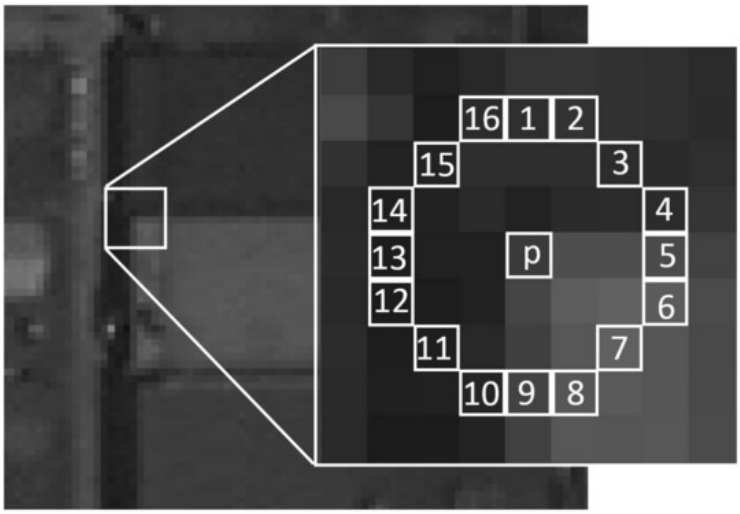


Figura 2.13: Ejemplo de comparación para el algoritmo FAST. [21].

2.1.2.7. Detección de líneas

En ocasiones se necesita obtener ciertos segmentos de una imagen que están delimitadas por líneas. Para ello hay diversas técnicas que permiten detectar las líneas, destacando entre ellas la Transformada de Hough [22].

- **Transformada de Hough [22]** Ésta es una técnica que localiza figuras en imágenes y es usada preferentemente para extraer líneas, círculos y elipses. En particular, para encontrar líneas en una imagen, lo primero es recordar que dada una parametrización cartesiana, los puntos colineales de una imagen con coordenadas "x" e "y" están relacionados por su pendiente "m" y el punto de intercepción "n" (con el eje "y") según la Ecuación (2.9).

$$y = mx + n \tag{2.9}$$

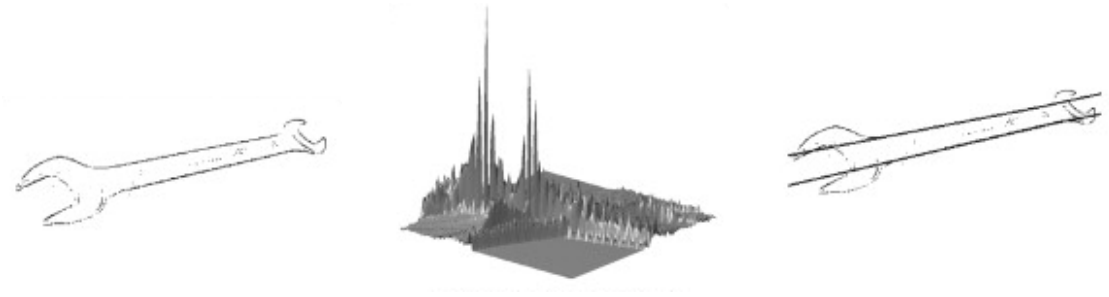
Esta ecuación también puede ser escrita en su forma homogénea como la Ecuación (2.10):

$$Ay + Bx + 1 = 0 \tag{2.10}$$

donde $A = \frac{-1}{n}$ y $B = \frac{m}{n}$. Esto quiere decir que una línea está definida por el par de valores "A" y "B". Entonces, la Transformada de Hough toma como referencia el punto (A,B) considerando que todos los puntos (x,y) definen la misma línea en el espacio (A,B). Esto da paso a la Ecuación (2.11).

$$Ay_i + Bx_i + 1 = 0 \tag{2.11}$$

El algoritmo de la Transformada de Hough encuentra esos puntos de manera eficiente, considerando que todos los elementos colineales de una imagen definen líneas duales con el mismo punto concurrente (A,B), y cuenta las soluciones potenciales en una matriz acumuladora. El conteo se hace trazando todas las líneas duales para cada punto (xi,yi). Cada punto en el trazado incrementa un elemento en la matriz, por lo que el problema de la extracción de líneas se transforma en el problema de localizar un máximo en el espacio del acumulador. Esta estrategia es robusta y ha demostrado ser capaz de manejar el ruido y la oclusión. Un ejemplo de aplicación se puede apreciar en la Figura 2.14, junto a su matriz acumuladora.



a Imagen original b Matriz acumuladora c Líneas detectadas

Figura 2.14: Ejemplo de aplicación de la Transformada de Hough [22]

2.2. Aprendizaje Profundo (*deep learning*)

La inteligencia artificial es un subcampo de la informática que busca que las computadoras sean capaces de solucionar tareas particulares de forma autónoma. Para lograr a cabo este proceso, existe un subcampo dentro de la inteligencia artificial a cargo de ello; éste corresponde al aprendizaje de máquina (*machine learning*), el cual consiste en un conjunto de algoritmos que derivan conocimiento a partir de datos de ejemplo, con el propósito de aprender un modelo que permita realizar predicciones sobre datos nuevos. Existen 3 maneras de aplicar el aprendizaje de máquina; éstos son:

- **Aprendizaje supervisado:** técnica que necesita datos de ejemplos etiquetados para generar un modelo que predice sobre datos nuevos de acuerdo a lo aprendido. Por ejemplo, se le entrega muchas imágenes de perros etiquetadas "perro" y de gatos etiquetadas

"gato" como referencia. Luego de eso el modelo generado aprende a reconocer perros y gatos, sin necesidad que sean los mismos de referencia.

- **Aprendizaje no supervisado:** procedimiento que genera información significativa de un conjunto de datos, sin haberle entregado un resultado conocido, como los ejemplos etiquetados del punto anterior. Por ejemplo, se le entregan imágenes de perros y de gatos sin decir a cuál corresponde cada una. El algoritmo extrae diferencias y puede agruparlas en perros y gatos sin saber qué son cada uno.
- **Aprendizaje por refuerzo:** consiste en desarrollar un sistema (agente) que mejore su rendimiento basado en interacciones con el entorno [23], donde cada vez que el sistema realiza algo de manera correcta se le otorga un refuerzo positivo para que el modelo busque querer hacer eso nuevamente.

Ahora bien, existen diversas técnicas para llevar a cabo el proceso de aprendizaje en *machine learning*. Uno de ellos corresponde al aprendizaje profundo (*deep learning*), cuya finalidad es modelar abstracciones de alto nivel en datos, usando una red neuronal artificial compuesta por un número de arquitecturas jerarquizadas [24]. Como se introdujo el concepto de red neuronal artificial, que es importante dentro de la inteligencia artificial, se hace necesario explicarlo en mayor profundidad.

2.2.1. Redes Neuronales Artificiales

La actividad mental del ser humano consiste principalmente en redes de células cerebrales llamadas neuronas. Inspiradas en esa hipótesis es que los primeros trabajos de inteligencia artificial apuntaron a crear redes neuronales artificiales. Con la finalidad de diseñar una red se necesita obtener un modelo matemático de una neurona, el cual se observa en la Figura 2.15.

Una red neuronal, en forma simplificada, es una colección de neuronas artificiales conectadas; las propiedades de la red son determinadas por su topología y las propiedades de las neuronas.

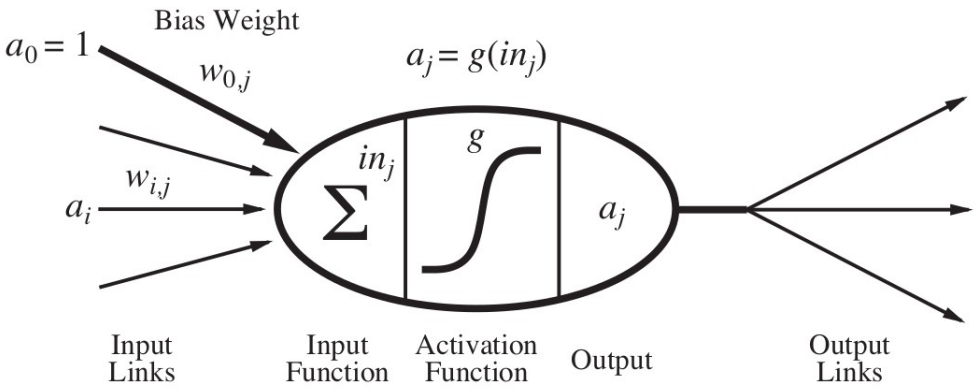


Figura 2.15: Modelo matemático de una neurona [25].

Las redes neuronales están compuestas por unidades conectadas entre sí llamadas nodos. Cada conexión entre nodos está dirigida con el objetivo de propagar la **activación** a_i desde i hasta j, la cual a su vez tiene asociado un peso numérico w_{ij} (equivalente a la sinapsis), que determina la fuerza de la señal de conexión. Por otro lado existe una operación de sumatoria que adiciona todas las señales de entrada (equivalente a la membrana de la célula que acumula carga eléctrica) y una función de activación que decide qué neurona disparar.

Luego de decidir el modelo matemático para la neurona individual, la siguiente tarea es conectarlas entre sí para formar una red. Principalmente, existen dos formas para hacer esto: una red *feed-forward* que tiene conexiones en una dirección y una red recurrente que retroalimenta el resultado de una neurona sobre sí misma. Las redes *feed-forward* son usualmente conformadas por nodos agrupados por niveles, llamadas en la literatura como capas (o *layer*), donde cada unidad recibe como entrada la salida de una neurona de una capa anterior [25].

Antes de profundizar en arquitecturas de redes neuronales multicapa más complejas, se describirá brevemente una arquitectura simple de estas mismas; la ***Multilayer Perceptron (MLP)***, de la cual su estructura se visualiza en la Figura 2.16.

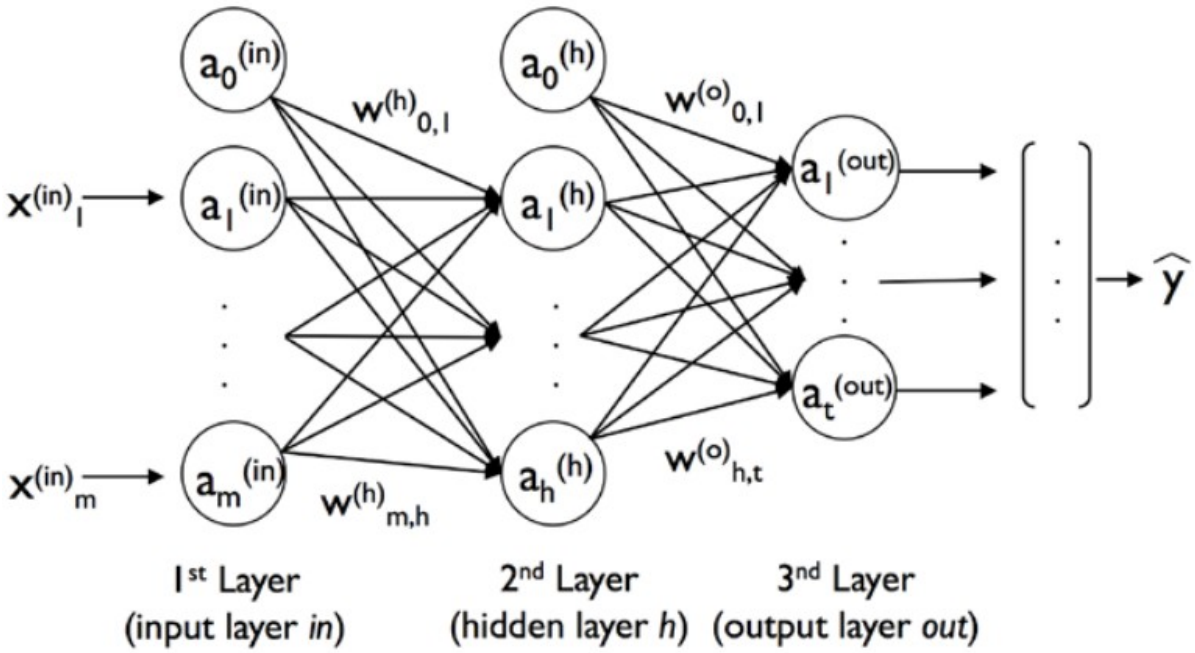


Figura 2.16: Estructura de una red *Multilayer Perceptron (MLP)* [23].

En la Figura 2.16 se observa que esta red se compone de una capa de entrada, una capa oculta y una capa de salida. Las unidades en la capa oculta están completamente conectadas a la capa de entrada y la capa de salida está completamente conectada a la capa oculta. Si tal red tiene más de una capa oculta, se dirá que es una red neuronal artificial profunda [23].

2.2.2. Aprendizaje Profundo para Visión Computacional

Uno de los principales desafíos actuales y que ha generado múltiples investigaciones y aplicaciones en el área de visión computacional, es la detección de objetos en imágenes. La detección de objetos consiste en la clasificación y localización del (o los) objeto(s) de interés en una imagen mediante un algoritmo.

Gracias al surgimiento de las redes neuronales profundas y, en particular, a su arquitectura más representativa llamada red neuronal convolucional (CNN por sus siglas en inglés) es que se han conseguido grandes avances en el área. Una red neuronal convolucional típica es **VGG16**, la cual se puede observar en la Figura 2.17. Una CNN extrae mapas de características de una imagen de entrada, donde cada elemento viene de una porción local de píxeles, el cual es conocido como campo receptivo local. Su nombre viene a que el campo receptivo local corresponde a un filtro, al cual se le aplica una operación de convolución a la imagen de entrada con ese filtro, dando como resultado el mapa de características. Las CNN tienen dos principales ventajas que se observan en las tareas relacionadas con imágenes, donde éstas son:

- **Conectividad-dispersa:** un único elemento en el mapa de característica sólo se conecta a una pequeña porción de píxeles. El número de conexiones es igual al tamaño del filtro para la operación de convolución.
- **Parámetros compartidos:** los mismos pesos (o parámetros) son usados en diferentes porciones de la imagen de entrada [23]. Esto quiere decir que se utiliza el mismo filtro para la convolución sobre toda la imagen.

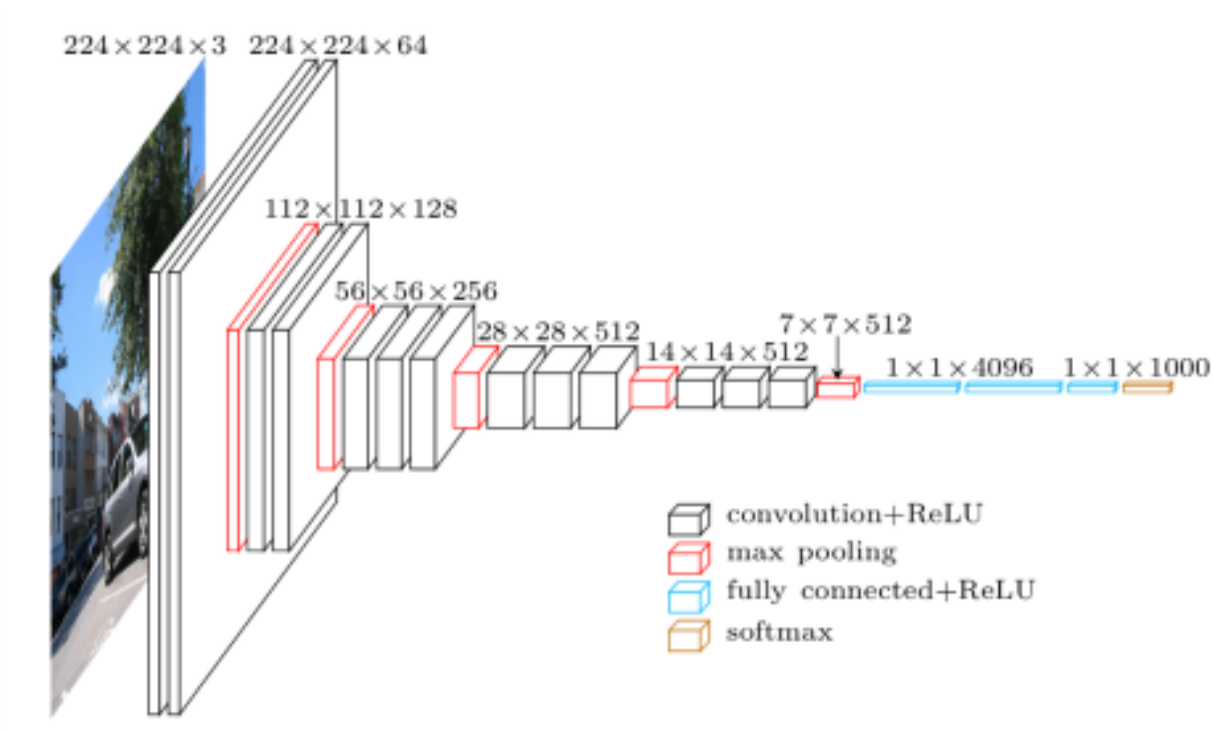


Figura 2.17: Estructura de una red neuronal convolucional [26].

Como una consecuencia directa de aquellos puntos, el número de pesos decrece drásticamente y se aprecia una mejora importante en la habilidad para capturar características destacadas. Intuitivamente, tiene sentido que los píxeles más cercanos entre sí sean más relevantes que los píxeles que están alejados entre sí.

Como se observa en la Figura 2.17, las CNN están compuestas de varias capas convolucionales y de reducción (***pooling***), que son seguidas por una o más capas completamente conectadas como término. Estas últimas son esencialmente perceptrones multicapa (ver Figura 2.16), donde cada unidad de entrada i es conectada a cada una de las unidades de salida j con pesos w_{ij} [23].

Las funciones que realiza cada tipo de capa se describen brevemente a continuación:

- **Capa de entrada:** esta capa recibe los datos en bruto de la imagen y los entrega a la siguiente capa sin cambio alguno.
- **Capa convolucional:** esta capa realiza la operación de convolución a la imagen que recibe y entrega como resultado un mapa de características. Este último es de igual tamaño que la imagen que recibe.
- **Capa de unidad lineal rectificada:** esta capa aplica una función de activación al resultado de la capa previa. Esta función es usualmente del tipo $max(0, x)$, la cual es necesaria para añadir no linealidad a la red para que pueda generalizar con éxito cualquier tipo de función.
- **Capa de reducción:** esta capa realiza un submuestreo al resultado de la capa previa, resultando en una estructura con dimensiones más pequeñas. La reducción ayuda a mantener sólo las características más prominentes a medida que se progresa en el procesamiento de la red. Hay distintos tipos de reducción; frecuentemente se usa la reducción máxima (***max pooling***) donde se elige el máximo valor en una ventana $k \times k$ dada que usualmente es de 2×2.
- **Capa Completamente Conectada:** ésta se utiliza generalmente como última capa donde calcula la puntuación de las salidas del nivel previo. Se denomina así porque, al contrario de una capa con conectividad-dispersa, conecta todos los puntos de la capa de entrada con cada neurona. La salida resultante es de tamaño $1 \times 1 \times L$, donde L es el número de clases en la colección de datos de entrenamiento [27].

2.2.2.1. Principales Arquitecturas para la Detección de Objetos

Antes de mencionar las arquitecturas para la detección de objetos, se deben mencionar qué son los métodos de clasificación, debido a que muchas arquitecturas de detección lo utilizan como punto final. Son técnicas que aprenden por medio de ejemplos a clasificar datos. Uno de los más utilizados es *Support Vector Machine* (SVM), el cual busca el hiperplano que mejor separa a los vectores de los datos de ejemplo. Luego, para datos nuevos basta ver a qué lado del hiperplano quedan y se clasifican como esa clase.

Los métodos de detección de objetos pueden ser categorizado en dos tipos: el primero sigue el flujo tradicional de detección de objetos, generando regiones propuestas en primera instancia, para luego clasificar cada región propuesta en su categoría respectiva; por el contrario, el segundo tipo considera la detección de objetos como un problema de regresión o clasificación, adoptando un *framework* unificado para conseguir resultados finales directamente (categorías y localizaciones). Los métodos basados en regiones propuestas incluyen: **R-CNN**, **SPP-net**, **Fast R-CNN**, **Faster R-CNN**, **R-FCN**, **FPN**, **Mask R-CNN**, algunos de los cuales están relacionados, ya que por ejemplo **SPP-net** modifica **R-CNN** con una capa **SPP**. Por otro lado, los métodos basados en regresión/clasificación incluyen: **MultiBox**, **AttentionNet**, **G-CNN**, **YOLO**, **SSD**, **YOLOv2**, **DSSD** y **DSOD** [28].

1. ***Framework* basado en Regiones Propuestas**

   a) R-CNN: es de significativa importancia mejorar la calidad de los cuadros de detección candidatos y tomar una arquitectura profunda para extraer características de alto nivel. Con la finalidad de subsanar ese problema, las **R-CNN**, en primera instancia, realizan una búsqueda selectiva para generar alrededor de 2000 regiones propuestas por cada imagen; luego cada región propuesta es deformada o recortada a una resolución fija, para que la **CNN** extraiga 4096 características como la representación final y, con un modelo del tipo **SVM (*Support Vector Machine*)** lineal pre entrenado para múltiples clases, se califican las diferentes regiones propuestas a un colección de regiones positivas y regiones de fondo. En última instancia las regiones calificadas son ajustadas con una regresión de cuadro de detección y filtradas con la técnica de supresión no-máxima para producir cuadros de detección que conservan la locación de los objetos [28]. EL diagrama de flujo del procesamiento de una **R-CNN** se puede observar en la Figura 2.18.

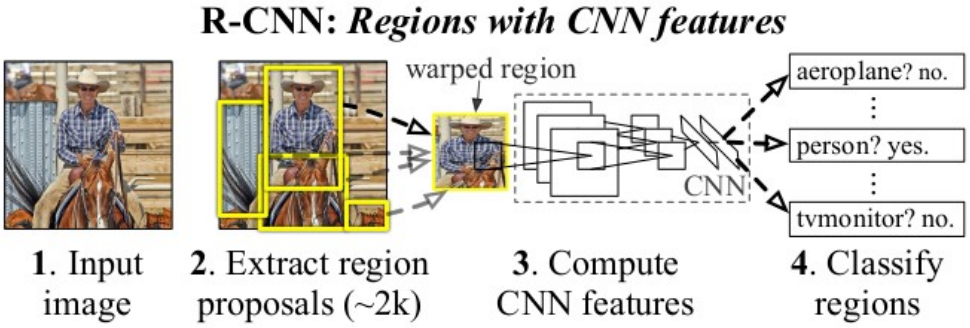


Figura 2.18: Diagrama de flujo de una R-CNN [28].

   b) SPP-net: las capas completamente conectadas reciben entradas de tamaño fijo, es por ello que las **R-CNN** redimensionan o recortan cada región propuesta a un mismo tamaño. Lo anterior implica que un objeto podría no estar contenido dentro de la imagen recortada o podría presentar una distorsión geométrica, lo cual produciría una reducción en la precisión del reconocimiento, especialmente cuando la escala de los objetos es variante.

   Con la finalidad de resolver este problema, se introduce una nueva arquitectura con una capa ***Spatial Pyramid Pooling* (SPP)**, la cual se agrega luego de la

última capa convolucional (ver Figura 2.19). Las capas SPP agrupan las características y generan salidas de un largo fijo, las cuales son las que alimentan las capas completamente conectadas. En otras palabras, se representa un conjunto de información en la etapa más profunda de la jerarquía de la red (entre las capas convolucionales y las capas completamente conectadas) para eludir la necesidad de recortar o deformar la imagen al comienzo del procesamiento [29].

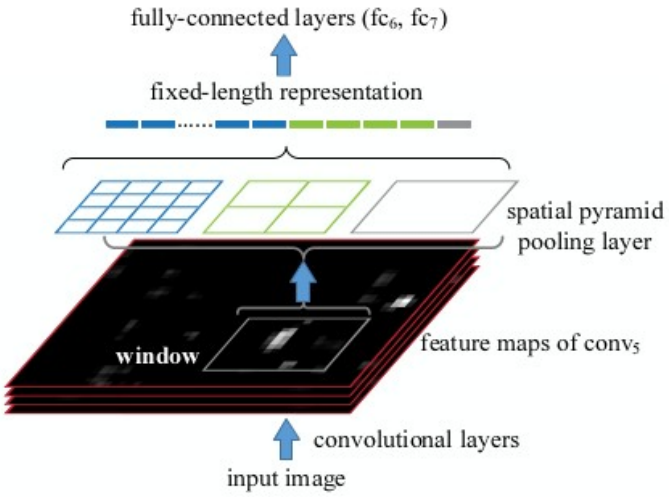


Figura 2.19: Arquitectura de SPP-net [29].

c) Fast R-CNN: a pesar que la arquitectura **SPP-net** muestra mejoras con respecto a la arquitectura **R-CNN**, todavía presenta ciertas desventajas. Ya que al igual que las **R-CNN**, corresponde a un flujo de procesamiento multi-etapa que involucra extracción de características, ajuste de la red, entrenamiento de un modelo SVM y finalmente ajuste de regresores de cuadros de detección; lo que se traduce en un costo de tiempo a considerar en su aplicación. Es por ello que se propone una nueva arquitectura de red llamada **Fast R-CNN** que se observa en la Figura 2.20, ya que es comparativamente más rápida de entrenar y evaluar.

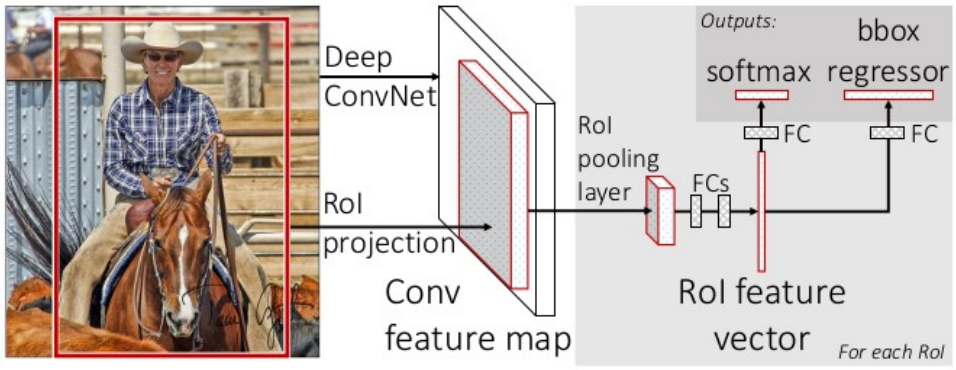


Figura 2.20: Arquitectura de una red Fast R-CNN [30].

Esta red presenta las siguientes ventajas: mayor calidad de detección (MAP) que

R-CNN y **SPP-net**, entrenamiento de una etapa usando función de pérdida multitarea y el entrenamiento puede actualizar todas las capas de la red.

En cuanto al flujo en el procesamiento de la red, ésta toma una imagen y una colección de objetos propuestos, luego procesa la imagen con varias capas convolucionales y capas de máxima reducción para producir un mapa de características. Entonces, por cada objeto propuesto, una capa de reducción de regiones de interés extrae un vector de características de largo fijo de un mapa de característica. Cada vector de características es la entrada de las capas completamente conectadas que, finalmente, se ramifican en dos capas resultantes "hermanas": una que produce la estimación de probabilidad *softmax* sobre las K clases de objetos más una clase *background* y otra capa que entrega cuatro números de valor real por cada una de las K clases de objetos. Cada conjunto de 4 valores codifica las posiciones de los cuadros de detección ajustados para una de las K clases [30].

d) Faster R-CNN: los algoritmos de generación de regiones propuestas como ***selective search*** y ***edgebox***, son el cuello de botella desde el punto de vista computacional en los sistemas de detección. Es por ello que en [31] se propone introducir una *region proposal network* donde su arquitectura se observa en la Figura 2.21 que comparte capas convolucionales con el estado del arte de las redes de detección de objetos. La idea principal se basa en que los mapas de características convolucionales, usados por los detectores basado en región, como el **Fast R-CNN**, también se pueden usar para generar regiones propuestas.

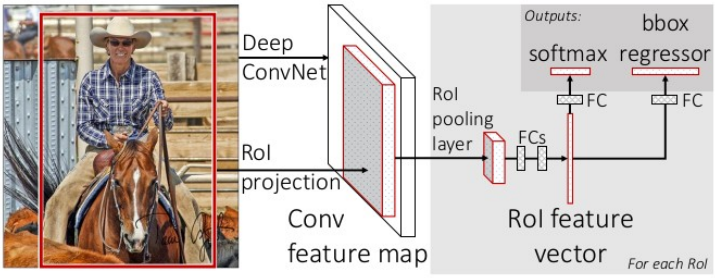


Figura 2.21: Arquitectura de una red RPN [31].

e) *R-FCN*: la familia predominante de redes profundas para detección de objetos es la que se compone de dos subredes: una red convolucional completamente compartida (independiente de regiones de interés) y una subred de regiones no compartida. Esta descomposición se origina de arquitecturas de clasificación pioneras como, **AlexNet** y **VGG16**, las cuales consisten de una subred convolucional y varias capas completamente conectadas separadas por una capa de reducción espacial específica. Las redes de clasificación de imagen recientes, tales como, ***Residual Nets (ResNets)*** y **GoogleNets**, son completamente convolucionales. Para adaptarse a aquellas arquitecturas, es natural construir una red de detección de objetos completamente convolucional sin una subred de región de interés. Sin embargo, aquello no es una buena solución, lo cual se origina debido al dilema entre invarianza de traslación en clasificación de imagen y varianza de traslación en detección de objetos; es decir, la traslación de un objeto dentro de una imagen no es significativo en tareas de clasificación de imagen, mientras que en tareas de detección de objetos

la traslación sí es importante.

Para resolver el dilema planteado anteriormente se inserta una capa de reducción de región de interés en las convoluciones que empeoran la invarianza de traslación, por lo tanto, las capas convolucionales posteriores a las regiones de interés ya no son invariantes de traslación cuando se evalúan a través de diferentes regiones. Sin embargo, este diseño sacrifica eficiencia en el entrenamiento y evaluación dado que introduce un considerable número de capas en términos de regiones. Es por ello que en [32] se desarrolla una arquitectura llamada **Region-based Fully Convolutional Network (R-FCN)** la cual se observa en la Figura 2.22. En ésta, la última capa convolucional produce un total de k^2 mapas de puntuación sensibles a la posición con una grilla fija de $k \times k$ primero y una capa de reducción de región de interés se añade para sumar las respuestas de aquellos mapas de puntuación. Finalmente, las k^2 puntuaciones sensibles a la posición presentes en cada región de interés son promediadas para producir un vector de $C + 1 - d$ y se calculan las respuestas *softmax* en todas las categorías. Otra capa convolucional de $4k^2 - d$ se añade para obtener cuadros de detección genéricos.

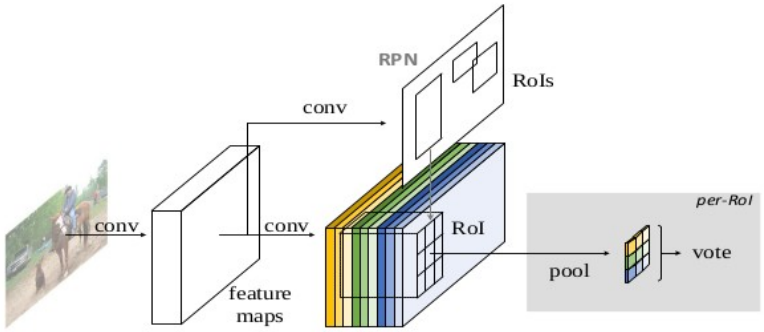


Figura 2.22: Arquitectura de R-FCN [32].

f) FPN: las pirámides de características son un componente básico en los sistemas de reconocimiento para detectar objetos en distintas escalas. Sin embargo, los detectores de objetos desarrollados con técnicas de *deep learning* han eludido las representaciones piramidales, porque son muy costosas en cuanto a cómputo y utilización de memoria. En [33] se propone una arquitectura descendente llamada **Feature Pyramid Network (FPN)**, la cual explota la inherente jerarquía piramidal multi-escala de las redes convolucionales profundas, para construir mapas de características semánticas de alto nivel en todas las escalas. Esta arquitectura muestra una mejora significativa como un extractor de características genéricas en muchas aplicaciones.

En la Figura 2.23(a) se muestra la construcción de una pirámide de características; los atributos se extraen en cada una de las escalas de las imágenes independientemente, lo cual es lento. Los trabajos más recientes de sistemas de detección utilizan un sólo mapa de características (ver Figura 2.23(b)) para acelerar la detección. Otra alternativa que se muestra en la Figura 2.23(c) es reusar la jerarquía de características piramidal calculadas por una red convolucional como si fuese una pirámide de imagen caracterizada. Finalmente, en la Figura 2.23(d) se presenta

la **Feature Pyramid Network**, la cual es tan rápida como (b) y (c), además de ser mucho más precisa que ellas.

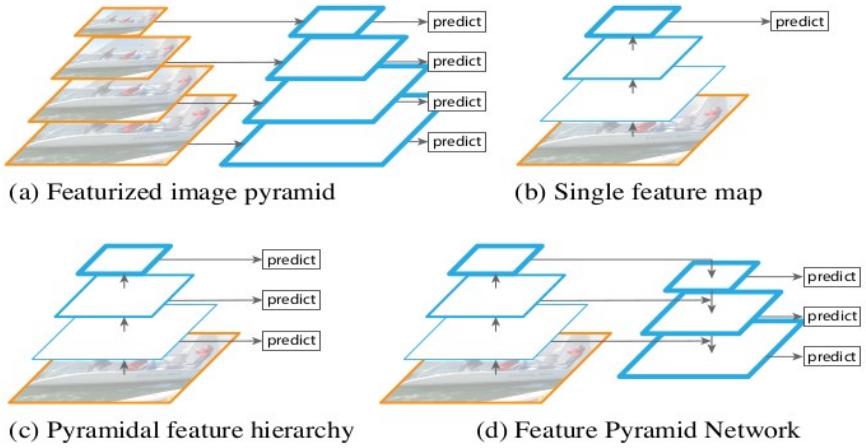


Figura 2.23: Idea principal de una FPN [33].

g) Mask R-CNN: desde un tiempo hasta ahora los resultados de detección de objetos y segmentación semántica han mejorado rápidamente. En gran parte, estos avances han sido impulsados por *frameworks* poderosos, tales como **Fast/Faster R-CNN** y las **Fully Convolutional Network (FCN)**, para detección de objetos y segmentación semántica, respectivamente. Estos métodos son conceptualmente intuitivos, ofrecen flexibilidad y robustez, así como también rápido entrenamiento y tiempo de inferencia. En la actualidad el objetivo que se persigue es desarrollar un *framework* comparable para la segmentación de instancia. La segmentación de instancia es detectar todos los objetos correctamente en una imagen, además de segmentar cada instancia de manera precisa. Por lo tanto, combina elementos de tareas de visión computacional clásicas de detección de objetos, donde se debe clasificar objetos individuales y localizar cada uno usando un cuadro de detección, y realizar segmentación semántica, donde se debe clasificar cada píxel en una colección fija de categorías sin diferenciar instancias de objetos. La arquitectura **Mask R-CNN**, extiende la arquitectura **Faster R-CNN** añadiendo una rama para predecir máscaras de segmentación en cada región de interés, en paralelo con la rama ya existente [34], que realiza clasificación y regresión de cuadros de detección, como se aprecia en la Figura 2.24.

2. ***Framework* basado en Clasificación/Regresión**

a) YOLO: las más recientes arquitecturas para detección de objetos como **R-CNN**, usan el método de regiones propuestas para generar cuadros de detección potenciales en una imagen y, entonces, aplican un clasificador sobre esos cuadros propuestos. Después de la clasificación, realizan un posprocesado para mejorar los cuadros de detección, eliminar detecciones duplicadas y revaluar los cuadros de detección basado en otros objetos en la escena. Este proceso es lento y complejo, además de difícil de optimizar ya que cada componente individual debe ser entrenado separadamente.

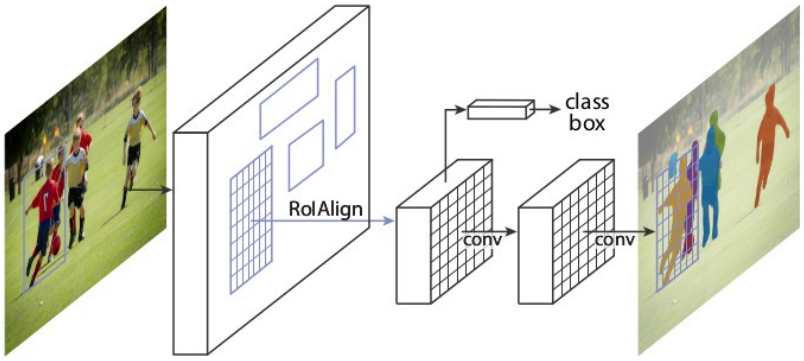


Figura 2.24: La arquitectura de la red Mask R-CNN [34].

Entonces, como una forma de solucionar la problemática anterior es que se origina la arquitectura **YOLO (You Only Look Once)**, la cual aborda la detección de objetos como un problema de regresión único para, espacialmente, separar los cuadros de detección y las probabilidades de clases asociadas. Una única red predice, en una evaluación, cuadros de detección y probabilidades de clases, directamente de las imágenes completas; es decir, *you only look once* ("sólo miras una vez") una imagen para predecir qué objetos están presentes y dónde están. Como se observa en la Figura 2.25, en **YOLO** una única red convolucional simultáneamente predice múltiples cuadros de detección y probabilidades de clases asociadas. **YOLO** se entrena sobre las imágenes completas y directamente optimiza el rendimiento de detección. Este modelo unificado tiene muchos beneficios sobre métodos tradicionales de detección de objetos, siendo uno de los principales la rapidez de detección, sin perjuicio de que está por debajo en precisión de los sistemas de detección a la fecha [35].

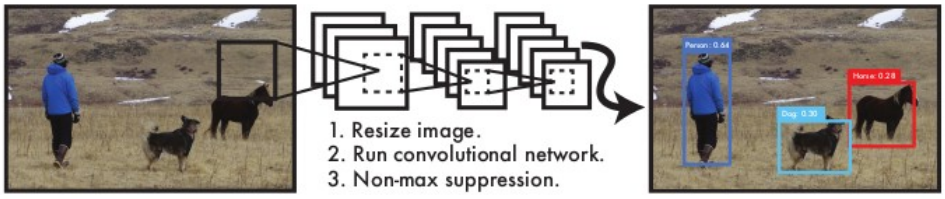


Figura 2.25: El sistema de detección de YOLO [35].

La idea base del modelo, como se observa en la Figura 2.26, es dividir la imagen de entrada en una grilla de $S \times S$, entonces si el centro de un objeto está contenido en una celda de la grilla, cada una de estas es responsable de predecir B cuadros de detección más una probabilidad asociada [35].

b) SSD: **YOLO** tiene dificultades para tratar con pequeños objetos en grupo [28], es por ello que con la finalidad de abordar ese problema, en [36] se propone la arquitectura **Single Shot MultiBox Detector (SSD)**, la cual discretiza el espacio de resultados de los cuadros de detección, en una colección de cuadros por defecto sobre diferentes relaciones de aspecto y escalas por ubicación del mapa de característica. Las contribuciones de la arquitectura **SSD** se pueden resumir en lo

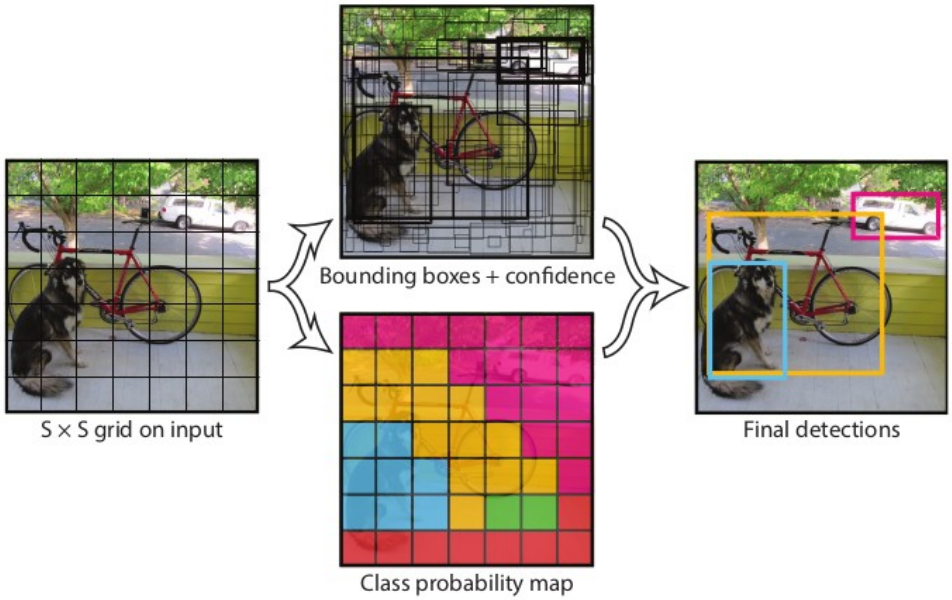


Figura 2.26: El modelo de YOLO [35].

siguientes puntos:

- **SSD** es más rápido que su predecesor en ***single shot detectors*** (YOLO), y significativamente más preciso.
- Es tan precisa como las técnicas más lentas que realizan regiones propuestas y reducción.
- Su diseño de características conduce a un entrenamiento fin a fin y alta precisión, incluso en imágenes de baja resolución.

2.2.2.2. Principales Arquitecturas para la Clasificación de Objetos

1. **MobileNetV2:** arquitectura basada en una estructura residual invertida donde las conexiones residuales están entre las capas del cuello de botella. La capa de expansión intermedia utiliza ligeras convoluciones en profundidad para filtrar las características como fuente de no linealidad [37]. En conjunto, la arquitectura de MobileNetV2 contiene la capa inicial totalmente de convolución con 32 filtros, seguida de 19 capas residuales de cuello de botella (ver Figura 2.27).

   Ésta optimiza el consumo de memoria y la velocidad de ejecución, a partir de un bajo costo en términos de su error. La rápida velocidad de ejecución hace que la experimentación y el ajuste de los parámetros sean mucho más fáciles, mientras que el bajo consumo de memoria es una cualidad deseable en el contexto de un conjunto de redes. Dos de los conceptos más importantes que describen la arquitectura de MobileNetV2 son la convolución separable en profundidad y el residuo invertido, los cuales permiten tener un costo computacional de la convolución menor al convencional [37]. A saber, el conjunto de Ecuaciones 2.12 muestran cómo la convolución con separación en profundidad ($C_{separable}$) permite un menor costo computacional que la convolución convencional (C_{normal}):

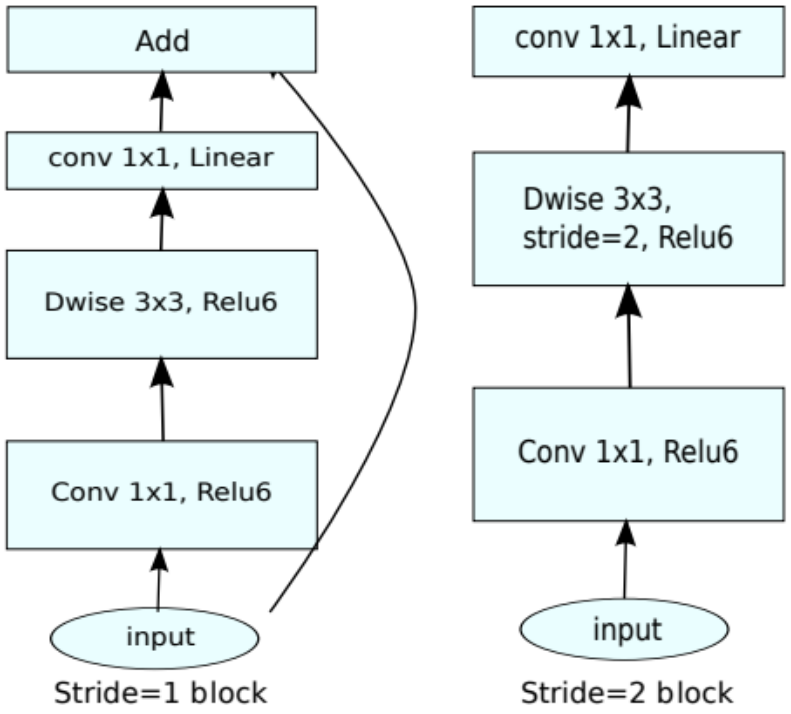


Figura 2.27: Esquema de la arquitectura MobileNetV2 [38].

[37]:

$$C_{normal} = h_i * w_i * d_i * d_j * k^2 \tag{2.12a}$$

$$C_{separable} = h_i * w_i * d_i * (d_j + k^2) \tag{2.12b}$$

Y, dividiendo la Ecuación (2.12a) con (2.12b) :

$$\frac{C_{normal}}{C_{separable}} = \frac{d_j * k^2}{d_j + k^2} \tag{2.12c}$$

donde i corresponde al índice de la capa de entrada, j al índice de la capa de salida, h_i a la altura del mapa de características de entrada, w_i al ancho del mapa de características de entrada, d_i al número de entradas del mapa de características, d_j al número de salidas del mapa de características y k al tamaño del filtro.

2.2.3. Métricas de evaluación

Al desarrollar un algoritmo de *machine learning*, se hace necesario obtener algún indicador que permita conocer objetivamente el desempeño del modelo. Para esto, existe el

concepto de Matriz de Confusión, que cruza los resultados Verdaderos y Falsos reales con los obtenidos por el modelo a evaluar. A saber, la Figura 2.28 muestra un ejemplo de Matriz de Confusión para una clase de clasificación, donde "1" indica que es verdadero y "0" que es falso.

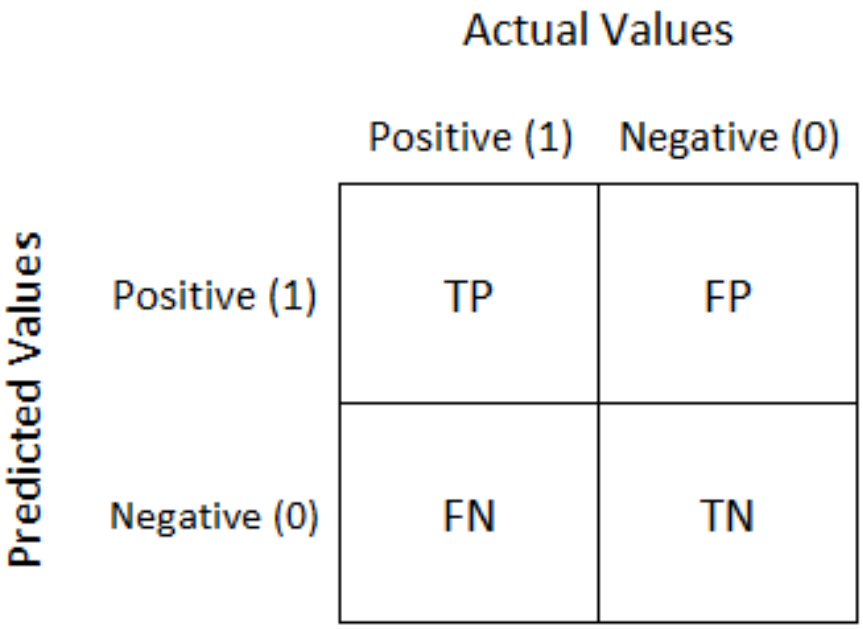


Figura 2.28: Ejemplo de la Matriz de Confusión [39]

A partir de la Matriz de Confusión es posible obtener métricas de evaluación que ayudan a leer y comprender, de mejor manera, los resultados de la Matriz de Confusión. Estos se presentan a continuación [39].

- **Precision**. La métrica Precision (precisión) mide la calidad de un algoritmo de *machine learning* en tareas de clasificación. La Ecuación 2.13 es la usada para calcular esta métrica.

$$precision = \frac{TP}{TP + FP} \tag{2.13}$$

- **Recall**. La métrica Recall (exhaustividad) mide la cantidad que un módulo de machine learning es capaz de identificar. La Ecuación 2.14 es la utilizada para calcular esta métrica.

$$recall = \frac{TP}{TP + FN} \tag{2.14}$$

- **F1**. Corresponde a una forma de representar las métricas anteriores (Precision y Recall) en un sólo valor, con el fin de poder comparar el rendimiento combinado entre diversas soluciones de *modelos*. La Ecuación 2.15 es la utilizada para obtener esta métrica.

$$F1 = 2 * \frac{precision * recall}{precision + recall} \tag{2.15}$$

Por otro lado, cuando la tarea realizada por un algoritmo de *machine learning* corresponde a detectar objetos, se utilizan dos métricas principalmente, a saber [39]:

- **Intersection over Union (IoU)**. La métrica IoU (Intersección sobre Unión) evalúa la superposición de los *bounding box* de la detección resultante (del modelo) con la detección real (proveniente del etiquetamiento), donde "1" representa a una superposición exacta y "0" a que no existe superposición. La Ecuación 2.16 es la usada para calcular esta métrica.

$$IoU = \frac{area\ of\ overlap}{area\ of\ union} \tag{2.16}$$

- **Mean Precision Average (mAP)**. La métrica mAP (Precisión Promedia Media) corresponde a la media de las Precisión Promedia (AP) de cada clase de detección, donde AP, por su parte, evalúa qué tan útil resulta ser la tarea de detección para la clase en cuestión. Entonces, existen tantos valores de AP como clases de detección existan, pero solo un valor de mAP que evalúa la eficacia de la tarea de detección del modelo. La Ecuación 2.17, que corresponde al área bajo la curva que se obtiene al graficar el Recall y Precision, es la utilizada para calcular cada AP.

$$AP = \int_0^1 p(r)\, dr \tag{2.17}$$

2.3. Estado del arte

En este apartado se revisan algunas publicaciones de otros autores, los cuales si bien no buscan el mismo objetivo general, utilizan herramientas que pueden ser útiles para resolver la problemática de este informe: el monitoreo de los postes de tendido eléctrico para prevenir fallas. Se encuentran distintas formas de enfrentar el problema y es por ello que a continuación se muestra cómo abordar un proyecto similar bajo las herramientas de visión computacional.

2.3.1. *Intelligent Image Recognition Research on Status of Power Transmission Lines* [40]

En este artículo se busca un monitoreo remoto e inteligente del estado en las líneas de transmisión de poder. El autor de este artículo propone utilizar técnicas de procesamiento de imágenes, siendo éstas las tradicionales (visualizadas en la Figura 2.29)

1. Obtención de la imagen en RGB.
2. Transformación a escala de grises.
3. Arreglo en el histograma.
4. Detección de bordes.

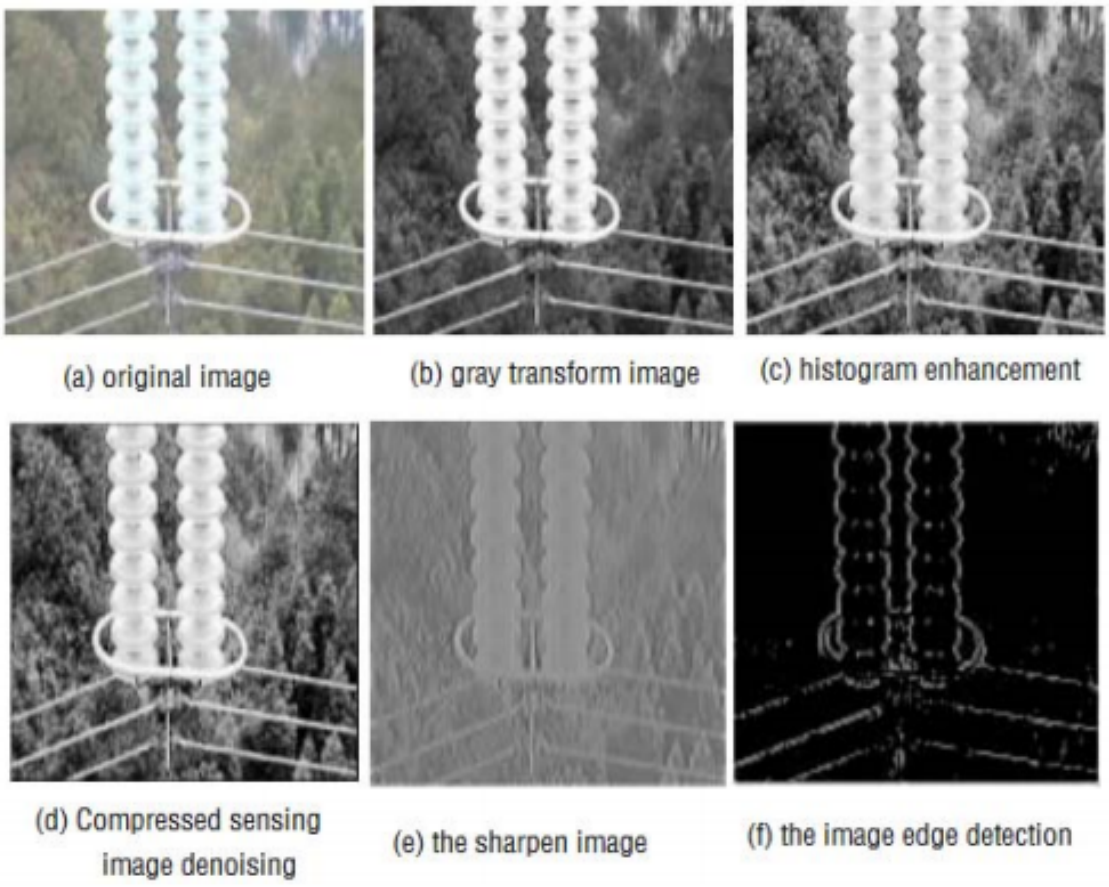


Figura 2.29: Metodología a aplicar para la detección de bordes [40].

Para el problema de este artículo en revisión, y tal como se aprecia en la Figura 2.29, como lo que se desea monitorear son las líneas eléctricas de poder, entonces la detección de bordes resulta ser una herramienta útil para visualizar y decidir si el estado de ese tramo se encuentra en perfecto estado o no, al compararlo con una base de datos de tramos en buen estado. Esto se podría replicar en la problemática del presente trabajo, con los distintos elementos de interés para monitorear en un poste de tendido eléctrico.

2.3.2. *An intelligent monitoring system based on Internet of Things for electric distribution network* [41]

Debido al explosivo crecimiento de las redes de transmisión eléctrica en China, se hace aún más necesario resguardar el estado de la red eléctrica. La operación tradicional de mantenimiento, que involucra capital humano, ya no es óptimo tanto por el avance tecnológico de la red eléctrica, lo que involucra una capacitación a todo el personal, como también al tamaño que han alcanzado las redes de transmisión, que está directamente relacionado al crecimiento demográfico de China. El artículo de la presente sección propone una solución a través del uso del *Internet of Things* (IoT) para construir un sistema inteligente de monitoreo de las redes de distribución eléctrica del país asiático.

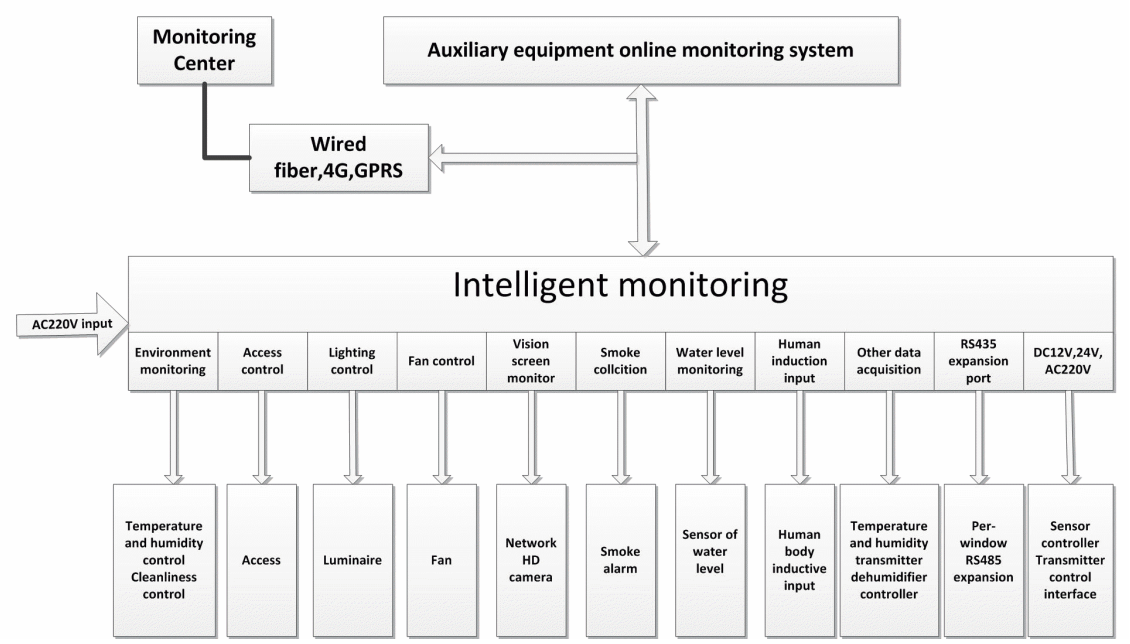
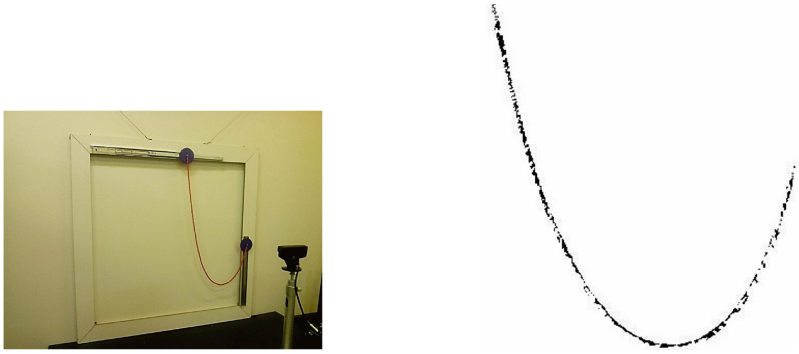


Figura 2.30: Diagrama esquemático de la composición del sistema inteligente de monitoreo de la red eléctrica en China [41].

2.3.3. *Extracting of Sagging Profile of Overhead Power Transmission Line Via Image Processing* [42]

En una línea de transmisión, la flacidez del cable conductor tiene una vital importancia en la seguridad, confiabilidad y eficiencia de la transmisión de potencia. Una línea de transmisión bien diseñada garantiza la máxima capacidad de carga estática. Esto se hace manteniendo el espacio vertical mínimo entre los cables, de una torre a otra, y el suelo. Sin embargo, el aumento de la longitud del cable entre dos postes de tendido eléctrico conduce a un alto costo de pérdida de material y energía eléctrica, además de aumentar la posibilidad de intervención. Por otro lado, la reducción de la curvatura en caida del cableado eléctrico induce una alta tensión en el conductor, lo que puede provocar daños en la línea. Para garantizar un perfil de flacidez de seguridad, una inspección es esencial en el establecimiento y mantenimiento de las líneas de transmisión de energía. En el artículo que se revisa en esta sección, primero se desarrolla la formulación matemática de cables largos y pesados, en seguida de un método mediante procesamiento de imágenes para obtener el valor de la flexión del cable.



a Montaje experimental

b Imagen binaria del perfil del cable

Figura 2.31: Montaje experimental y resultado del procesamiento de imagen [42].

2.3.4. *Power line detection using Hough transform and line tracing techniques* [43]

En las redes eléctricas, la rápida detección de sus líneas de tensión en imágenes es útil en aplicaciones de fotogrametría como, por ejemplo, la medición del voltaje en el cable y su curvatura. Para realizar este tipo de mediciones, se utilizan imágenes obtenidas de tramos de líneas eléctricas, las cuales pueden incluir grandes cantidades de curvatura en los cables. Los trabajos sobre detección de líneas eléctricas, que preceden al artículo que se revisa en esta sección, han centrado su metodología en imágenes aéreas o de proximidad cercana, donde no es posible visualizar curvaturas de la línea eléctrica. Este documento evalúa la viabilidad de las imágenes terrestres utilizando cámaras de teléfonos inteligentes, junto con la detección rápida y robusta de la línea de alimentación, utilizando dos técnicas comunes de transformación de Hough y un algoritmo de trazado de línea.

El método utilizado por los autores del documento en revisión, consisten en la aplicación de técnicas de procesamiento de imágenes. Como se aprecia en la Figura 2.32, en el sentido de las agujas del reloj desde la parte superior izquierda, se tienen: imagen de entrada, aplicación de la transformación de perspectiva, filtrado en la imagen, detección de las líneas eléctricas, aplicación de la transformación de perspectiva inversa a los cables, finalmente, se distinguen las líneas separadas en la imagen original.

Figura 2.32: Método utilizado para detectar líneas de la red eléctrica [43]

La problemática del artículo que se revisa se acerca bastante a lo que se desea solucionar en este trabajo. Sin embargo, la mayoría de las imágenes que se analizan de la red eléctrica en el artículo, presentan un escenario visual adecuado en cuanto al fondo que se tiene en la posición del cableado eléctrico, ya que es un color casi uniforme (el cielo) y, además, contrasta muy bien con los cables. Esto indica que la solución propuesta en el documento en revisión es óptima si se aplica en un lugar que cumpla esas características, como, por ejemplo, algún pueblo pequeño o similar. Sin embargo, las características del entorno con el que se trabajarán son, en su mayoría, zonas urbanas, con edificios de una altura considerable. Por consiguiente, la hipótesis sugiere que la aplicación de la transformada de Hough no será útil para detectar y filtrar las líneas de la red eléctrica nacional.

No obstante, existe una técnica utilizada por los autores del paper en revisión, que sí podría ser útil en el desarrollo del proyecto. Esta es la transformación de perspectiva (segunda imagen superior de la Figura 2.32), que transforma la imagen de tal manera que al entregarle la ubicación de los pixeles que son vértices de una área trapezoidal de interés, se genera una nueva imagen donde esa área se alínea con tal de conformar el plano de esta nueva imagen. Esta transformación de la perspectiva es necesaria para realizar la parabolización de las imágenes tomadas desde varios ángulos.

Para la aplicación correcta de la transformación de la perspectiva en el contexto del *paper*, se requiere alguna información adquirida previamente, a saber: la distancia a los polos, las pixelaciones de los polos en las imágenes y los datos intrínsecos de las cámaras.

Capítulo 3

Metodología

3.1. Procesos del proyecto

El flujo operacional del presente proyecto considera las siguientes acciones: imágenes o videos de la infraestructura de la red eléctrica de distribución entran al modelo computacional como el archivo principal de entrada. Luego, el modelo mediante varias etapas (llamados módulos) procesa, calcula y detecta los elementos de la red eléctrica que interesan, tales como los postes, las catenarias, la altura de éstas con respecto al suelo como para determinar si es peligrosa, etc. Finalmente, todos los resultados de interés son mostrados en la plataforma de Sistema de Información Geográfica (SIG); información lista y precisa para que las empresas de distribución eléctrica la utilicen en sus procesos de mantención preventiva de la red eléctrica.

A partir de lo anterior, en lo que respecta al desarrollo del presente proyecto, éste se divide en tres grandes procesos, los cuales son:

- **Obtención de los datos**. Los datos del proyecto, que consistirán en imágenes y/o videos de la infraestructura eléctrica de distribución, deberán ser definidos en cómo se obtienen, dado que inicialmente no se cuenta con algún registro visual de tramos de la red eléctrica como para ser usado indirectamente en este proyecto. Esto implica que tanto los datos de entrenamiento, de prueba y de validación que se ocuparán en el desarrollo del modelo computacional, serán obtenidos por los autores. Y bajo esas características, que se definirán en la obtención del recurso visual, serán las necesarias a considerar para la operación futura del presente proyecto.
- **Procesamiento de los datos**. Bajo los requerimientos que presenta este proyecto, se deben crear algoritmos computacionales que obtengan como resultado lo solicitado y funcionen conjuntamente en el modelo operativo. Por ejemplo, uno de estos módulos, al entregarle los datos de entrada correspondientes, debe ser capaz de detectar y entregar al usuario las catenarias críticas, mientras que otro se encargará de entregar su posición con un cierto margen de error.
- **Presentación de los resultados**. Posteriormente, los resultados que se obtendrán en el modelo computacional serán procesados en una plataforma para que el usuario pueda acceder a ellos de manera clara, permitiendo una mejor toma de decisiones en relación a la mantención de la infraestructura de la red eléctrica.

De lo anterior, las dos primeras partes corresponden al trabajo que el autor de esta Memoria ha realizado, ya que son parte del objetivo general ya señalado en la sección de Introducción. Los objetivos espécificos, por su parte, serán parte directa de la segunda gran etapa: Procesamiento de datos. Pero para poder trabajar en esta etapa es necesario definir cómo será la obtención de datos y las características que presentarán los archivos, los que serán la unidad de entrada a los modelos computacionales de detección y decisión. Esto se verá a continuación.

3.2. Montaje para la obtención de videos

Para obtener registros de videos de los sectores de interés en los que se quiere detectar eventos de la red eléctrica de distribución, se decidió el siguiente montaje:

- Un vehículo con el cual se recorrerá el área de interés y en el cual estará colocada la cámara que registrará el recorrido.
- Una cámara IP de 12V, tal que pueda ser alimentada por una batería y pueda funcionar mediante IoT (vía wifi)
- Un router, que estará alimentado por una batería y proporcionará conexión a Internet para conectar la cámara con el dispositivo que la controlará.
- Un Notebook, que se encargará de controlar la cámara (ejecutar el grabado, enfoque/desenfoque, etc.). Éste debe poseer un controlador para conectarse a Internet vía inalámbrica y el software correspondiente a la cámara IP.
- *Software* para registrar la ubicación del recorrido. El archivo que genere el programa a utilizar debe tener las extensión ".gpx" o ".kmz".
- Un inversor, que transformará la corriente de la batería.
- Una batería.
- Otros implementos que permitan fijar la cámara al vehículo, mantengan el orden en las conexiones, alargador si es necesario, etc.

En la Figura 3.1 se aprecia el montaje para poder realizar la grabación de videos. Además, en el Anexo A - Detalles de los recorridos, se puede encontrar más información acerca de las cámaras utilizadas, las rutas realizadas y las problemáticas que se generaron durante el proceso.

3.3. Características del video

Los videos poseen un códec de H.264, formato ".avi", dado que permite una buena resolución y un bajo tamaño del archivo. En la Figura 3.2 se aprecia un ejemplo de un video capturado por los autores.

Figura 3.1: Vehículo con la cámara instalada para obtener videos del recorrido. [Original de los autores, 2020].

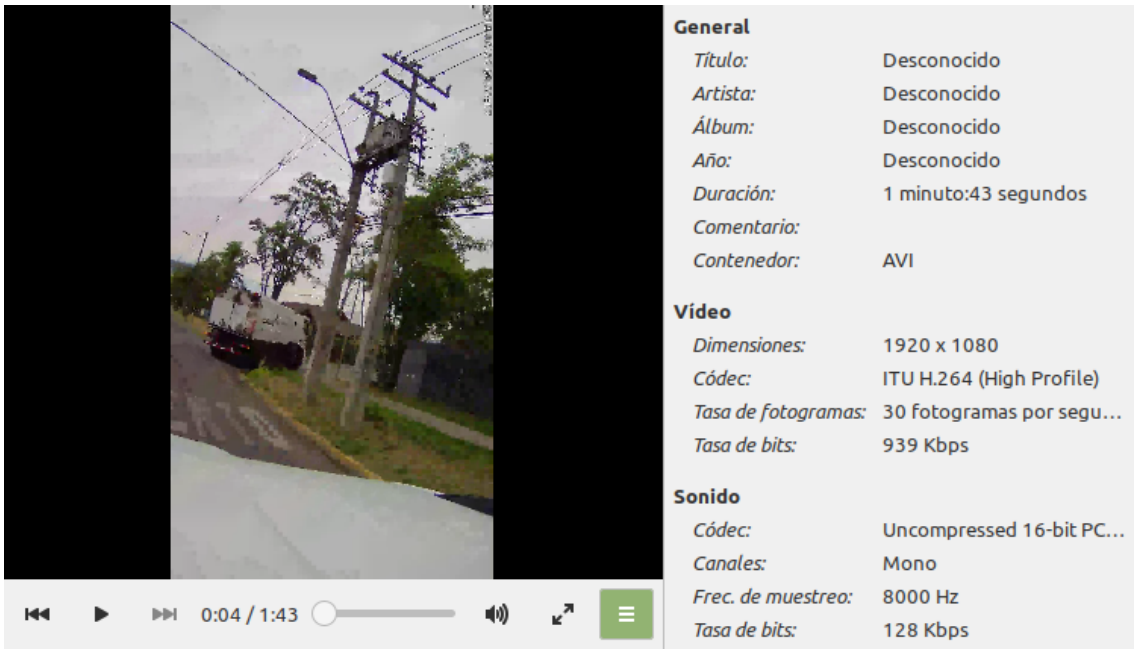


Figura 3.2: Ejemplo de video capturado para usar en los modelos computacionales [Original de los autores, 2019].

3.4. Transformación a secuencia de imágenes

Para trabajar con los videos capturados en los módulos computacionales, se realizó un *script* en el lenguaje de programación de Python que transformará un video en una secuencia de imágenes, con un salto de *frames* definido por el usuario. En esta ocasión, se decidió que las secuencias de imágenes fueran cada 10 *frames* del video capturado, dado que todos los eventos de importancia que se presentan en el video no son omitidos en el conjunto de imágenes y la posición, en términos de pixeles de un evento entre una secuencia y otra, tiene una diferencia aceptable para el caso; de lo contrario, la diferencia de la posición de un evento entre una imagen y otra sería mínima y podría generar problemas en el entrenamiento y validación de los modelos computacionales, puesto que, prácticamente, la misma imagen estaría presente en ambos conjuntos. Lo anterior queda reflejado en la Figura 3.3.

Por otro lado, en términos de tamaño, si por ejemplo un video que dura 6 minutos con 23 segundos y pesa en memoria 56.7 MB, ingresa al *script* desarrollado y se fija un salto de 1 solo *frames* entre una secuencia y otra, entonces se transforma en una secuencia de imágenes compuesta por 9570 elementos y un peso total de 2750.3 MB. Lo anterior explica porqué se realiza el salto de *frames*, ya que permite disminuir la cantidad de imágenes y, por ende, el peso total de ellos al realizar la transformación de un video a un conjunto de imágenes. Siguiendo el ejemplo, si se considera un salto de 10 *frames*, entonces se obtendrían solo 957 imágenes y con un peso total de 275.3 MB.

El motivo por el que se busca disminuir la cantidad de imágenes (y su peso total) es debido al costo computacional de etapas posteriores; a saber, al procesar un video en el modelo de detección, serán menos los *frames* a analizar y, por ende, disminuirá la cantidad de archivos temporales y tiempo de ejecución que si no se hubiese ocupado un salto de *frames*.

3.5. Etiquetamiento manual de los eventos de interés

El último proceso antes del desarrollo, entrenamiento y validación de los algoritmos computacionales de cada módulo, consiste en etiquetar manualmente los eventos de importancia que se presentan en el conjunto de imágenes obtenidas en la etapa anterior. Éstos son: los postes y las catenarias (sean éstas peligrosas o no). Para lograrlo se ocupó una herramienta computacional diseñada por FORCAST que facilita el etiquetamiento manual, permitiendo exportar todos los objetos etiquetados en un archivo de extensión .csv que contiene principalmente la ubicación de los vértices del *bounding box* de cada objeto etiquetado.

La regla que se siguió para el etiquetamiento consideró lo siguiente:

1. **"Poste"**
    - Se debe encerrar el poste que es visible en sus partes inferior y superior.
    - El *bounding box* debe ser lo más cercano posible al poste, sin encerrar áreas innecesarias. Esto es importante para la precisión del modelo que estima alturas, que se presenta más adelante.
    - No encerrar los postes a los cuales no es posible identificar los cables asociados a él.

a Sin salto de *frame*

b Con salto de 10 *frame*

Figura 3.3: Secuencia de imágenes con y sin salto de *frames* [Original de los autores, 2019].

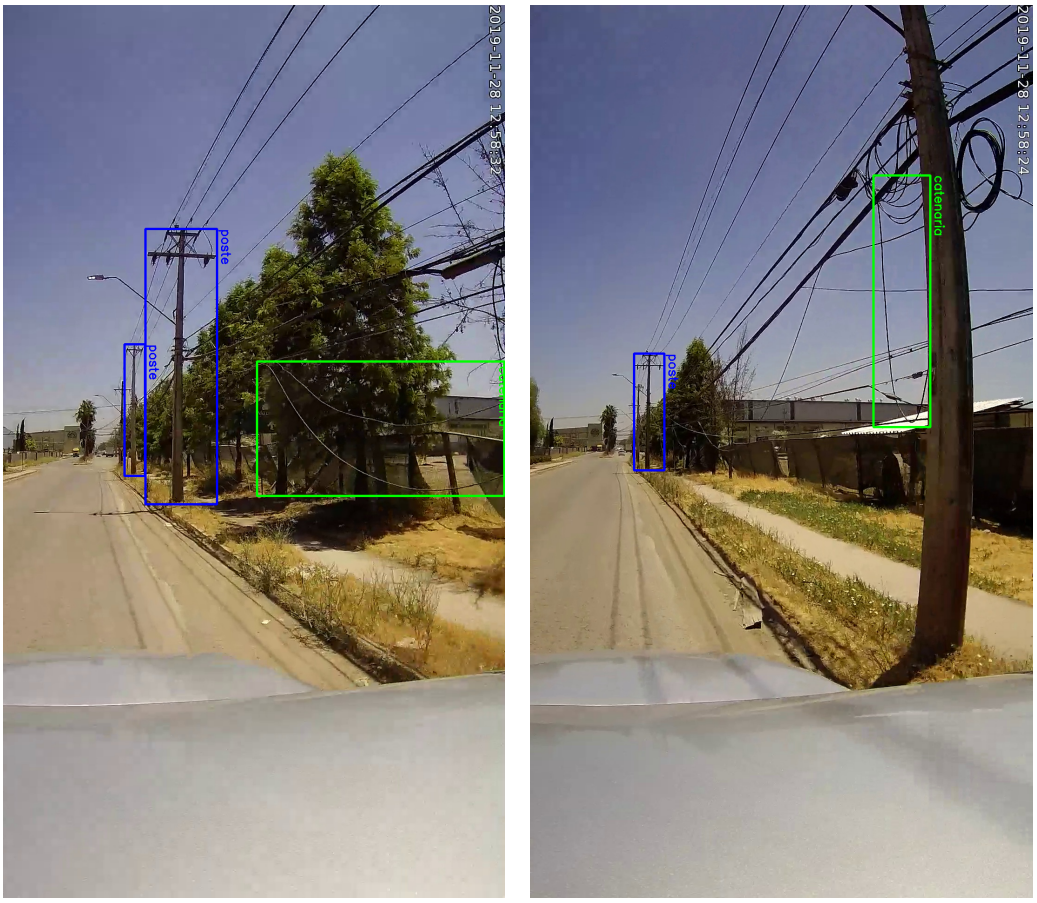
2. **"Catenaria"**

   - Considerando la catenaria como una parábola, su vértice debe estar a una altura menor con respecto al cable de red más cercano al suelo.
   - El área a encerrar deben ser donde parte la parábola, además de no encerrar áreas

innecesarias.

- Si hay más de una curvatura en una misma sección de cables, solo encerrar la que se encuentra más cercana al suelo.

En la Figura 3.4 se muestran ejemplos se etiquetamiento para estas dos clases.



a Etiquetamiento de todos los elementos.

b Elemento no cumple con reglas de etiquetamiento.

Figura 3.4: Ejemplo de etiquetamiento de la clase "poste" en una imagen, aplicando *bounding box* a cada objeto [Original de los autores, 2019].

El etiquetamiento se debe realizar a todo el *set* de imágenes. Luego, se considera todo ese *set* para el entrenamiento y validación de los distintos módulos que presentará el modelo computacional. Además, es importante señalar que no se realizó la separación del conjunto de imágenes en dos partes (entrenamiento y validación), ya que debido a que el *set* es amplio y para evitar el fenómeno de *overfitting* en los entrenamientos, al comienzo de cada iteración se realizá la separación, con las imágenes escogidas al azar, de todo el conjunto en dos partes: la de entrenamiento que considera el 70 % del total, y la de validación, que se considera el 30 % dentro del total.

Por otro lado, y si bien la cantidad de postes que se etiquetan resultan ser mayores a los de la clase catenaria, se debe procurar que la presencia de cada clase de evento ("poste", "catenaria"), en cada conjunto, cumpla con alguna proporcionalidad. De lo contrario, podría darse el caso que la detección de un evento no quede bien desarrollada, o no sea validada correctamente, si es que en algunas iteraciones del entrenamiento se presenta una proporción casi nula de las clases en el conjunto de entrenamiento o de validación, respectivamente.

Capítulo 4

Trabajo Realizado

A continuación se presentan los módulos realizados, explicando su funcionamiento y mostrando su desempeño.

4.1. Módulo de detección de eventos

Con el conjunto de imágenes ya etiquetado se diseñó el primer modelo computacional que fuera capaz de detectar las distintas clases de eventos de interés al entregarle un video como archivo de entrada. Lo complejo del módulo recae en la detección de eventos, el cual se logra mediante el desarrollo, adaptación, entrenamiento y validación de redes neuronales. Para lograrlo, se utilizó la API Object Detection de la librería de Tensorflow para el lenguaje de programación Python, la cual se debió adaptar al contexto de la problemática y de las características que presenta el conjunto de imágenes. Luego, para ajustar los pesos de la red neuronal, se realizaron varias iteraciones en el entrenamiento, hasta lograr los resultados esperados, que fueron validados con el conjunto de validación.

Algunos resultados que se pueden extraer de la validación son los siguientes:

- **Precisión de los *bounding box*: 76 %**. Esto es para los objetos de las distintas clases que sí logra detectar, pero que no los encierra a la perfección en comparación al etiquetamiento manual que se mencionó en la sección anterior. El valor porcentual hace referencia al área que sí se logró encerrar en un *bounding box*, donde se tiene que las mayores áreas que no lograron encerrarse totalmente fueron a los objetos de la clase "poste". En cambio, para los de la clase "catenaria", generalmente se encierra una área mayor a la que realmente corresponde.
- **Porcentaje de pérdida: 5 %**. Quiere decir que, prácticamente, 1 de cada 20 eventos no son detectados por el modelo. Detalladamente, esto ocurre principalmente con la clase "catenaria", debido a que el área que representa tal fenómeno es muy variable y, además, las capas de fondo de donde se produce el evento "catenaria" dependen de la ubicación de este y de donde se realizó la captura. Por ejemplo, a veces la capa de fondo corresponde al cielo, mientras que en otras puede corresponder a un edificio o a un árbol, lo que genera confusión en el aprendizaje y, por ende, en la detección cuando se usa el módulo.

Teniendo los pesos ajustados de la red neuronal que realiza la detección, se diseñó un *script* que al recibir un video fuera capaz de detectar y encerrar en un *bounding box* cada clase de evento, junto a su *confidence score* o porcentaje de confianza en la detección. En la Figura 4.1 se aprecian algunos *frame* del video de salida que genera el modelo.

En términos numéricos, los resultados del modelo computacional de detección, aplicada a algunos videos de prueba de corta duración, se muestran en las Tablas 4.1 y 4.2.

Tabla 4.1: Matriz de confusión de la clase "poste", normalizado a 100 eventos

		Resultado de la detección		
		p	**n**	**total**
Valor real	**p′**	91	2	P′
	n′	7	0	N′
	total	P	N	

Tabla 4.2: Matriz de confusión de la clase "catenaria", normalizado a 100 eventos

		Resultado de la detección		
		p	**n**	**total**
Valor real	**p′**	62	8	P′
	n′	30	0	N′
	total	P	N	

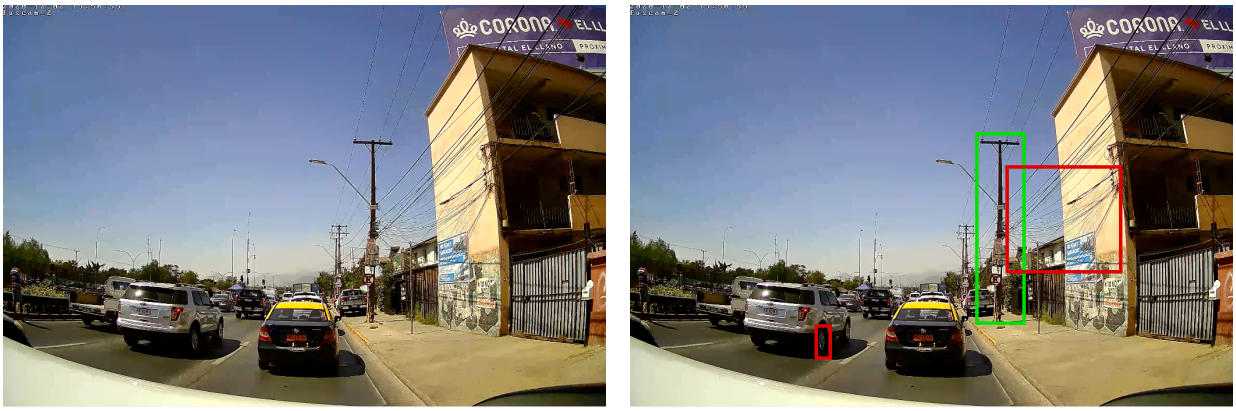
A partir de los resultados tabulados, se pueden obtener las métricas de evaluación para estos tipos de modelos. A saber, en la Tabla 4.3 se muestran estos, donde se destaca un alto rendimiento para la clase "poste". Sin embargo, para la clase "Catenaria" se tiene un valor de

Precision bajo el 0.7, lo que significa que casi $\frac{1}{3}$ de las catenarias detectadas por el módulo, en realidad, no son catenarias.

Tabla 4.3: Métricas de evaluación del módulo de detección de eventos.

	Poste	**Catenaria**
Precision	0,9286	0,6739
Recall	0,9785	0,8857
F1	0,9529	0,7654

Lo anterior ocurre debido a diversos motivos: los eventos de catenarias no ocurren con una alta frecuencia en un video a diferencia de la aparición de postes, lo que puede generar que el módulo considere como "catenarias" ciertos objetos que no lo son (Figura 4.1 (b)) a partir del entrenamiento no balanceado. Además, como se observa en la Tabla 4.2, ciertos eventos de catenarias no son detectados por el modelo y esto corresponde a cuando se tiene un cable de una larga longitud que posee una curvatura pronunciada, o cuando se entremezcla visualmente con un árbol u otra capa de fondo que produzca tal fenómeno visual.



a Imagen original. b Resultados módulo de detección de eventos.

Figura 4.1: Ejemplo de resultado en módulo de detección de eventos [Original de los autores, 2021].

Entonces, se hace necesario un post-procesamiento de los resultados de detección de la clase "catenaria".

4.2. Módulo de optimización catenarias

En este módulo se optimiza la detección de catenarias, utilizando información del módulo anterior y técnicas de procesamiento de imágenes. Para ello, se realiza una serie de etapas de filtrado:

- Primero, se verifica que el objeto detectado como "catenaria" haya sido detectado en el video una cierta cantidad de veces. El umbral depende del valor del salto de *frames* y de un parámetro definido por el usuario.

- Luego, se analiza que los *frames* que contienen ese evento presente, a lo menos, dos postes detectados. Esto se realiza a partir de información temporal generada por el módulo anterior.
- Finalmente, se verifica que la catenaria esté entre ellos (o alguno de ellos). Este proceso se realiza mediante técnicas de procesamiento de imágenes, destacando el uso de una máscara dinámica, que depende de la ubicación de los pixeles de los postes.

Mediante estos pasos, se logra eliminar una gran cantidad de falsos positivos de "catenarias" (cerca del 90 % de los FP de los resultados del módulo anterior). Como se aprecia en la Figura 4.2, se elimina la "catenaria" mal detectada que estaba presente en la Figura 4.1(b).

Figura 4.2: Ejemplo de resultado en módulo de optimización de catenarias [Original de los autores, 2021].

Finalmente, como se redujo en un 90 % los falsos positivos para la clase "catenaria", se tiene que el nuevo valor de *precision* es de 0,954, junto a un F1 actualizado de 0,9186. Ahora queda determinar si la catenaria es peligrosa o no.

4.3. Módulo de clasificación de catenaria peligrosa

Este módulo, para determinar si una catenaria es peligrosa o no, ocupa como *input* imágenes en donde la catenaria y los postes asociados a ellas estén presentes y, luego, mediante operaciones que consideran la ubicación de estos objetos en unidades de ´pixeles, encuentra a cuántos pixeles se encuentra la catenaria con respecto al suelo. Finalmente, mediante el valor en metros de esos postes (ya sea por mediciones reales o estimadas), se aplica la Ecuación

4.1, que consiste en una regla de tres entre el valor de los pixeles y la altura del poste.

$$Altura_{catenaria}[m] = \frac{(pixel\_inf_{catenaria} - pixel\_inf_{poste})[pixeles] * Altura_{poste}[m]}{(pixel\_sup_{poste} - pixel\_inf_{poste})[pixeles]} \quad (4.1)$$

El *frame* que se considera para aplicar la Ecuación 4.1 es el que presentó un mayor porcentaje de precisión en el *bounding box* de detección para el evento de la clase "catenaria" en cuestión (en la Figura 4.3 se puede apreciar esto). Luego, se considera el poste detectado más cercano (si es que hay más de uno) a la catenaria. Entonces, con el valor anterior, pasa a una simple comparación con el valor de 4.5[m] (que es la altura mínima habitual que las ordenanzas municipales establecen), teniendo dos escenarios de clasificación:

- Si es mayor o igual a 4.5, entonces se considera como una catenaria **NO** peligrosa.
- Si es menor a 4.5, entonces **SI** corresponde a una catenaria peligrosa.

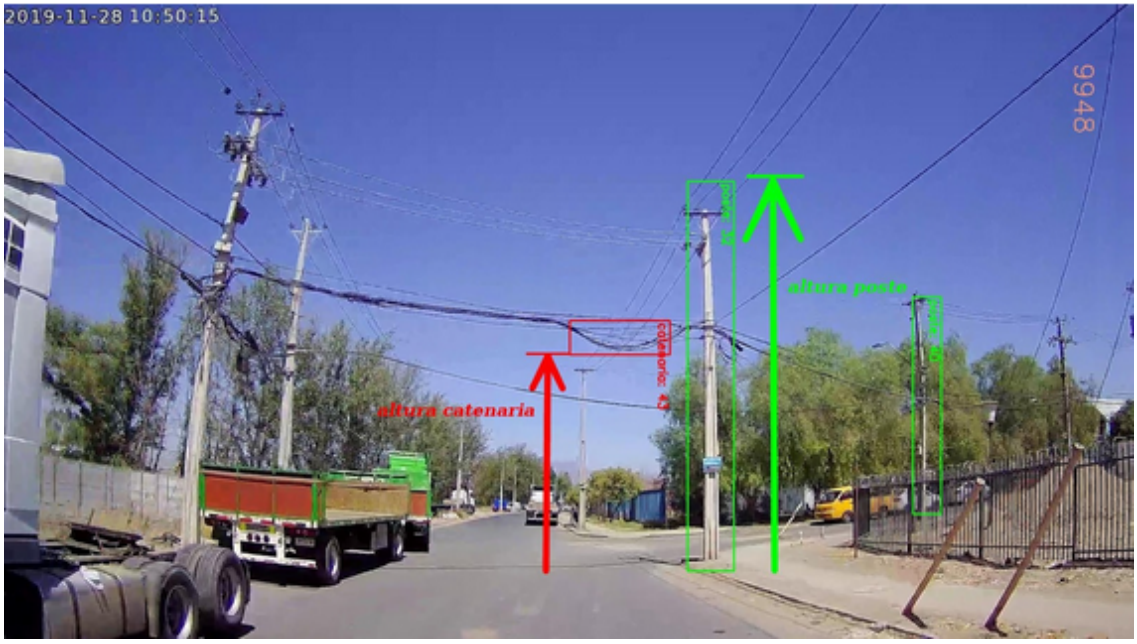


Figura 4.3: Catenaria y poste a considerar para cálculo de altura. [Original de los autores, 2021].

4.4. Módulo de georreferenciación

Considerando el archivo georreferencial ".gpx" o ".kmz", se hace coincidir la fecha de los registros de ubicación con la fecha del evento. Para ello, los pasos a seguir son los siguientes:

- Se toma el nombre del video, que es generado automáticamente por la cámara IP, el cual contiene la hora, minuto y segundo en que comienza ese registro.

- Luego, para el *frame* que contiene los elementos a georreferenciar, mediante el valor de FPS (*frames per seconds*) y del número de *frame*, se obtiene el tiempo de registro de ese evento.

- Posteriormente, sobre el archivo de georreferenciación se busca esa hora, minuto y segundo para asociarle la posición espacial. En caso de que no lo encuentre, entonces busca los dos valores más cercano (el más cercano por abajo y por arriba) y realizando una operación de promedio, se obtiene un estimado de la ubicación georreferencial de ese evento.

Este módulo no presenta fallos y su desempeño es siempre correcto, si se le entregan los *input* correspondientes.

4.5. Módulo de clasificación de postes defectuosos

Ocupando las detecciones de los postes como entrada, este módulo analiza esas imágenes para determinar si presentan algún defecto en su infraestructura o no. El proceso lo realiza mediante *deep learning*, al igual que el módulo de detección de postes y catenarias.

Sin embargo, este módulo no ha tenido buenos resultados debido a que las imágenes de los postes no son nítidas, ya que como se puede apreciar en la Figura 4.4, en (a) se tiene la imagen que se guarda temporalmente en el módulo de detección anterior sobre el mejor *frame* en donde se tiene el poste de ID "XX" (donde el criterio es que esté lo más cercano a uno de los bordes y, además, se pueda distinguir en la imagen la parte inferior y superior del poste); luego, se realiza la extracción del poste y se guarda en una imagen (b) para que el modelo lo utilice en algún otro módulo. Entonces, si el módulo de clasificación de postes defectuosos utiliza esa imagen de entrada, determina que el poste si presenta una falla.

No obstante, para el mismo poste del ejemplo, si se tiene una imagen más cercana de él (que corresponde a *frames* más adelantes en el video, donde se pierde la parte superior del poste en el registro audiovisual), que corresponde a la imagen (c), se aprecia que no corresponde a una falla en la infraestructura, sino a una área oscurecida del poste (producto de alguna contaminación visual pasada).

Como el módulo lleva a clasificaciones erróneas en cuanto a si un poste es defectuoso o no, es que se ha descartado su utilización momentáneamente, hasta que se cuente con un mejor *hardware* de captura de videos.

a Mejor imagen encontrada por el modelo para un poste “XX”.

b Extracción de poste “XX” en el mejor *frame*.

c Extracción de poste “XX” para otro *frame*.

Figura 4.4: Ejemplo de resultado en módulo de detección de eventos [Original de los autores, 2021].

Capítulo 5

Conclusiones

Lo realizado en este documento demuestra que es posible diseñar y aplicar tecnología emergentes en contextos nacionales, conociendo previamente que no es común la práctica de estas aplicaciones en el país. Una buena planificación permite obtener los insumos para desarrollar lo que se requiere; en este caso, los viajes para obtener registros de la infraestructura eléctrica chilena, puesto que si bien existen bases de datos de infraestructura eléctrica genéricas, estas varian considerablemente según la política de construcción de cada país (tanto de la infraestructura en sí, como de los elementos que lo rodean).

En cuanto al proyecto, lo realizado en ese trabajo de memoria, da grandes avances para pensar en una futura operación; a saber, las etapas de desarrollo restantes involucran solo posibles mejoras en los módulos y la creación de la plataforma *web* de visualización e interacción con los resultados, aspecto que se aleja de los objetivos de este trabajo de Título.

Por otro lado, en cuanto a los módulos desarrollados, estos presentan buenos resultados: *precision, recall* y F1 en su mayoria superiores a 0.9, tanto para la detección de los elementos de la clase "poste" como "catenaria". Sin embargo, como se vio en el documento, la clase "catenaria" presentó una métrica con un valor bajo a la hipótesis inicial: *precision* menor a 0.7. Pero dado que posibles modificaciones al módulo podrían alterar los resultados de los otras métricas, es que se decidió crear otro módulo que actuara como post-procesamiento de los resultados de detección de la clase "catenaria", resultando en una solución exitosa tanto por la mejoría en la detección de catenarias peligrosas, como también en el tiempo tomado para desarrollar tal módulo.

De todas formas, si eventualmente existiera tiempo en un futuro para mejorar el proyecto en lo que corresponde a las tareas de detección, no se descarta totalmente la idea de mejorar el valor de *precision* en la clase de "catenaria" sin incurrir en un modulo post-procesamiento. Para ello, una posible solución sería en aplicarla técnica de *data augmentation* [44] para solucionar el entrenamiento no balanceado de "poste" y "catenaria".

Finalmente, se espera con este documento mostrar la viabilidad de aplicar tecnologías de Inteligencia Artificial en el contexto Energético del país, debido a que se cuentan con los recursos para desarrollar diversas soluciones a distintas problemáticas o entregar un nuevo valor agregado a un proceso o acción similar.

Capítulo 6

Bibliografía

[1] *Memoria Anual Enel Distribución Chile 2016*. [Online]. Available: https://www.enel.cl/content/dam/enel-cl/inversionistas/enel-distribucion-chile/reportes/memorias/2016/Memoria-EnelDistribucionChile-2016.pdf

[2] "Gobierno pide catastro de usuarios electrodependientes para evitar muertes en catástrofes," *El Dínamo*, Jul 2017. [Online]. Available: https://www.eldinamo.cl/nacional/2017/07/20/gobierno-pide-catastro-de-usuarios-electrodependientes-para-evitar-muertes-en-catastrofes/

[3] *Memoria Anual Enel Distribución Chile 2017*. [Online]. Available: https://www.enel.cl/content/dam/enel-cl/inversionistas/enel-distribucion-chile/reportes/memorias/2017/Memoria-EnelDistribucionChile-2017.pdf

[4] "Corte suprema confirma multa de $856 millones de la SEC contra Enel Distribución por baja calidad de suministro," *Electricidad*, Nov 2018. [Online]. Available: http://www.revistaei.cl/2018/11/23/corte-suprema-confirma-multa-de-856-millones-de-la-sec-contra-enel-distribucion-por-baja-calidad-de-suministro/

[5] Agenda Digital 2020, "*Smart Cities*: Pilotos de Ciudades Inteligentes para Chile." [Online]. Available: http://www.agendadigital.gob.cl/#/seguimiento/medida/Smart-Cities:-Pilotos-de-Ciudades-Inteligentes-para-Chile

[6] Facultad de Ciencias Físicas y Matemáticas, Universidad de Chile, "Departamento de Ingeniería Eléctrica," accessed: 29 December, 2019. [Online]. Available: http://ingenieria.uchile.cl/departamentos/87703/ingenieria-electrica

[7] R. C. Gonzalez and R. E. Woods, *Digital image processing*. Prentice-Hall, 2002.

[8] I. Mohamed, "Detection and tracking of pallets using a laser rangefinder and machine learning techniques," Ph.D. dissertation, 09 2017.

[9] K. Ciążyński and A. Fabijańska, "Detection of qr-codes in digital images based on histogram similarity," *Image Processing Communications*, vol. 20, no. 2, p. 41–48, 2015.

[10] R. an Image from Projection Data MATLAB Simulink Example, "Rgb," https://www.mathworks.com/help/matlab/ref/rgb2gray.html, accessed: 7 December, 2019.

[11] opencv dev team, "Miscellaneous image transformations," https://docs.opencv.org/2.4/modules/imgproc/doc/miscellaneous_transformations.html, accessed: 7 December, 2019.

[12] M. A. Khuhro, D.-J. Huang, S.-N. Huang, and L.-L. Yang, "Improved gaussian filtering model for highly noised image by salt and pepper," *DEStech Transactions on Computer Science and Engineering*, no. cst, 2017.

[13] R. Klette, "Concise computer vision," *Undergraduate Topics in Computer Science*, 2014.

[14] Intel, "Intel hyperflex architecture high-performance design handbook," https://www.intel.com/content/www/us/en/programmable/documentation/jbr1444752564689.html#jbr1446659725070.

[15] R. Raju, T. Maul, and A. Bargiela, "New image processing pipelines for membrane detection," *Journal of the Institute of Industrial Applications Engineers*, vol. 3, no. 1, p. 15–23, 2015.

[16] M. Morales, L. Barrientos-Lozano, S. Del Amo-Rodríguez, J. Horta-Vega, and C. Venegas-Barrera, "Plantas prioritarias para conservación y manejo sustentable en alta cimas (reserva de la biósfera el cielo), tamaulipas, méxico," *TecnoIntelecto ISSN 1665-983X*, vol. 10, pp. 5–19, 10 2013.

[17] R. Stone, "Image Segmentation Using Color Spaces in OpenCV + Python," accessed: 12 April, 2021. [Online]. Available: https://realpython.com/python-opencv-color-spaces/

[18] Anon., "Cascade classification - opencv 2.4.13.7 documentation," https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_fast/py_fast.html, 2017, accessed: 9 December, 2019.

[19] "Thresholding for mobile ocr: An introduction – part 2," accessed: 9 December, 2019. [Online]. Available: https://anyline.com/news/thresholding-otsu-adaptive-multilevel/

[20] Anon, "Open cv - canny edge detector," https://webnautes.tistory.com/687, 2018, accessed: 7 December, 2019.

[21] J. Huang, G. Zhou, X. Zhou, and R. Zhang, "A new fpga architecture of fast and brief algorithm for on-board corner detection and matching," *Sensors*, vol. 18, no. 4, 2018. [Online]. Available: https://www.mdpi.com/1424-8220/18/4/1014

[22] M. S. Nixon and A. S. Aguado, "Chapter 5 - high-level feature extraction: fixed shape matching," in *Feature Extraction Image Processing for Computer Vision (Third Edition)*, third edition ed., M. S. Nixon and A. S. Aguado, Eds. Oxford: Academic Press, 2012, pp. 217 – 291. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9780123965493000057

[23] S. Raschka and V. Mirjalili, *Python Machine Learning - Second Edition*, 2nd ed. Packt Publishing., 2017.

[24] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, p. 1798–1828, 2013.

[25] S. J. Russell and P. Norving, *Artificial Intelligence: a modern approach.* Prentice-Hall, 2010.

[26] A. Kamilaris and F. Prenafeta Boldú, "A review of the use of convolutional neural networks in agriculture," *The Journal of Agricultural Science*, vol. 156, pp. 1–11, 06 2018.

[27] P. Joshi, *Artificial intelligence with Python: build real-world artificial intelligence applications with Python to intelligently interact with the world around you.* Packt Publishing Ltd., 2017.

[28] Z.-Q. Zhao, P. Zheng, S.-t. Xu, and X. Wu, "Object detection with deep learning: A review," 2018. [Online]. Available: http://arxiv.org/abs/1807.05511

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1904–1916, 2015.

[30] R. Girshick, "Fast r-cnn," *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015.

[31] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, p. 1137–1149, 2017.

[32] J. Dai, Y. Li, K. He, and J. Sun, "R-fcn: Object detection via region-based fully convolutional networks," in *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Curran Associates, Inc., 2016, pp. 379–387. [Online]. Available: http://papers.nips.cc/paper/6465-r-fcn-object-detection-via-region-based-fully-convolutional-networks.pdf

[33] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, no. 2, 2017.

[34] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask r-cnn," *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017.

[35] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, p. 779–788, 2016.

[36] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," *Computer Vision – ECCV 2016 Lecture Notes in Computer Science*, p. 21–37, Oct 2016.

[37] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[38] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation," *CoRR*, vol. abs/1801.04381, 2018. [Online]. Available: http://arxiv.org/abs/1801.04381

[39] K. M. Ting, *Confusion Matrix*. Boston, MA: Springer US, 2017, pp. 260–260. [Online]. Available: https://doi.org/10.1007/978-1-4899-7687-1_50

[40] P. Lifeng, "Intelligent image recognition research on status of power transmission lines," *Sensors Transducers*, vol. vol. 179, no. 9, pp. 174-179, 2014.

[41] Z. You, Kailiu, W. Mao, Q. Wu, and Y. Guo, "An intelligent monitoring system based on internet of things for electric distribution network," in *2018 China International*

Conference on Electricity Distribution (CICED), Sep. 2018, pp. 1256–1262.

[42] A. Molaei, H. D. Taghirad, and J. Dargahi, "Extracting of sagging profile of overhead power transmission line via image processing," in *2018 IEEE Canadian Conference on Electrical Computer Engineering (CCECE)*, May 2018, pp. 1–5.

[43] L. Baker, S. Mills, T. Langlotz, and C. Rathbone, "Power line detection using hough transform and line tracing techniques," 11 2016, pp. 1–6.

[44] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, Jul. 2019. [Online]. Available: https://doi.org/10.1186/s40537-019-0197-0

Anexo A

Detalle cronológico de los recorridos

El proyecto de este documento se inició en agosto del 2019. Luego de dos meses, se realizaron los primeros viajes para obtener la primera base de datos de la infraestructura de la red eléctrica. En ese periodo, la cámara utilizada fue la cámara IP *full* HD Foscam fi9928P. Los primeros recorridos fueron realizados en la Región Metropolitana, con énfasis en zonas industriales por dos motivos principales: presencia de mayores fallas en la infraestructura eléctrica y menor "ruido" externo.

El primer motivo se justifica en que en esas zonas hay bastante movilización de grandes vehículos con diversas cargas, los que están más propensos a chocar con postes eléctricos y/o pasar a llevar cables. En tanto, el segundo motivo se refiere a que la grabación de videos de la infraestructura eléctrica tendrán menor presencia de elementos externos que puedan dificultar el desarrollo inicial de los módulos de detección, como árboles, letreros publicitarios, edificios residenciales, entre otros.

La segunda etapa de recorridos se realizaron durante febrero del 2020, con el fin de obtener esta vez un registro más diverso, al integrar zonas úrbanas no necesarimentes industriales, aumentando el número de comunas de la Región Metropolitana.

La tercera etapa de recorridos se iba a realizar en abril del 2020 y con una cámara IP *stereo*. Sin embargo, debido a la pandemia del Covid-19 y a las distintas medidas sanitarias que se impusieron en el país, se canceló esa tercera etapa de manera indefinida.

Finalmente, la tercera etapa se reagendó para los meses de noviembre u diciembre del 2020. En ella, se agregaron más comunas de la Región Metropolitana en la ruta de recorridos, además de ser, hasta la fecha, la etapa más extensa en la obtención de registros de videos. Si bien se tenía pensado utilizar una cámara IP *stereo*, por asuntos logísticos no pudo ser importada antes de la realización de esta etapa. No obstante, se utilizó otra cámara en comparación a las etapas anteriores: esta fue la cámara IP Foscam G4P.

Anexo B

Primeros módulos diseñados

En esta sección de anexo se presenta la primera iteración de módulos diseñadas, en donde se destaca principalmente la inclusión del evento de la clase "cable colgando", adicional a los principales del proyecto: "poste" y "catenaria".

B.1. Módulo de detección de eventos

Con el conjunto de imágenes ya etiquetado se diseñó el primer módulo computacional que fuera capaz de detectar las distintas clases de eventos de interés al entregarle un video como archivo de entrada. Lo complejo del módulo recae en la detección de eventos, el cual se logra mediante el desarrollo, adaptación, entrenamiento y validación de redes neuronales. Para lograrlo, se utilizó la API Object Detection de la librería de Tensorflow para el lenguaje de programación Python, la cual se debió adaptar al contexto de la problemática y de las características que presenta el conjunto de imágenes. Luego, para ajustar los pesos de la red neuronal, se realizaron varias iteraciones en el entrenamiento, hasta lograr los resultados esperados, que fueron validados con un conjunto de validación dinámico. Esto último con el fin de evitar el *overfitting*.

Algunos resultados que entregó la validación, luego de realizar 10000 iteraciones en el entrenamiento, son los siguientes:

- **Precisión de los *bounding box*: 84 %**. Esto es para los objetos de las distintas clases que sí logra detectar, pero que no los encierra a la perfección en comparación al etiquetamiento manual que se mencionó en la sección anterior. El valor porcentual hace referencia al área que sí se logró encerrar en un *bounding box*, donde se tiene que las mayores áreas que no lograron encerrarse totalmente fueron a los objetos de la clase "poste".
- **Porcentaje de pérdida: 12 %**. Quiere decir que, prácticamente, 7 de cada 8 eventos son detectados, independiente de la precisión. Detalladamente, esto ocurre principalmente con la clase "cable colgando", debido a que el área que representa tal fenómeno es muy pequeño, en comparación a los eventos "catenaria" y "poste".
- **Clasificación errónea de un *bounding box*: 4 %**. Esto quiere decir que 1 de cada 20 detecciones que realiza el modelo, falla en la clase con la que etiqueta a ese evento.

Principalmente esto ocurre cuando el modelo detecta el área de un cable colgando y lo etiqueta como "catenaria". También se da el caso contrario, pero en menor escala. No así con los postes, los cuales nunca fueron confundidos en la clasificación de su clase.

Teniendo los pesos ajustados de la red neuronal que realizó la detección, se diseñó un *script* que al recibir un video fuera capaz de detectar y encerrar en un *bounding box* cada clase de evento, junto a su *confidence score* o porcentaje de confianza en la detección. En la Figura 4.1 se aprecian algunos *frame* del video de salida que genera el modelo.

En términos numéricos, los resultados del modelo computacional de detección aplicada a un video de prueba de corta duración se muestran en los Cuadro 1, 2 y 3.

Tabla B.1: Matriz de confusión de la clase "poste"

		Resultado de la detección		
		p	**n**	**total**
Valor real	**p′**	87	0	P′
	n′	13	0	N′
	total	P	N	

Tabla B.2: Matriz de confusión de la clase "catenaria"

		Resultado de la detección		
		p	**n**	**total**
Valor real	**p′**	58	17	P′
	n′	25	0	N′
	total	P	N	

Tabla B.3: Matriz de confusión de la clase “cable colgando”

		Resultado de la detección		
		p	**n**	**total**
Valor real	**p′**	47	12	P′
	n′	41	0	N′
	total	P	N	

Los resultados desagregados para cada clase de evento en el modelo computacional de detección aplicada a un video de prueba de corta duración son los siguientes:

- **Clase “poste”**. Todos los postes logran ser detectados por el módulo. Sin embargo, hay falsos positivos, ya que ciertos postes que no son parte del tendido eléctrico (como por ejemplo postes de luz), son detectados y etiquetados erróneamente por el módulo como un objeto de la clase “poste”.
- **Clase “catenaria”**. Las catenarias que ocurren en un tramo corto del tendido eléctrico logran ser detectados y etiquetados correctamente por el módulo, pero las que ocurren a lo largo del tendido eléctrico (donde el punto más pronunciado de su curva ocurre a la mitad del tramo del cable entre dos postes) no logran ser detectados por el módulo.
- **Clase “cable colgando”**. Los cables que se encuentran colgando en medio del tendido eléctrico no logran ser detectados, en su mayoría, por el módulo de detección. Los que sí se logran detectar, además, no son clasificados siempre de forma correcta, ya que el mmódulo suele etiquetarlo como un evento de la clase “catenaria”.

Si bien el resultado en la detección de los objetos de la clase “poste” es lo que se busca -salvo en algunos casos, como por ejemplo en la Figura B.1 (c) en el que se encuentra duplicado la detección para un mismo objeto de esta clase-, no se obtiene lo mismo con las otras dos clases por distintos motivos, de los cuales se destacan: estos eventos no ocurren con una alta frecuencia en un video a diferencia de la aparición de postes, lo que puede generar una omisión en su detección (Figura B.1 (d)); y son fácilmente confundibles entre ellos (“catenaria” y “cable colgando”, como ocurre en la Figura B.1 (c)), debido a que ambos ocurren en el tendido eléctrico.

Como la hipótesis que se manejaba era que ocurriera lo anterior, es que previamente se tenía considerado el desarrollo de un segundo módulo para mejorar los resultados en estos dos eventos y lograr lo esperado, tal como sí se logra en la Figura B.1(a) y B.1(b).

a Se detectan correctamente los objetos.

b Se detectan correctamente los objetos.

c No se detectan correctamente los objetos.

d Se omite la detección de algunos objetos.

Figura B.1: Algunos *frame* del video de salida en el módulo de detección de eventos [Original de los autores, 2019].

B.2. Módulo de detección de catenarias

Con los vértices del área del cableado eléctrico, este módulo busca clasificar las catenarias presentes y determinar cuál(es) representa(n) un peligro para la comunidad. Para ello, se realizan tres pasos: el primero consiste en considerar el *frame* en donde fue detectada la catenaria en el módulo anterior y realizar una transformación de perspectiva en el área de los cables (utilizando la posición del *bounding box* de la catenaria), que en un principio está en tres dimensiones, a dos dimensiones. Luego, a esa imagen ya transformada se le aplican los algoritmos de detección de catenarias y, finalmente, a lás áreas detectadas se les determina si su ángulo de curvatura y distancia aproximada con respecto al suelo superan el umbral crítico para ser catalogadas como catenarias peligrosas. En la Figura B.2 se aprecia el primer paso.

a *Frame* original (3D) b Transformación 2D

Figura B.2: Transformación a 2D[Original de los autores, 2019].

La estimación de altura se realizan mediante procesamiento de imágenes, utilizando como referencia el poste cercano a la catenaria, suponiendo una altura 9.5 [m], y utilizando la posición de los pixeles de la parte inferior de la catenaria y del *bounding box* del poste. Sin embargo, las alturas estimadas para las catenarias no logran ser precisas, lo que produce erróneas clasificaciones en si las catenarias son peligrosas o no.