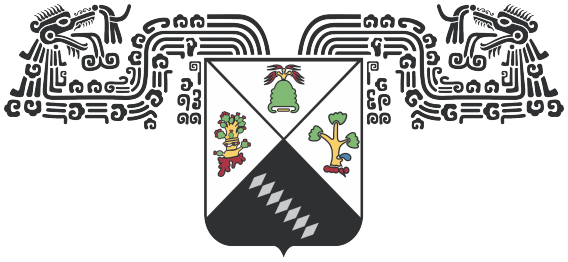


UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MORELOS

UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MORELOS
INSTITUTO DE INVESTIGACIÓN EN CIENCIAS BÁSICAS Y APLICADAS
CENTRO DE INVESTIGACIÓN EN CIENCIAS

"Detección automática de torres eléctricas y cadenas de aisladores en imágenes aéreas"

TESIS

QUE PARA OBTENER EL GRADO DE

LICENCIADO EN CIENCIAS

PRESENTA

ANDRÉS ALBERTO REYES RAMÍREZ

DIRECTOR DE TESIS:
Dr. Jorge Alberto Fuentes Pacheco

SINODALES:
Dr. Jorge Hermosillo Valadez (Secretario)
Dra. Lorena Díaz Gonzales (Suplente)
Dr. Juan Manuel Rendón Mancha (Presidente)
Dr. Audberto Reyes Rosas (Suplente)

Cuernavaca, Morelos *Junio, 2022*

Vaya, este escrito es el resultado de un chingo de cosas. Malas experiencias, buenos momentos, amor, cariño, risas, traumas, problemas, aciertos, personas que me formaron y marcaron de por vida, enseñanzas, profesores buenos y malos, aprendizaje y confusión, y mil emociones nuevas, únicas, y sentimientos que no se pueden plasmar en palabras ni formas. Este escrito, esta tesis, esta carrera, y las consecuencias de haber cursado y concretado esta licenciatura, fue posible gracias a muchas personas que tienen un valor inmesurable en mi vida, y quiero dedicarles esto y agradecerles por todo su apoyo y amor.

Es difícil expresar tanto afecto a tantas personas con tantos momentos y recuerdos, solo puedo agradecer cada día de mi vida por haberme cruzado con ellos y que me permitieran compartir mi vida con ellos, y ellos conmigo. Gracias totales, diría el Cerati.

A mi madre. Sin las enseñanzas, el apoyo, las experiencias, la motivación, y sobre todas las cosas el amor que me diste, no estaría aquí, ni estas palabras estarían aquí, ni yo sería la persona que soy hoy en día. Eres una mujer increíble mamá, me impresiona la fuerza y la perseverancia con la que existes, y siempre te voy admirar por todo lo que has logrado y hecho y formado en ti, y el reflejo de toda esa belleza en mi. Gracias Má, te amo infinitamente hoy y siempre.

A mi tía Lety, por siempre apoyarme y creer en mi, darme tu amor y enseñarme que apoyar a los demás es de las cosas más puras y más enriquecedoras que hay en la vida, sin duda alguna eres un ángel y un ser humano excepcional y espero en algún momento llegar a tener la pureza que tienes en el corazón para ayudar a los que me rodean como tú lo haces. Te amo con toda mi alma, tía.

A Borksh. Sin duda alguna bro, esto no hubiera sido posible sin ti. El apoyo y el cariño que me has dado, la manera de alentarme apoyarme e incluso resolverme los pedos, solo hacen que me sienta con un motivo más para agradecer estar vivo y respirando y poder compartir la existencia contigo. Te amo Koksh.

A Jaime por creer en mi cuando yo no lo hacía y siempre estar ahí como un amigo incondicional, no podría expresar la cantidad de cosas que te agradezco en este texto, pero ps, luego nos echamos unos tacos con mucha sal y te platico a detalle como pusiste un chingo para que esto fuera posible. Te amo Yeimz.

A mi papá, Guchi. Me acuerdo bien de muchas veces que platiqué contigo en la carrera y me hacías sentir que no pasaba nada si reprobaba 3 materias en primer semestre. Me acuerdo también de las veces que me llevabas a la uni aunque el carro no anduviera al 100, o cuando no había dinero y de alguna manera llegaba a la uni con dinero para comer. Te amo papá, es difícil expresar lo que siento contigo, pero sin duda alguna te amo con toda mi alma. Gracias.

A Cris. Auretik, orain, etta betti. Maite zaitut. Gracias Picio.

A Rodo, a Sam, a Piñon, al Chefo, al Faker, a Fernanda, al Eibrajam, a Valeria, a Paola, a Lalo, a Holey, a Gus, a Hermosillo, a Bruno, a Lorena, a Fountain, a Rengar, a Chiran, a Paco, a Jules, a Mauricio, a Surendra, a Tooga, a Alexa.

Y a mi, gracias por no rendirte aunque la hayas pasado mal, sin duda tuviste muchas oportunidades para decidir alejarte de esto y tirar la toalla. Pero te rifaste y eso es lo que importa, que eres capaz aunque no te la creas. Muchas gracias Rem.

Es difícil escribir agradecimientos, por cierto. Si olvidé a alguien, luego me reclaman please.

Índice general

Índice de figuras

Capítulo 1

Introducción

Gran parte del avance tecnológico se debe a la cantidad de herramientas y utilidades desarrolladas por la humanidad, desde lápices para plasmar ideas y conceptos, hasta un sistema automatizado que puede vencer al campeón mundial de Go.

Muchas de estas herramientas, fueron diseñadas con el propósito de brindar apoyo en las actividades que los seres humanos no pueden realizar, o que son demasiado complicadas y pesadas para realizarse por mano propia.

Una de estas actividades, es la inspección visual de estaciones eléctricas.

1.1. Planteamiento del problema

Las líneas de transmisión de electricidad se han visto afectadas por la demanda, que ha creado una necesidad por aumentar su cantidad, y esto ha incrementado el mantenimiento que necesitan para conservarse en óptimas condiciones.

Actualmente, las inspecciones que se realizan en la CFE (Comisión Federal de Electricidad) se realizan por personas y de forma esporádica debido al alto costo que representa y el tiempo que se toma en llevar a cabo una inspección completa en una subestación eléctrica. Esto origina altos gastos de mantenimiento y cortes de energía recurrentes.

En este trabajo, se pretende implementar un sistema automatizado de inspección visual a través del uso de drones y visión por computadora. Para lograr esta meta, es necesario localizar los objetos de interés, como son torres eléctricas y cadenas de aisladores. Estos objetos juegan un papel de vital importancia en las torres eléctricas, y en su mayoría son los que reciben la mayor cantidad de daño o deterioro a lo largo de su vida útil. Para alcanzar este objetivo, se hará uso de técnicas de aprendizaje de máquina y visión por computadora: una red neuronal convolucional (YOLO (del inglés *You Only Look Once*) [1]) y teoría de detección de objetos.

1.2. Hipótesis

Es posible entrenar a una red neuronal tipo YOLO con imágenes sintéticas de torres eléctricas y cadenas de aisladores, y lograr obtener un buen desempeño en la detección de estos objetos en imágenes capturadas en entornos reales.

1.3. Justificación

Implementar un sistema de supervisión automática proporciona ventajas para la inspección de las líneas de transmisión de energía, desde una forma más rápida y eficiente de localizar posibles defectos, hasta una optimización realmente grande en los costos que las rutinas de inspección pueden generar. Un sistema automático no solo reduce costos y agiliza el proceso, también reduce riesgos que pueden causar otros métodos de inspección como lo es el método manual.

Para lograr implementar un sistema de inspección completamente automático, es necesario poder identificar los objetos más relevantes para el funcionamiento correcto de la estación. Al lograr identificar y aislar los objetos de interés de la imagen, se acelera el procesamiento y análisis de estos, permitiendo una mejor toma de decisiones para optimizar el funcionamiento de estos.

Una vez que los objetos de interés han sido correctamente identificados, se puede pensar en realizar una segmentación, y después de esto, la inspección se reduce meramente a un problema de clasificación para saber cuáles objetos podrían o no tener problemas, y con base en estos resultados tomar las medidas pertinentes para mantener las líneas de transmisión en óptimas condiciones.

1.4. Objetivo General

Analizar e implementar una red neuronal convolucional que sea capaz de detectar cadenas de aisladores y torres eléctricas en imágenes aéreas usando para su entrenamiento imágenes sintéticas.

1.5. Objetivos Específicos

- Estudiar e implementar una red con una arquitectura tipo YOLO con el fin de detectar los componentes de interés.
- Poner en funcionamiento un simulador de entornos y un simulador de cuadricópteros para generar un ecosistema símil al que se encontraría un dron en entornos reales.
- Crear una base de datos de imágenes aéreas sintéticas obtenidas de un simulador de entornos.

1.6. Alcances y Limitaciones

A continuación, se definirán los alcances y limitaciones del proyecto:

1.6.1. Alcances

- Los objetos de una misma clase tienen variaciones entre ellos, es decir, son considerados cadenas de aisladores eléctricos de diferentes tipos y torres eléctricas de distintos modelos.
- El trabajo está enfocado particularmente en la detección de los objetos a través de una caja contenedora marcada en la imagen.
- Parte de los objetos de interés pueden verse parcialmente por otros elementos de la estación que se encuentran en primer plano.

1.6.2. Limitaciones

- El entrenamiento de la red es realizado con datos sintéticos, debido a la poca cantidad de imágenes en la red de los objetos de interés.
- Solamente se trabaja con dos clases de objetos: las cadenas de aisladores y las torres eléctricas.
- Los procesos de entrenamiento e inferencia de la red neuronal son ejecutados fuera de línea.

Este proyecto es de gran apoyo para la tarea de inspección de subestaciones eléctricas. Al lograr que una red neuronal tenga un buen desempeño en la predicción al usar imágenes sintéticas en su entrenamiento, se logran expandir los horizontes en el área de la detección de objetos, ya que se podrán utilizar objetos representados de manera artificial para el entrenamiento de la red.

Capítulo 2

Marco Teórico

Un compendio de los conceptos y técnicas que tienen repercusión directamente en el diseño del modelo.

2.1. Vehículos Aéreos No Tripulados (VANT)

Los VANT han tenido un incremento en sus usos en la última década, diversas aplicaciones han surgido para estos vehículos, algunos son utilizados como máquinas de guerra, agricultura de precisión y mensajería. Cabe destacar aquí el contraste en sus aplicaciones, y es muy recomendable tomarse un momento para reflexionar al respecto.

2.2. Visión Artificial

Según la AIA (del inglés, *Automated Imaging Association*), la visión artificial abarca todas las aplicaciones industriales y no industriales en las que una combinación de *hardware* y *software* brinda un guiado operativo a los dispositivos en la ejecución de sus funciones de acuerdo con la captación y procesamiento de imágenes [2].

Esta disciplina proporciona un panorama sumamente amplio para el procesamiento de imágenes, el sistema de detección de torres eléctricas y cadenas de aisladores de este proyecto, está completamente basado en esta disciplina y más concretamente en la rama de **detección de objetos.**

2.2.1. Detección de objetos

La detección de objetos suele confundirse frecuentemente con otras técnicas de visión por computadora que involucran identificar objetos, como lo es la segmentación de objetos y la clasificación de imágenes, ver figura 2.1.

La clasificación de imágenes tiene como meta dar una etiqueta de clase, para cada imagen introducida, es decir, a qué conjunto de objetos pertenece el objeto de la imagen.

La segmentación de objetos, por otro lado, es un proceso de clasificación por pixel que asigna una categoría a cada pixel de la imagen procesada, permitiendo obtener la región que contiene a el objeto de interés a nivel pixel.

La detección de objetos realiza la tarea de la localización y de la clasificación, pero en el caso de la clasificación la realiza a nivel objeto. Determina la ubicación de los objetos en la imagen, y le asigna una etiqueta de clase a la que pertenece cada objeto. Por ejemplo: Si procesamos una fotografía de una casa, un árbol y un buzón de correo, la salida deberá ser tres cuadros delimitadores, cada uno con su respectiva etiqueta de clase, que podrían ser 'construcciones', 'árboles' y 'buzones'; respectivamente.

Siendo el problema por atacar un desafío de localización y de clasificación, se optó por la técnica de detección de objetos como la más adecuada para la tarea.

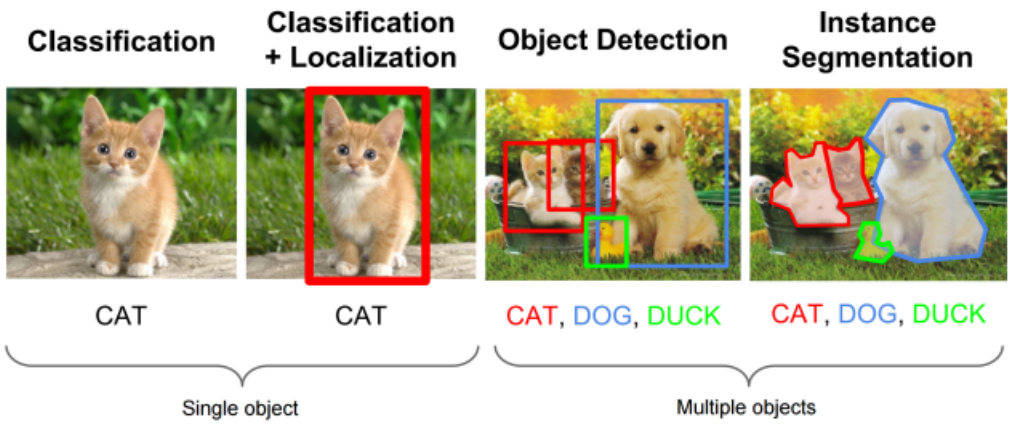


Figura 2.1: Diferencias entre técnicas de localización de objetos. Imagen extraída de [3]

2.3. Aprendizaje de Máquina

El aprendizaje de máquina es un proceso de inducción del conocimiento con origen en la Inteligencia Artificial cuyo objetivo es crear sistemas capaces de aprender, o mejorar su rendimiento actual, en función de los datos que consumen. Es una técnica sumamente utilizada en la actualidad.

2.3.1. Aprendizaje supervisado

El aprendizaje supervisado, consiste en brindarle información al agente sobre las salidas esperadas con respecto a las entradas posibles, de forma que al recibir los datos de las imágenes el sistema sea capaz de determinar la salida. Al igual que los niños cuando aprenden a identificar formas y

figuras por medio de imágenes, el aprendizaje supervisado le enseña a la máquina a imitar este comportamiento.

Es el tipo de aprendizaje automático más utilizado, ejemplos de este tipo de algoritmos, son los árboles de decisión y las SVM, del inglés (*Support Vector Machines*.

2.3.2. Aprendizaje no supervisado

El aprendizaje no supervisado, por otra parte, es independiente, la computadora aprende a identificar procesos y patrones sin la necesidad de que un ser humano esté monitoreando todo el proceso de aprendizaje. Al igual que con el ejemplo anterior, este tipo de algoritmos buscan similitudes entre las imágenes y las separa en clases, asignando una etiqueta a cada una de ellas.

Ejemplos de este, son los algoritmos de clusterización.

2.4. Redes Neuronales Artificiales

El trabajo en redes neuronales artificiales tuvo el origen de su motivación en el reconocer que el cerebro humano procesa la información de una forma completamente diferente en comparación con una computadora convencional.

2.4.1. Conceptos básicos

Su diseño e implementación fue inspirado en una analogía del cerebro humano y su funcionamiento, aunque esto es algo complicado de afirmar, ya que realmente se desconoce el verdadero procesamiento ocurrido dentro de las neuronas cerebrales convencionales.

Neurona (Perceptrón)

Las neuronas son el principal componente de las redes neuronales, en ellas sucede el procesamiento de las variables de entrada para generar un modelo de regresión lineal, ver figura 2.2.

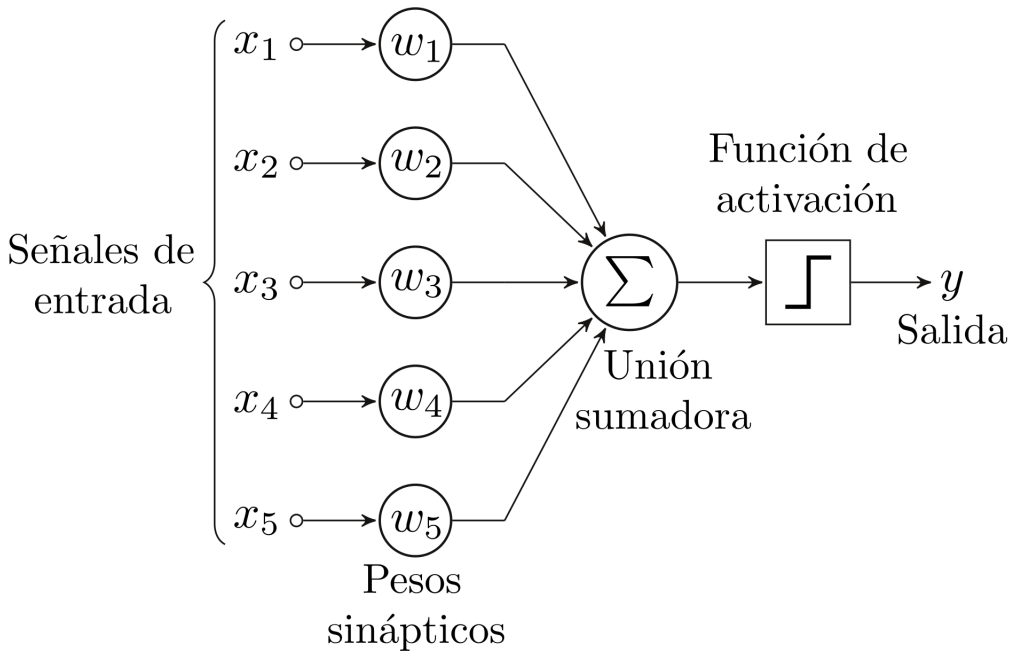


Figura 2.2: Perceptrón de cinco unidades. Imagen extraída de [4]

Capa

Hay tres tipos de capas en una red neuronal artificial:

- Capa de entrada:

  Esta capa determina la entrada de datos que reciben las neuronas de la primera capa oculta, esto inspirado en la sinapsis que realizan las neuronas humanas, en este caso, el dato proporcionado se vería como un neurotransmisor.

- Capa oculta:

  La capa oculta puede o no estar constituida de varias 'capas', no deben confundirse con la de entrada y salida, estas capas son parte de la capa oculta y contienen las neuronas que hacen el procesamiento principal de la red.

- Capa de salida:

  Puede tener una o más salidas, dependiendo el modelo de la red neuronal y la cantidad de características que se deseen encontrar en los datos dados.

Función de activación

La función de activación es parte de las neuronas, recibe de entrada un valor de la forma:

$$W * x + b \tag{2.1}$$

donde W son los pesos de la neurona y x son las entradas, y la b representa el sesgo($BIAS$) agregado a este valor. Este valor no tiene ningún tipo de restricción, y cuando se utilizan redes neuronales profundas puede llegar a una magnitud muy alta por la cantidad de parámetros.

La principal funcionalidad de la función de activación es acotar los valores de salida que generan las neuronas, de tal forma que se puedan mapear entre valores más representativos.

2.4.2. Parámetros

Función de pérdida

La función de pérdida cuantifica el error entre los valores de las predicciones y los valores esperados, realizando una diferencia y representándolo como un número real. Ésta se puede utilizar para maximizar o minimizar el valor esperado. Si se minimiza usualmente a este valor se le llama 'pérdida', 'error' o 'coste'.

Pesos

Los pesos de la red neuronal son los valores que reciben las conexiones entre neuronas. El objetivo del entrenamiento de una red neuronal es actualizar estos pesos para decrementar la función de error.

BIAS

El *BIAS* es una entrada extra para las neuronas, y tiene su propio peso en la conexión a la neurona. Este término controla la activación de la neurona, es decir, un sesgo alto provoca que las neuronas requieran una entrada más alta para generar una salida de 1. Un sesgo bajo requiere una entrada menor para generar una salida de 1.

2.4.3. Backpropagation

El algoritmo de *backpropagation* [5] es un algoritmo que consiste en minimizar la función de costo ajustando los pesos de la red y el *BIAS*. Este algoritmo se puede ver como un conjunto de técnicas que se encargan de garantizar que el entrenamiento sea correcto.

Primero se calculan los valores de los pesos de la red haciendo un *forward pass*, obteniendo los valores de salida y todos los pesos de la red. Una vez hecho esto, se obtiene el error total que es la diferencia entre las salidas obtenidas, y las salidas esperadas.

$$E_{total} = \sum \frac{1}{2}(salida\ esperada - salida\ obtenida)^2 \tag{2.2}$$

Ahora, con el error obtenido, se actualizan los pesos en la fase de *backward pass*. Para actualizar los pesos, se calcula el error correspondiente a cada peso con ayuda del error total. El error en el peso w_i se calcula realizando la derivada parcial del error total, con respecto a w.

$$Error_w = \frac{\partial E_{total}}{\partial w} \tag{2.3}$$

2.4.4. Gradiente Descendente

Es considerado una base del aprendizaje en muchos métodos y técnicas de *machine learning*. Ha sido sumamente implementado en *deep learning* para entrenar redes neuronales, también es necesario para la regresión logística. La base del gradiente descendente es ajustar los parámetros de una función de forma iterativa para minimizarla.

Este algoritmo requiere como parámetros de entrada: la **tasa de aprendizaje(α)** y el número de iteraciones. La tasa de aprendizaje determina la cantidad de iteraciones para que el algoritmo converja.

Se tiene una función diferenciable $f : \Omega \subset \mathbb{R}^n \to \mathbb{R}$, este algoritmo permite encontrar un w en Ω tal que $f(w)$ es un mínimo. El conjunto que determina este algoritmo se define de esta forma:

$$w \in argmin f(w_t) \mid w \in \Omega \tag{2.4}$$

donde $argmin f(x) \mid x \in X = x \in X : f(x) = y_{min}$
Para determinar los valores de w que forman parte del conjunto 2.4, el algoritmo itera y calcula un valor diferente para cada w basándose en un error total y el error de cada w.

2.4.5. Gradiente Descendente Estocástico

El gradiente descendente estocástico es una variante del gradiente descendente que toma una muestra aleatoria del conjunto de datos que van a ser utilizados para calcular el gradiente por iteración. Al utilizarse un solo dato del conjunto para calcular el gradiente, actualizar los pesos y calcular el valor de la función de pérdida, el proceso se vuelve mucho más rápido [6].

Al calcular un dato por iteración, eventualmente debe recorrer todo el conjunto de datos. Cuando esto ocurre, recibe el nombre de **época**.

Funcionamiento

Primero se inicializa el vector, y durante las épocas, se toman puntos de datos del conjunto de datos de manera aleatoria. Después se utiliza un elemento de entrenamiento y su peso, para calcular el gradiente. Se actualiza el vector de pesos, y se regresa el valor actualizado de estos.

Un problema que tiene este algoritmo es que al ser estocástico puede realizar los cambios en los pesos de forma abrupta, y tardar demasiado en converger a un mínimo. Para este problema se propuso el uso de un parámetro llamado *momentum*, ver figura

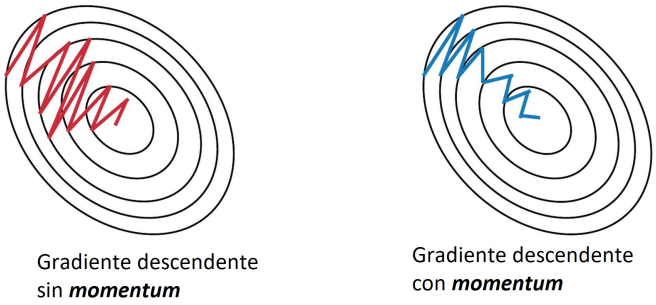


Figura 2.3: Gradiente descendente estocástico son y sin *momentum*. Imagen extraída de [6]

2.4.6. Redes Neuronales Convolucionales

Las redes neuronales convolucionales cuentan con capas convolucionales, estas consisten en múltiples filtros de una o más dimensiones. Se utilizan para extraer características de las imágenes, y después del procesamiento de estas capas, suelen utilizarse capas para realizar la clasificación.

Núcleos

Los núcleos(*kernel*) son matrices cuadradas de $n \times n$ dimensiones, con n menor a las dimensiones de la imagen de entrada. El *kernel* no es más que un filtro que se utiliza para extraer características de la imagen. Se desplaza sobre los datos de entrada, realiza un producto punto con una subregión de la imagen, y genera un mapa de características por medio de un proceso de **convolución**, ver figura 2.4.

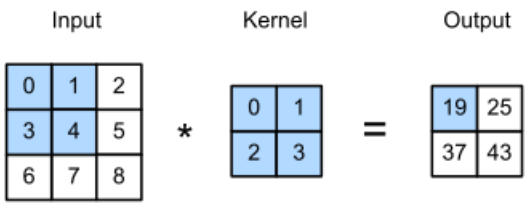


Figura 2.4: Representación de la función de un *kernel*.

Convolución

La convolución es el proceso que realiza el *kernel* sobre las regiones de la imagen. Se realiza un producto punto con la región sobre la que está el *kernel*, y el resultado de este producto punto es asignado a la primera coordenada del mapa de características. El *stride* define el cambio de pixeles a la hora del desplazamiento del ***kernel***. Si se tiene un *stride* de 1, el *kernel* se va a desplazar de izquierda a derecha, de arriba abajo, pixel por pixel. Si el *stride* es de 2, se realiza el mismo proceso, pero el desplazamiento se realiza de 2 en 2 pixeles, ver figura 2.5.

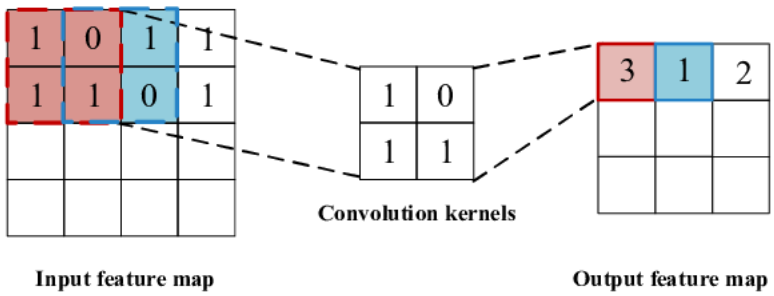


Figura 2.5: Diagrama general de la convolución. Imagen extraída de [7]

Capítulo 3

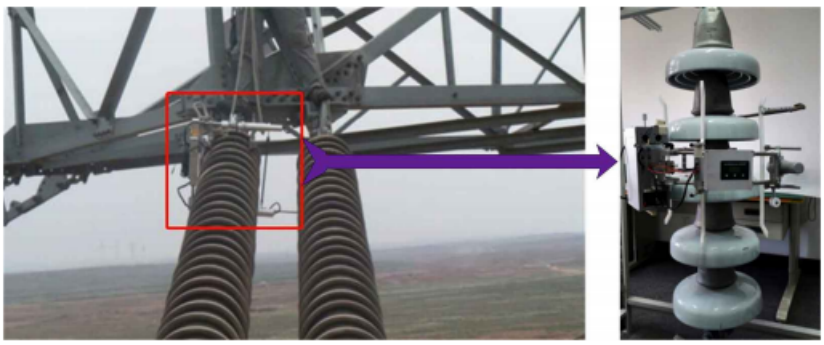
Estado del Arte

3.1. Técnicas de localización de objetos

En este capítulo se presentan algunas de las implementaciones que existen y sus posibles puntos a favor y en contra, tomando en cuenta los factores externos a la precisión de los modelos.

3.1.1. Inspección con robots escaladores

La inspección realizada con robots escaladores se realiza a través del transporte de los robots a lo largo de las líneas de transmisión. Debido a que es una inspección a corta distancia, la precisión de la inspección es muy alta, ver figura 3.1.



(a)

Figura 3.1: imágenes tomadas por un *climbing robot* y un modelo de estos. Imagen extraída de [10]

A grandes rasgos, los *climbing robots* son una buena solución para las técnicas de inspección de las líneas de transmisión. El problema son los factores que hay que considerar para que estos funcionen de una manera óptima y generalizada.

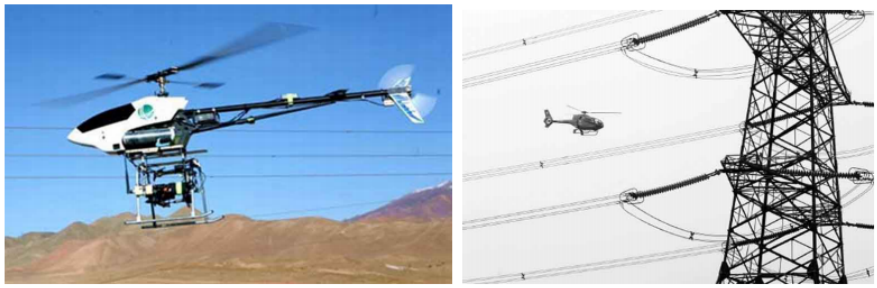
Los robots que tiene una estructura sencilla muchas veces no pueden terminar la inspección debido a la complejidad de las torres eléctricas. De otro modo, una estructura compleja en el robot incrementará su peso. Un robot muy pesado podría causar problemas para la línea de transmisión de electricidad.

Entonces, la problemática con estos robots es no tener un mecanismo lo suficientemente flexible para poder enfrentarse a cualquier diseño y adversidad que pueda proveer uno de los diferentes modelos de torres eléctricas a inspeccionar.

3.1.2. Inspección con helicópteros

Debido a la velocidad y maniobrabilidad de los helicópteros, estos han sido utilizados en gran manera en la inspección de líneas de electricidad, especialmente en terrenos complejos. Los equipan con distintos sensores como telescopios, cámaras aéreas, y sistemas de observación fotoeléctrica, ver figura 3.2.

Realmente son opciones viables para realizar una inspección correcta y de larga duración, sin embargo, tienen un detalle de alto costo. Ya que las inspecciones que se realizan abarcan una larga distancia y logran una inspección veloz, demasiados detalles pueden llegar a ser adquiridos y afectar la precisión de la detección [10].



(a) (b)

Figura 3.2: (a) Helicóptero equipado con equipo para inspección. Imagen extraída de [11] (b) Inspección en china realizada sobre líneas de 500kv.

3.1.3. Inspección con VANT

Un VANT puede ser equipado con herramientas para darle una percepción del entorno, como en los helicópteros, muchos sensores son integrados al VANT para darle este aumento de percepción del entorno. Se necesita un sistema de navegación autónomo, un algoritmo de detección de objetos, un algoritmo para realizar el rastreo de estos, entre otras cosas.

Su problemática principal es que tienen un periodo activo muy corto antes de tener que recargar sus baterías, y su distancia de inspección es muy corta por la misma razón. Una solución propuesta

para solucionar este problema [12], fue implementar hangares inteligentes capaces de solucionar el problema de la capacidad de energía de las baterías, el modelo gráfico se ve en la figura 3.3.

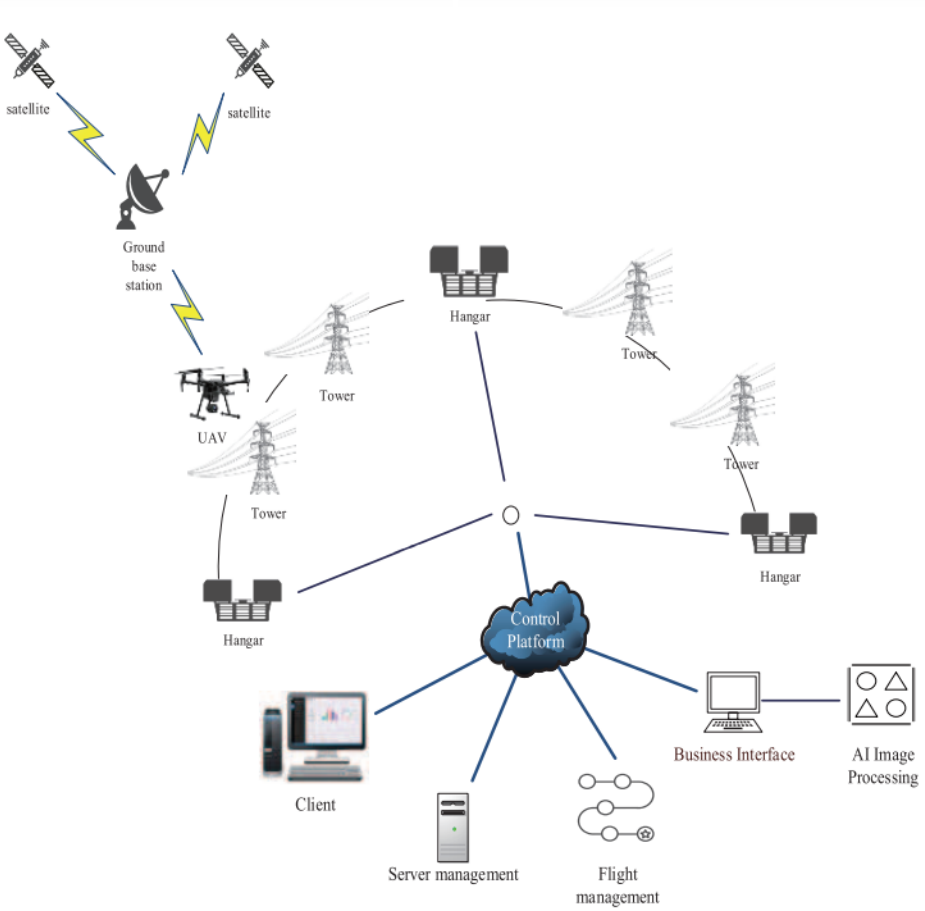


Figura 3.3: Solución propuesta para la navegación continua de un VANT. Imagen extraída de [12]

3.2. Implementaciones en detección de objetos

Existen muchas implementaciones para realizar la detección de objetos, pero el factor a considerar es cuál es la técnica más adecuada para la tarea a realizar. Aquí se enlistan las implementaciones actuales, sus puntos fuertes y débiles.

3.2.1. R-CNN

Las R-CCN (del inglés, *Region-Based Convolutional Neural Network*) es un conjunto de redes neuronales basadas en regiones. La primer implementación de esta red fue realizada en el 2014 por Ross Girshick [13], logrando una ejecución exitosa de redes neuronales convolucionales al problema de localización, detección y segmentación de objetos.

La red consta de 3 módulos, ver figura 3.4.

El módulo 1 realiza una generación y definición de posibles regiones independientemente de su categoría, es decir, posibles cajas contenedoras de objetos de interés. El módulo 2 extrae las características de cada una de las regiones propuestas, aquí es donde se hace uso de una red neuronal convolucional profunda. En el módulo 3, se clasifican las características para saber a qué clase pertenecen.

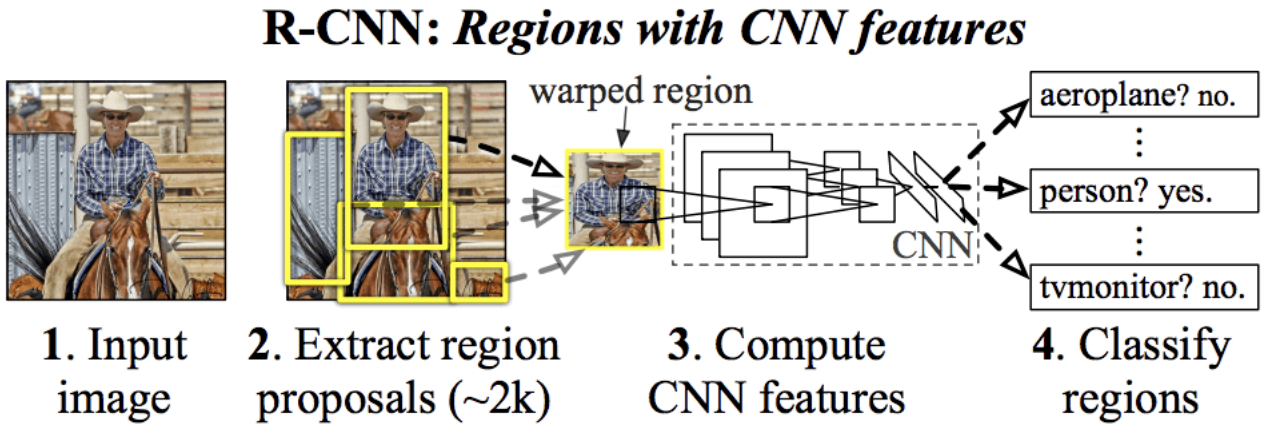


Figura 3.4: Arquitectura de la red R-CNN. Imagen extraída de [13]

La técnica de visión por computadora llamada 'Búsqueda selectiva' [14] se emplea en el modelo para definir las cajas contenedoras, esta consiste a grandes rasgos en definir regiones en toda la imagen, realizar iteraciones, y en cada iteración fusionar las regiones que sean muy parecidas entre sí.

El extractor de características empleado en este modelo es la red neuronal convolucional profunda *AlexNet* [15]. La salida de esta red consiste en un vector de 4096 dimensiones que describe el contenido de la imagen.

La salida de la red neuronal convolucional, es procesada por una máquina de soporte vectorial (*SVM*, por sus siglas en inglés). Donde para cada clase, se realiza el entrenamiento de una máquina de soporte vectorial.

La limitante más grande de este acercamiento es su velocidad, esto debido a que se realiza una extracción de características basada en una red neuronal por región, y como el modelo lo describe, se proponen alrededor de 2000 regiones por imagen durante el tiempo de prueba.

3.2.2. Fast R-CNN

Como su nombre lo indica, este modelo es una versión más rápida que su versión original, Ross Girshick [16] describe al inicio del artículo las principales limitaciones de R-CNN como sigue:

- **El entrenamiento se realiza en 3 modelos separados**
  Esto involucra la preparación y el manejo de cada uno de estos.
- **Entrenar una CNN con demasiadas regiones por imagen**
  Esto tiene demasiado costo temporal.
- **La detección de objetos es lenta**
  Al utilizar una CNN para la detección, nuevamente el tener tantas regiones por imagen vuelve el proceso muy lento.

Una solución al problema de la extracción de características fue una técnica llamada *Spatial Pyramid Pooling* [17] que consiste en agregar una capa entre las capas convolucionales y las capas completamente conectadas, y su trabajo es hacer un mapeo de una entrada de cualquier tamaño a un tamaño fijo de salida. Esto lo realiza dividiendo la imagen en cuadriculas de tamaño fijo, y concatenando los resultados, de esta forma se mantiene el mismo tamaño en cualquier imagen que se reciba en la entrada.

La arquitectura del modelo toma un conjunto de regiones como entrada, que son procesadas por una red neuronal convolucional profunda. Una CNN como VGG-16 [18] es utilizada para la extracción de características, y la salida de esta es la entrada para una capa completamente conectada, que el modelo bifurca en dos salidas: una para la predicción de las clases con una capa *softmax* y la otra con un regresor lineal para cada caja contenedora. Este proceso se repite múltiples veces para cada región propuesta para la imagen.

En la figura 3.5 se puede ver este proceso.

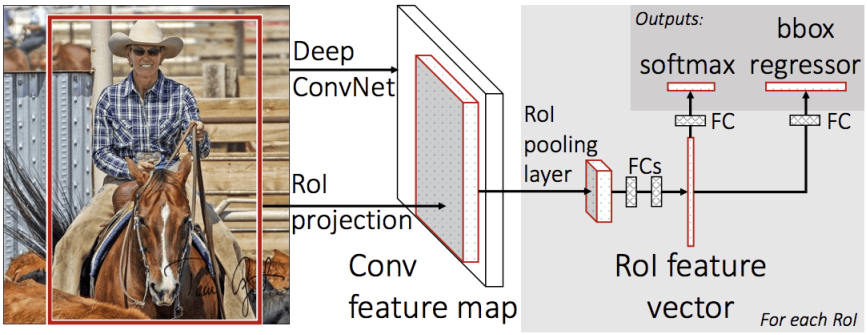


Figura 3.5: Arquitectura de la red R-CNN a grandes rasgos. Imagen extraída de [16]

Este modelo es realmente superior en cuestiones de velocidad a la hora de hacer el entrenamiento y hacer predicciones, pero de igual manera necesita un conjunto de regiones para cada imagen de entrada.

3.2.3. Faster R-CNN

Este modelo está compuesto de dos módulos: el primer es una red neuronal convolucional profunda que genera regiones de interés, y el segundo es el detector que utiliza *Fast R-CNN* para utilizar esas imágenes.

La técnica que utilizan en este modelo [19] consiste en entrenar a la red convolucional que genera las regiones, y con la salida de esta red, entrenar el clasificador. El clasificador también recibe una salida a la capa que propone las regiones, aportando información sobre donde podrían encontrarse estos objetos, ver figura 3.6.

Para disminuir el número de regiones candidatas que genera la capa convolucional, se utiliza un *Non Max Supression* en las regiones candidatas basándose en el valor de la intersección sobre la unión.

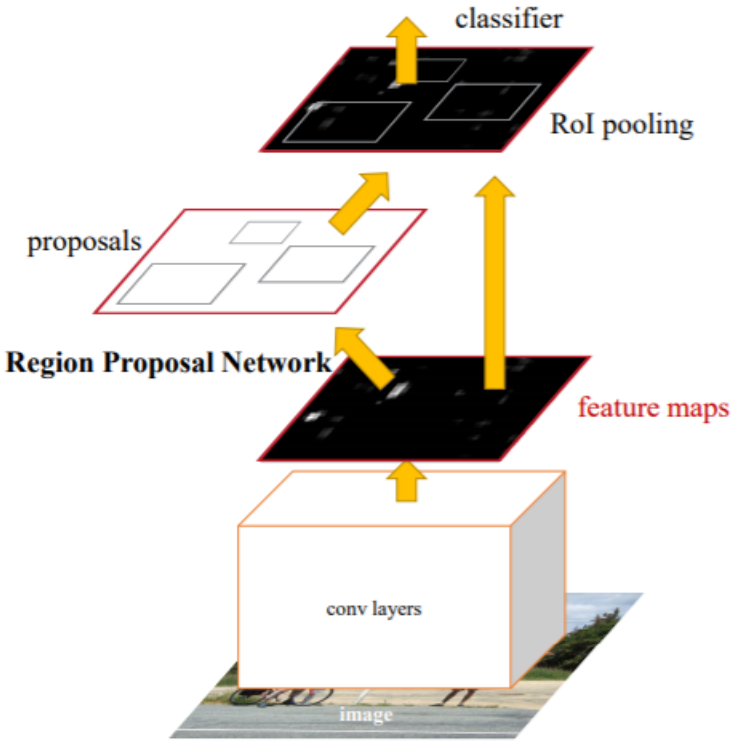


Figura 3.6: *Faster R-CNN* a grandes rasgos. Imagen extraída de [19]

3.2.4. Red YOLO - YOLOv3

La red neuronal YOLO fue propuesta por primera vez por Joseph Redmon [1], también Ross Girshick, autor de la R-CNN, contribuyó para este trabajo.

El mecanismo de detección de YOLO está basado en una sola CNN, que predice las cajas con-

tenedoras de los objetos y la probabilidad de detección de la clase del objeto para cada caja contenedora. La figura 3.7 ilustra este método:

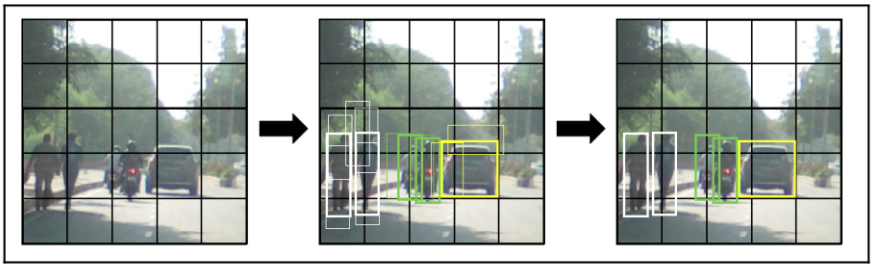


Figura 3.7: Funcionamiento general de YOLO. Imagen extraída de [8]

Para este proyecto, utilizamos la red YOLOv3 [9] ya que las mejoras realizadas mostraron un incremento considerable en la precisión de la clasificación.

Desde la YOLOv2, se implementó el uso de cajas ancla, que son cajas contenedoras de una medida predefinida antes de realizar el entrenamiento. Esto con el fin de detectar múltiples objetos por celda.

Arquitectura de la red

La red consiste en 23 bloque residuales y de convolución entre las capas 1 y 74. donde la imagen de entrada baja de su resolución de entrada (en este caso, se utilizará 416×416 como resolución) a 13×13, y la profundidad aumenta de 3 a 1024 esto gracias a la alternación entre un filtro de 3×3 y uno de 1×1, ver figura 3.8.

El ***stride*** se mantiene como 1 en la mayoría de los casos, con excepciones en donde el valor del *stride* es 2 para reducir el tamaño de la entrada y en las capas donde se realizan las detecciones, junto con un filtro de 3×3.

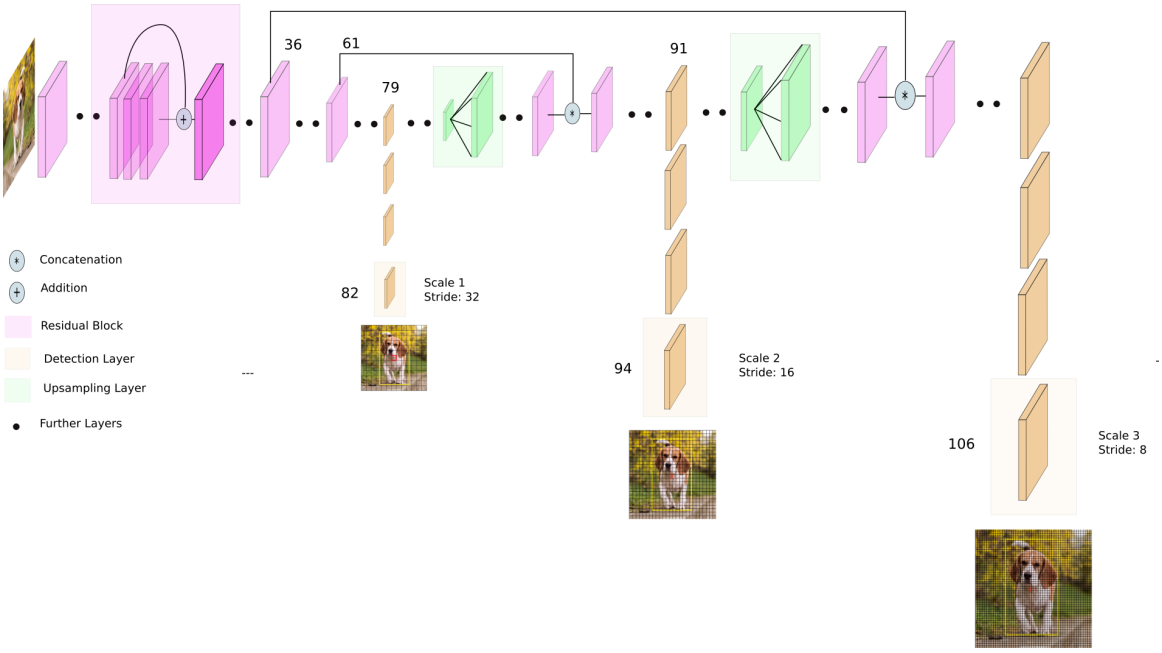


Figura 3.8: Arquitectura de YOLO. Imagen extraída de [8]

3.2.5. Funcionamiento general de YOLOv3

Para YOLOv3, tenemos que la red realiza la detección en 3 capas diferentes, y en cada una de estas capas, la red divide la imagen en un número de celdas, y realiza un número de predicciones correspondiente al número de cajas anclas que se estén utilizando.

Primero se divide con un *stride* de 32, resultando en una imagen de 13×13. Para cada celda de esta imagen, se realiza la detección con las cajas ancla para la detección de objetos grandes. Después se hace un *upsampling* para incrementar el tamaño de la imagen, se concatena la entrada de la capa 61 a la capa anterior a la segunda detección para aumentar el número de características. Se realiza la segunda detección utilizando un *stride* de 16, resultando en una imagen de 26×26 y aquí se detectan objetos de un tamaño medio. La salida vuelve a pasar por un proceso de —*upsampling*, y se concatena la salida con la entrada de la capa 36. La última detección se realiza en la capa 106, con un *stride* de 8, el resultado es una imagen de 52×52 para realizar la detección de los objetos más pequeños.

Después de realizar las detecciones, la red utiliza un algoritmo de *Non-Max Supression* para eliminar las predicciones que corresponden al mismo objeto, esto debido a las cajas ancla, terminando con una sola caja con el mayor índice de confianza.

Capítulo 4

Metodología de Solución

Para el proyecto, se pretende utilizar un VANT equipado con una cámara fotográfica para realizar las tomas de los objetos de interés en entornos simulados, una vez que estas imágenes se obtengan, se procesarán utilizando algoritmos de visión artificial.

4.1. Recolección de datos

Para solucionar el problema de la poca cantidad de imágenes disponibles, se utilizó *AirSim* [20], en conjunto con *Unreal Engine* [21] para realizar la recolección de datos. Estos dos *softwares* en conjunto brindan una gran capacidad de fidelidad para la generación de entornos artificiales realistas.

Se utilizaron cuatro entornos montados con diferentes características en *Unreal Engine*, cada uno de ellos fue dotado del sistema de simulación de VANT *AirSim*, de esta forma se lograron capturar imágenes digitales durante el tiempo de simulación desde la perspectiva del dron, ver figura 4.1.

Figura 4.1: Imágenes de los entornos utilizados para la recolección de datos.

4.1.1. Modelos de los objetos

Para obtener datos de entrenamiento que generalicen a los distintos objetos que se encuentran en los entornos reales, se utilizaron distintos modelos tanto de torres eléctricas, como de cadenas de aisladores. En esta sección se muestran los objetos utilizados y sus características.

Aisladores

Se utilizaron dos modelos de cadenas de aisladores, que a su vez pueden tener o no tener fallas, esto con la finalidad de generar datos diversos para el entrenamiento:

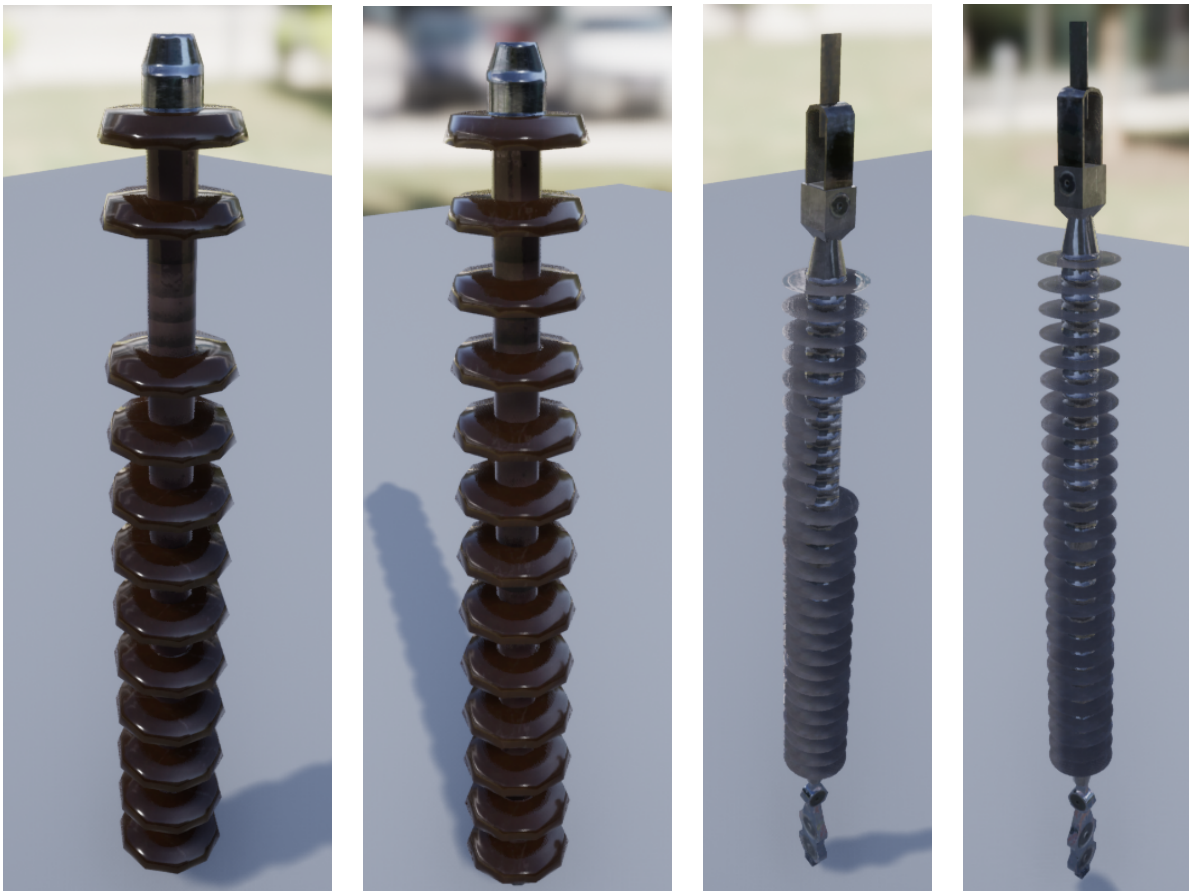


Figura 4.2: De izquierda a derecha: Cadena de aisladores de tipo 1 con y sin fallas. Cadena de aisladores de tipo 2 con y sin fallas.

Torres

Para las torres eléctricas, se utilizaron dos modelos de distinta forma y tamaño, con diferente número y diseño de cadenas de aisladores:

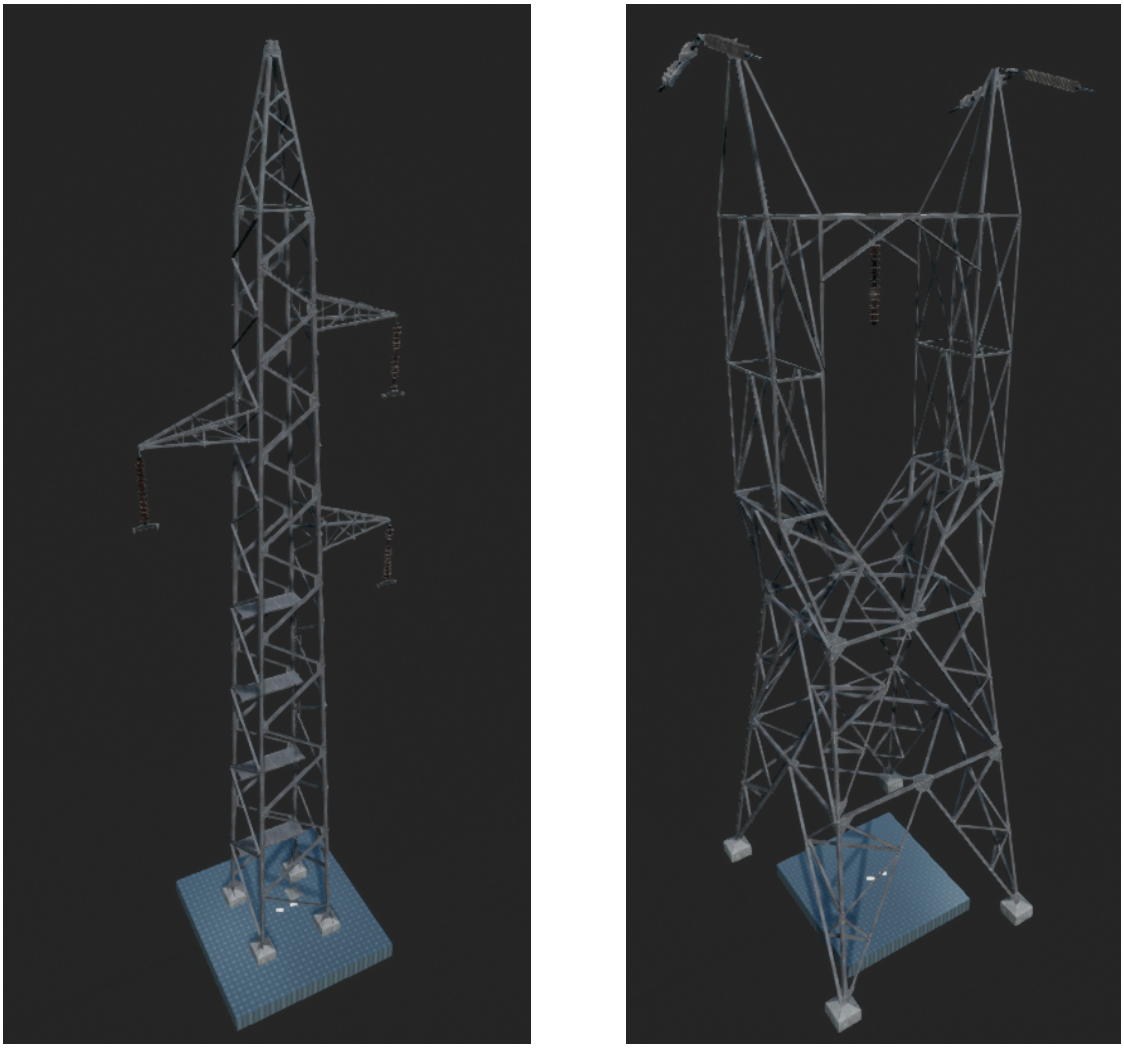


Figura 4.3: Torre 1 (izquierda) y Torre 2 (derecha)

La torre 1 cuenta con un cuerpo largo central, 3 brazos pequeños que sobresalen del cuerpo, y 3 aisladores del tipo 1 al final de cada brazo.

La torre 2 es más grande y ancha, su estructura central cuenta con un hueco central que tiene un aislador tipo 1 y termina en punta, donde se encuentran 2 aisladores tipo 2 apuntando a cada uno de los extremos.

4.1.2. Unreal Engine

Unreal Engine es un motor gráfico bastante utilizado en el desarrollo de *software*, videojuegos y aplicaciones 3D. Actualmente es el motor de libre acceso más utilizado en el mundo. Su gran capacidad gráfica y simulaciones físicas logran satisfacer las condiciones necesarias para recrear entornos fieles y reales a lo que podría encontrarse en la vida diaria.

Este permite la edición y el renderizado de los entornos en tiempo real, dando una gran capacidad de modelación y personalización para generar tomas y escenarios como se deseen, ver figura 4.4.

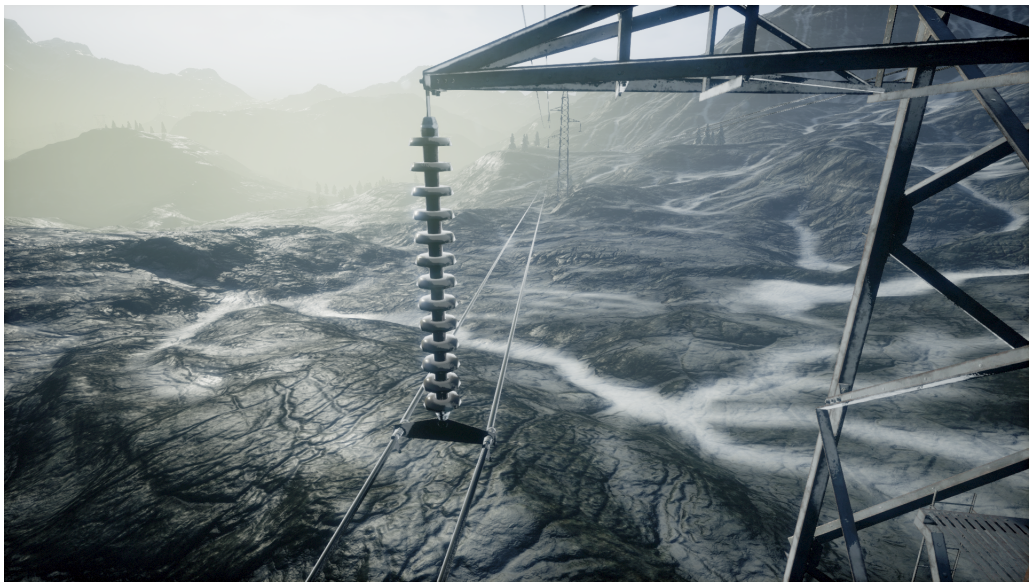


Figura 4.4: Captura de un entorno de *Unreal Engine* con una cadena de aisladores colocada en una torre eléctrica.

Este motor resulta el mejor candidato para la generación de datos sintéticos a partir de las imágenes obtenidas, no obstante, para lograr generar datos fidedignos, es necesario poder simular un VANT dentro de estos entornos.

Los entornos utilizados son:

- *Landscape Mountains*: Este entorno consiste en un lago grande, rodeado de pendientes, montes y árboles. Al ser muy extenso permite colocar las torres con el espacio adecuado entre ellas y encontrar diferentes escenas para las tomas.
- *EF Grounds*: Es más pequeño que *Landscape Mountains*, y tiene una atmósfera más oscura, al igual que cambios en la iluminación. Es una buena opción por la forma en la que se distribuyen los objetos y los árboles en el entorno
- *Automotive Beach*: Es pequeño y está compuesto por una playa larga junto con edificios y el cielo en el fondo. El contraste que tienen las torres en este entorno es muy distinto a los antes mencionados, por lo que proporciona datos variados para el entrenamiento.
- *City Park*: Es un parque muy grande de una ciudad, tiene un gran lago al centro, varios sitios comunes en las áreas urbanas, como canchas de deporte y calles.

4.1.3. AirSim

El *software AirSim* de Microsoft es utilizado para poder crear una instancia de simulación de un vuelo de un VANT, de tal forma que combinándolo con un entorno adecuado de *Unreal Engine*, se lograron generar ecosistemas apropiados para la generación de imágenes sintéticas de los componentes eléctricos de interés, ver figura 4.5.

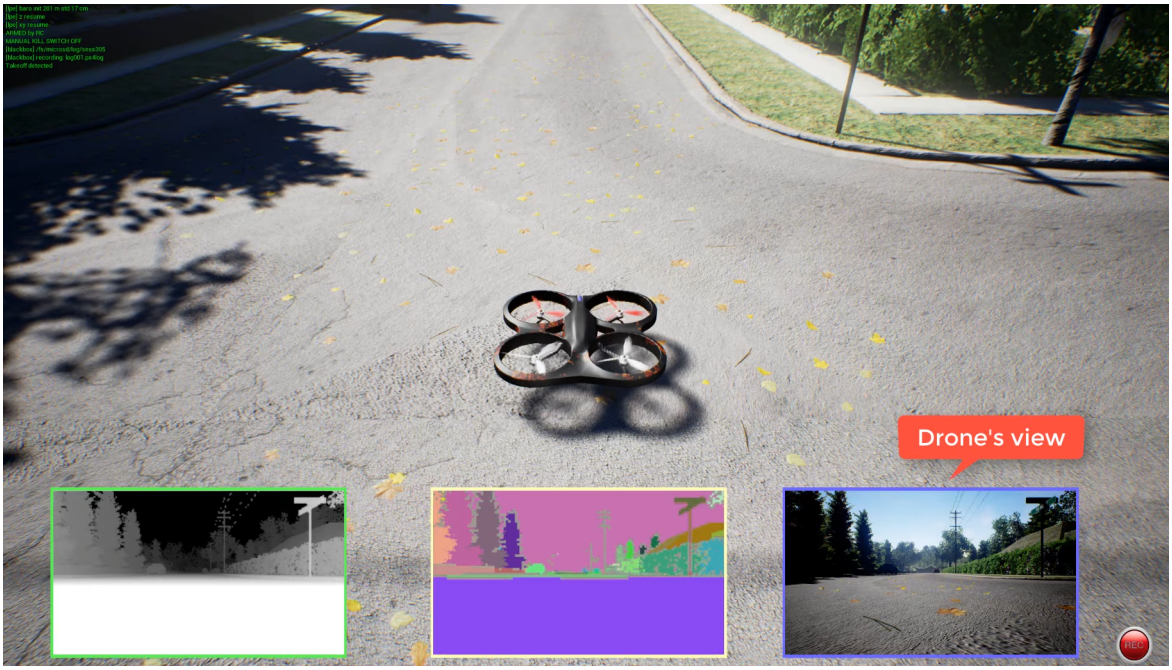


Figura 4.5: *AirSim* montado en *Unreal Engine*

4.1.4. Python

Python ha tomado el trono de la popularidad en los últimos años, y no es solo por la practicidad en la tipificación del lenguaje, también es su alta capacidad y flexibilidad para implementar modelos de *machine learning*. Se utilizará Python para la construcción del código del sistema por diversos motivos, entre ellos: su alta capacidad para la representación de datos y la cantidad de documentación e implementaciones de aprendizaje de máquina.

Python fue utilizado como base de programación para el uso de los modelos y las librerías necesarias para llevar a cabo el entrenamiento y las pruebas correspondientes de la red neuronal.

4.2. Etiquetado de datos

Para el etiquetado de los datos, se usó el programa *LabelImg* [23]. *LabelImg* es gratuito, y cuenta con las herramientas necesarias para generar un archivo .txt con las etiquetas y los valores de las

imágenes. YOLO recibe los datos en forma de *.txt*, por lo que esto es adecuado para esta red, ver figura 4.6.

El etiquetado genera unas coordenadas (X, Y) en la esquina superior izquierda de la imagen, una altura, un ancho y una etiqueta de clase a cada una de las cajas contenedoras.

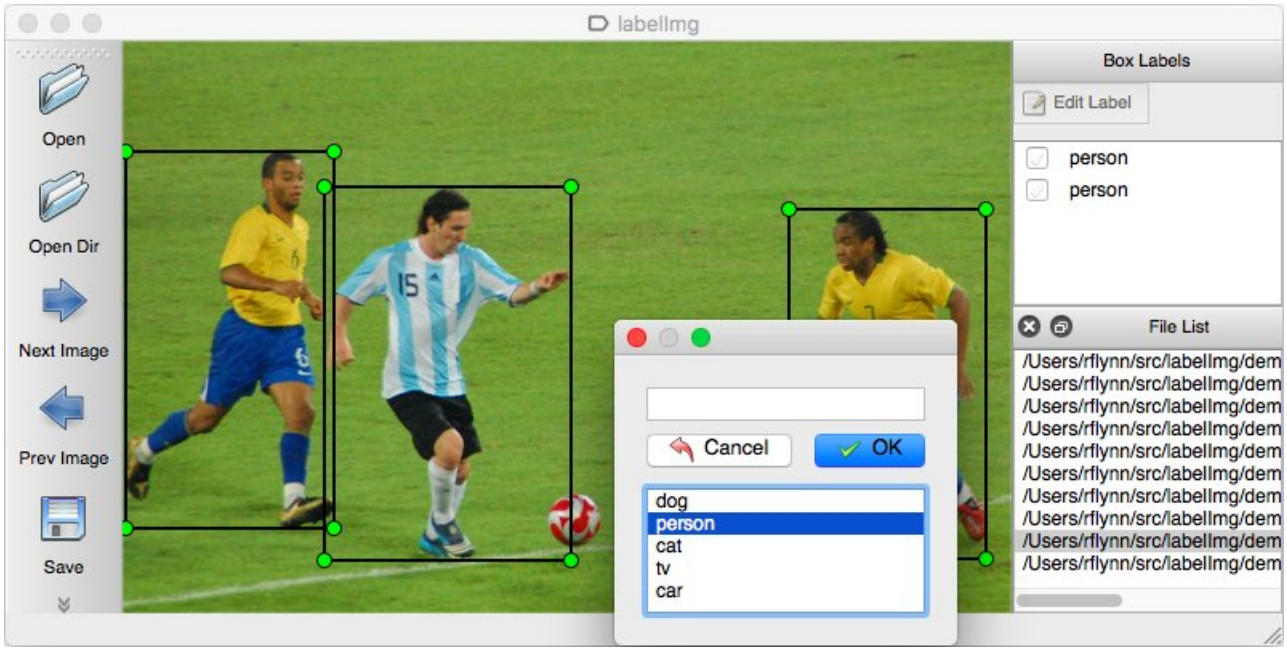


Figura 4.6: Ejemplo de las cajas que generan los datos en *LabelImg*. Imagen extraída de [23]

4.2.1. *Roboflow*

Roboflow es una plataforma que crea productos de software como servicio para facilitar y agilizar el desarrollo de modelos de visión por computadora. Este proporciona herramientas de manejo, etiquetado y ordenamiento de conjuntos de datos para el entrenamiento de redes neuronales.

Este software es empleado para ordenar los conjuntos de datos, y revisar el etiquetado de las cajas contenedoras de las cadenas de aisladores y las torres eléctricas. De igual manera, permite generar diferentes versiones del conjunto de imágenes que pueden o no contener aumento de datos.

Su implementación fue de gran utilidad para depurar, ordenar y exportar los datos en el formato necesario para el funcionamiento correcto de la red neuronal, ya que el formato de las cajas contenedoras para YOLO se basa en el centro de la imagen.

4.3. Construcción de conjuntos de entrenamiento, validación y *test*

Se tienen 4 entornos, de los cuales se obtuvieron 500 imágenes por entorno. 250 imágenes para las torres eléctricas y 250 imágenes para las cadenas de aisladores, que a su vez están divididos en dos clases, con fallas y sin fallas, esto para poder incrementar la variedad de datos en el entrenamiento.

Para que la entrada de la red pueda recibir las imágenes sin causar ningún problema, estas tienen que ser reescaladas a 416×416 pixeles. YOLO recibe una entrada de cualquier tamaño, y el reescalado puede realizarlo la red, pero esto puede causar conflictos con las cajas contenedoras y causar problemas en esa imagen, resultando en fallos en la detección.

En un inicio, el conjunto de entrenamiento constaba de un total de 1500 imágenes, y el conjunto de validación de 500, pero al notar ruido en algunas imágenes durante el entrenamiento, se descartaron imágenes problemáticas, dejando al final un total de 1476 para el entrenamiento, y 492 para la validación.

El conjunto de entrenamiento está compuesto por imágenes de los entornos *Landscape Mountains*, *EF Grounds* y *Automotive Beach*, mientras que el conjunto de validación tiene en su totalidad imágenes del entorno *CityPark*.

También, el conjunto de *test* está constituido por 54 imágenes, las cuales corresponden a entornos reales.

Python y conjuntos

Haciendo uso de Python, se crearon los sets de entrenamiento y de prueba mezclando los directorios de las imágenes de los conjuntos en un solo .txt; este código se utilizó para generar el archivo .txt de los 3 conjuntos, donde vienen los directorios de las imágenes y su etiqueta correspondiente.

4.4. Configuración de la red

Para la optimización del modelo y la disminución de las pérdidas se modificaron hiperparámetros, los cuales se enlistan a continuación.

Batch size

Para el entrenamiento, se usara un *batch size* de valor 16, a diferencia del valor original de YOLO de 64. Esto significa que 16 imágenes serán utilizadas por iteración, para actualizar los parámetros de la CNN.

Momentum

El *momentum* es un parámetro encargado de minimizar los grandes cambios en los pesos entre *batches*, esto porque solamente una pequeña porción de la imagen es procesada en cualquier punto de tiempo. El valor predeterminado es 0.9, y es el que utilizaremos para realizar el entrenamiento.

Learning rate

El *learning rate* indica que tan rápido un *batch* se está aprendiendo. El valor predeterminado para la red es 0.001 y no se pretende cambiar.

Anchor Boxes

Las cajas ancla son el mecanismo que permite que YOLO pueda realizar múltiples detecciones por celda. La primer versión de YOLO generaba un vector de la forma $[P_i, X_i, Y_i, W_i, H_i, C_0, C_i, ...C_n]$ donde C es una clase asignada a la caja contenedora del conjunto de datos. Estas cajas se definen en las capas de detección de YOLO, y sirven para asignarle más de una detección a los vectores de salida. Un vector de salida con dos cajas ancla y dos clases se ve de la forma:

$$y = [P_i, X_i, Y_i, W_i, H_i, C_0, C_1, P_j, X_j, Y_j, W_j, H_j, C_0, C_1] \tag{4.1}$$

Un vector de salida contiene un conjunto de datos correspondiente a cada una de las cajas ancla. Si se tienen 9 cajas ancla, como es el caso de este proyecto, se tendrán 9 tuplas de datos para cada una de las celdas en las que sea dividida la imagen, todas en el mismo vector de salida.

Las anclas por defecto que utiliza YOLO tienen las siguientes medidas:

$$[10, 13, 16, 30, 33, 23, 30, 61, 62, 45, 59, 119, 116, 90, 156, 198, 373, 326]$$

Estas anclas ayudan al modelo a aproximar las cajas contenedoras de las predicciones, y proporcionan la información para realizar la detección de múltiples objetos. Ahora, estas anclas pueden ser calculadas para un modelo personalizado. Utilizando el script de Alexey [9], podemos hacer uso de un algoritmo de clusterización KMeans.

Usando los datos de las anotaciones y realizando el cálculo, tenemos que las anclas personalizadas para nuestro conjunto de datos son las siguientes:

$$[4, 16, 6, 30, 7, 1419, 49, 25, 93, 31, 1378, 219, 136, 58, 151, 282]$$

Pesos pre-entrenados *darknet53*

Estos pesos son utilizados en nuestro modelo con la finalidad de facilitarle las detecciones de los objetos al modelo. Se utilizan los pesos *darknet53* entrenados en el conjunto de datos de *ImageNet* [24].

4.5. Métricas de desempeño

La manera de evaluar el sistema y sus resultados es a través de métricas que representan la precisión y los errores de las predicciones.

4.5.1. IoU (*Intersection Over Union*)

IoU es una métrica de evaluación para la detección de objetos basada en el grado de solapamiento que tiene la caja contenedora de la predicción, y la caja contenedora del *ground truth*, que es la caja contenedora que es etiquetada a mano. Un ejemplo de esto se puede ver de la siguiente manera:

$$IOU = \frac{\text{área de solapamiento}}{\text{área de la unión}} = \frac{\text{área compartida por las cajas contenedoras}}{\text{área total abarcada por ambas cajas contenedoras}} \tag{4.2}$$

Y en su representación gráfica, ver figura 4.7.

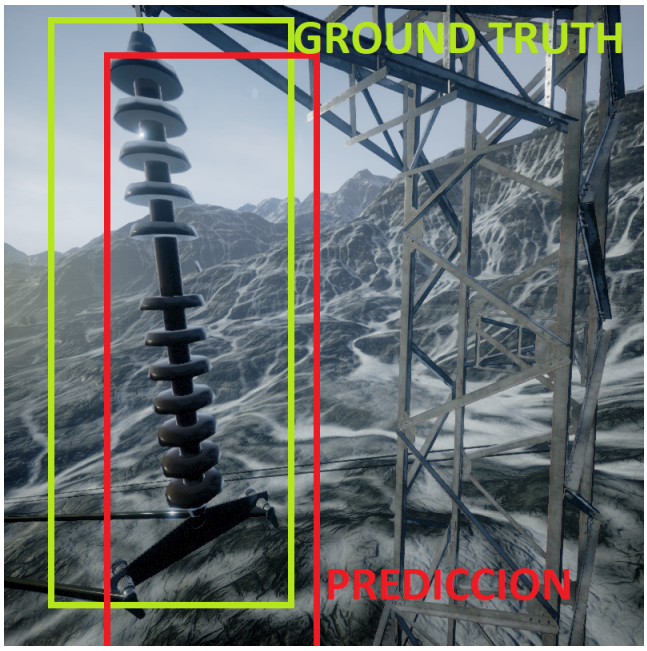


Figura 4.7: Ejemplo gráfico de la métrica *IOU*. Imagen extraída de [21]

Si las dos cajas contenedoras no se solapan, entonces el valor IoU es 0, por otra parte, si las dos cajas se solapan al 100 %, entonces el valor IoU es 1. Un mayor solapamiento lleva a valores de IoU más cercanos a 1, que a su vez implica una mejor ubicación de las cajas contenedoras definidas por la predicción.

4.5.2. Función de pérdida GIoU

Una problemática del IoU como métrica, es el caso en el que las predicciones son malas y la caja contenedora de la predicción no tiene intersección con la caja del *ground truth*, lo que provoca que el IoU sea 0. Si una predicción futura obtuviera un mejor acercamiento a la etiqueta del *ground truth*, el IoU siempre sería 0 sin importar la distancia de las cajas, y tendría un gradiente 0. Esto afecta la calidad del entrenamiento y el ritmo de convergencia. Una solución propuesta por Hamid Rezatofighi [25] es el ***GIoU***.

El $GIoU$ (*Generalized Intersection over Union*) es una métrica de precisión que a su vez puede ser utilizada como función de pérdida. Se calcula de la siguiente manera:

$$GIoU = \frac{|A \cap B|}{|A \cup B|} - \frac{|C \backslash (A \cup B)|}{|C|} = IoU - \frac{|C \backslash (A \cup B)|}{|C|}$$

Figura 4.8: $GIoU$. Imagen extraída de [25]

donde A es el área de la caja contenedora del *ground truth*, B es el área de la caja contenedora de la predicción y C es la caja contenedora más pequeña que abarca a estas dos. Aquí un ejemplo gráfico:

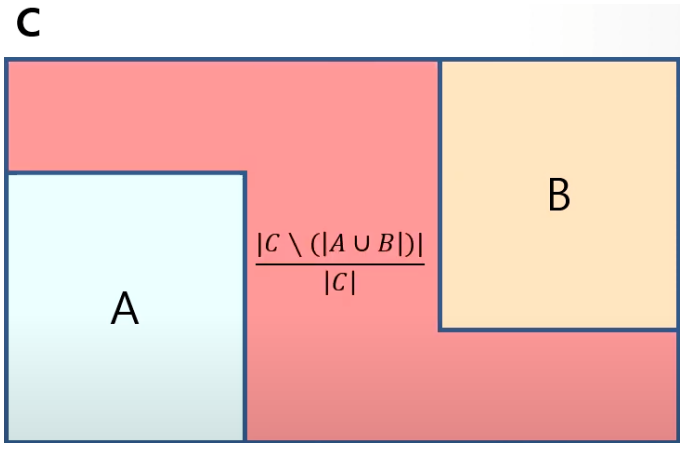


Figura 4.9: $GIoU$. Imagen extraída de [25]

Como se muestra en la figura 4.9, el *GIoU* tiene gradiente en todos los casos posibles, lo que hace posible utilizarla como una función objetivo. Si las cajas están muy separadas entre ellas, el *GIoU* devuelve números negativos, por lo que si las cajas siguen alejándose este valor converge a -1.

Para el cálculo del *GIoU* como pérdida, se utiliza la siguiente ecuación:

$$GIoU_L = 1 - GIoU \tag{4.3}$$

4.5.3. Binary Cross-Entropy

Para la tarea de clasificación de las cajas contenedoras, se utiliza la función de pérdida de la entropía-cruzada binaria, definida de la siguiente manera:

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i))$$

Figura 4.10: *BCE*. Imagen extraída de [26]

donde y_i es la etiqueta para las clases, y $p(y_i)$ es la predicción de la probabilidad de la caja contenedora para pertenecer a una clase, para cada caja contenedora.

Esta métrica es utilizada para calcular la pérdida de las predicciones realizadas por el modelo para su índice de *objectness* y para la clasificación de los objetos.

4.5.4. Métrica F1

La precisión y el *recall* por si solos no son una buena unidad de medida para evaluar la precisión de un modelo, pero ambos valores son igual de importantes en este tipo de evaluación. La métrica F1 se define como la media armónica entre la precisión y el *recall*, esto es:

$$F1 = \frac{2(P)(R)}{P + R} \tag{4.4}$$

Esto facilita el poder comparar el rendimiento combinado de estos dos valores en distintas soluciones.

4.5.5. mAP@0.5

Una métrica utilizada para el cálculo de la precisión de un modelo es el AP o *Average Precision*. El AP se calcula graficando el *recall* y la precisión de los valores de las predicciones, en un margen de IoU específico para cada imagen. Esto es, para cada valor de confianza de cada caja contenedora, se determina si son falsos positivos o verdaderos positivos utilizando un valor de IoU, en este caso, 0.5.

Luego, una vez calculada la precisión y el *recall*, estos valores se grafican, resultando de la siguiente manera:

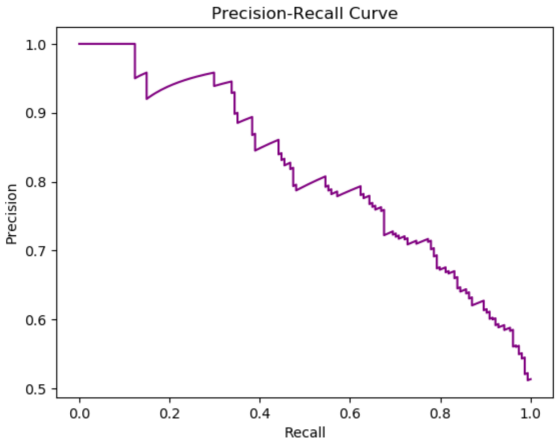


Figura 4.11: Ejemplo de gráfica entre precisión y *recall*

Después de obtener la gráfica, se calcula el área bajo la curva, y es de esta manera como se obtiene el AP. Para obtener el mAP, se realiza el mismo procedimiento para cada una de clases del modelo, y se promedia entre el número de clases que existen. Esto de manera generalizada se ve en la siguiente ecuación:

$$mAP = \frac{1}{N} \sum_{k=1}^{N} AP_i \tag{4.5}$$

Capítulo 5

Pruebas y resultados

En esta sección, se recopilan las pruebas realizadas para la conclusión del modelo final, tomando en cuenta las restricciones que determinan las herramientas con las que se cuenta actualmente. Así mismo, se discuten las diferentes configuraciones en cada una de estas y el por qué fueron o no de utilidad en el modelo final.

5.1. Detección de objetos en imágenes sintéticas

En los primeros entrenamientos, se realizó un ajuste de hiperparámetros tratando de minimizar las pérdidas y maximizar la precisión de este.

Inicialmente se entrenó sin modificar los hiperparametros, utilizando:

- 1476 imágenes de entrenamiento
- 492 imágenes de validación
- 1114 aisladores totales.
- 1036 torres totales.
- 100 épocas
- 16 de *batch size*
- Cajas ancla por defecto de YOLOv3

obteniendo valores de precisión y recuerdo entre 0.80 y 0.81, un $GIoU$ de 0.5 aproximadamente y F1 de 8.3.

Para intentar mejorar el rendimiento del sistema, se utilizó una rotación de imágenes en un rango de ángulos de [0°,15°], esto para incrementar los ángulos en relación a las tomas de los objetos desde el dron. Sin embargo, los valores de pérdida del $GIoU$ incrementaban hasta 0.8, y la precisión y

el recuerdo se estancaron alrededor de 0.7 a 0.72. Esto deterioró el rendimiento del modelo, por lo que esta configuración fue descartada.

De igual manera, se utilizó un conjunto de validación de 246 imágenes para realizar una comprobación de ambas mitades del conjunto. Los resultados fueron malos, el $GIoU$ se estancó en 0.5 y la precisión no logró superar los 0.7. Por lo que este esquema de pruebas también fue descartado.

Fueron utilizadas diversas técnicas de aumento de datos, como son traslación, rotación, flip, brillo, *cutout* y sus combinaciones, pero los resultados no mejoraron. Por lo que no se utilizaron en el modelo final.

Figura 5.1: Aumentos de datos, de izquierda a derecha: Original, *cutout*, flip horizontal, brillo y rotación

Caso Roboflow

Haciendo uso de *Roboflow*, se generaron imágenes con aumento de datos de flip horizontal, esto debido a que en las predicciones realizadas en el conjunto de validación, se realizaban detecciones con aisladores posicionados en una diagonal derecha, pero no en una diagonal izquierda. Con esta mejora, se generaron 697 imágenes más para el conjunto de entrenamiento. Quedando de la siguiente manera:

- 2193 imágenes de entrenamiento
- 492 imágenes de validación
- 1506 cadenas de aisladores totales
- 1420 torres totales
- 100 épocas
- 16 de *batch size*
- Cajas ancla por defecto de YOLOv3

Los resultados en el conjunto de validación fueron realmente prometedores, la precisión fue mayor, al igual que la métrica F1, como se aprecia en la siguiente tabla:

Class	Images	Targets	Precision	Recall	mAP@0.5	F1
all	492	521	0.905	0.930	0.952	0.916
insulator	492	271	0.850	0.926	0.933	0.887
tower	492	250	0.959	0.934	0.971	0.946

Cuadro 5.1: Resultados en el conjunto de validación con aumento de datos de Roboflow

Sin embargo, en la evaluación con las imágenes de *test*, los resultados del modelo fueron peores que los del modelo final utilizando las cajas ancla personalizadas. Esto es probable que se deba a un *overfitting* del modelo al conjunto de entrenamiento al contener más imágenes.

Modelo final

Con base en todos los resultados descritos en esta sección, el modelo final resultante utilizó la siguiente configuración:

- 1476 imágenes de entrenamiento
- 492 imágenes de validación
- 1114 aisladores totales
- 1036 torres totales
- 100 épocas
- 16 de *batch size*
- Cajas ancla personalizadas calculadas con KMeans.

Resultados del entrenamiento en la última época:

Class	Images	Targets	Precision	Recall	mAP@0.5	F1
all	492	521	0.849	0.931	0.033	0.888
insulator	492	271	0.842	0.926	0.929	0.882
tower	492	250	0.856	0.936	0.938	0.984

Cuadro 5.2: Resultados obtenidos en el conjunto de validación

Las gráficas resultantes del entrenamiento y del conjunto de validación, son las siguientes:

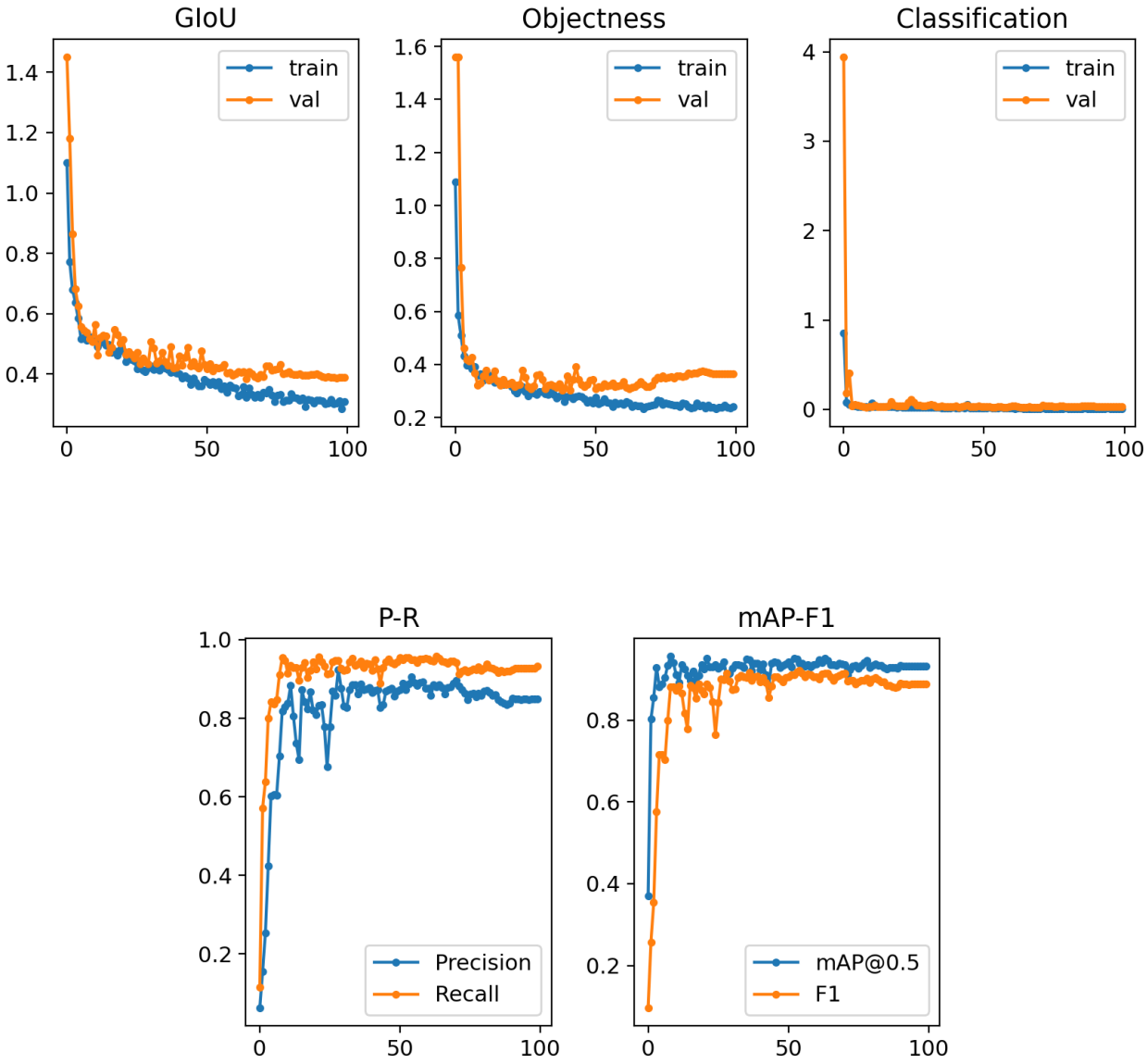


Figura 5.2: Gráficas finales del entrenamiento y de la validación

Las gráficas corresponden al comportamiento del modelo a lo largo de las 100 épocas de entrenamiento. En las gráficas superiores, las líneas azules son el comportamiento del entrenamiento, y las naranjas son el comportamiento del conjunto de validación.

Se puede observar que para el conjunto de validación, el *GIoU* decrece alrededor de 0.4, donde su valor mínimo es 0.38.

La medida de *objectness* decrece a un mínimo de 0.3, pero aumenta nuevamente terminando en 0.4. Esto quiere decir que el modelo al final comenzó a perder precisión en el cálculo de la probabilidad de que exista un objeto para las cajas contenedoras.

Para la pérdida de la clasificación, esta disminuye hasta 0.03, lo que implica que el modelo no tiene problemas clasificando si detecta una torre o una cadena de aisladores.

En las inferiores, está representado el comportamiento de la precisión y el *recall* en las 100 épocas para el conjunto de validación, y como se estabilizan en las últimas épocas.

La gráfica de *mAP-F1* proyecta el comportamiento de estas dos métricas, lo cual es un buen indicador para entender qué modelo estaba realizando una buena detección, y si las detecciones que realiza son precisas.

5.2. Detección de objetos en imágenes reales

Utilizando el modelo final descrito en la sección anterior, se realizó la inferencia en imágenes reales, las cuales fueron recolectadas de la red. Se usaron 54 imágenes, en las cuales había un total de 80 objetos, 29 cadenas de aisladores y 51 torres eléctricas. Los resultados de la inferencia fueron los siguientes:

Class	Images	Targets	Precision	Recall	mAP@0.5	F1
all	54	80	0.850	0.922	0.950	0.884
insulator	54	29	0.834	0.863	0.930	0.848
tower	54	51	0.866	0.980	0.971	0.920

Cuadro 5.3: Resultados de evaluación con imágenes reales

5.2.1. Casos de imágenes exitosas

A continuación, se presentan las salidas del modelo en las imágenes reales de manera cualitativa. Los resultados son bastante buenos, y se puede apreciar que incluso torres lejanas y modelos no utilizados en el entrenamiento son detectados.

Figura 5.3: Resultado de YOLOv3 en imágenes con diferentes torres eléctricas reales

Figura 5.4: Resultado de la detección de cadenas de aisladores eléctricos reales usando YOLOv3

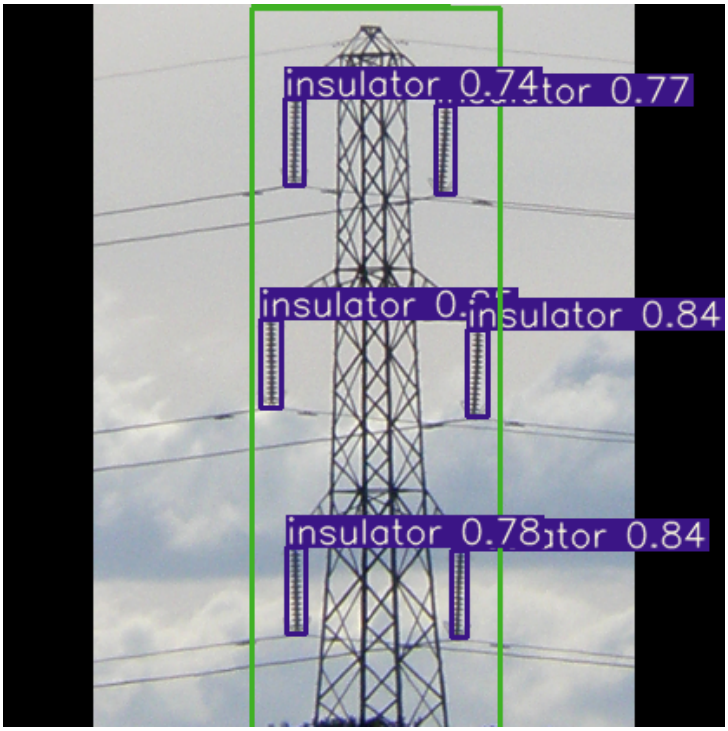


Figura 5.5: Resultado de la detección de una torre eléctrica y diferentes aisladores usando YOLOv3

5.2.2. Casos de imágenes fallidas

En esta sección se revisan las imágenes fallidas en detección o en comportamiento de detección, de igual manera se discute el por qué podría estar ocurriendo estos errores y posibles soluciones.

En la figura 5.6, el modelo no logró detectar la torre eléctrica ni las cadenas de aisladores. Esto es muy probablemente por el ángulo de la toma, y la cercanía de la torre. Las imágenes de entrenamiento son tomas completamente aéreas, por lo que una toma desde el suelo puede abarcar un área distinta en la imagen, y los detalles ser distantes a los aprendidos por la red.

Figura 5.6: Caso de torre eléctrica no detectada con el modelo YOLOv3 entrenado

En la figura 5.7, solo se detectó un aislador teniendo 3 en la imagen. La probabilidad correspondiente al aislador es de 0.5, por lo que probablemente los demás si hayan recibido una caja contenedora, pero su valor de confianza fue menor a 0.5, que es nuestro rango de error del IoU. De igual manera, los aisladores son realmente diferentes a los utilizados durante el entrenamiento, por lo que esa puede ser la explicación del valor tan bajo de confianza.

Figura 5.7: Caso de aisladores eléctricos no detectados con el modelo YOLOv3 entrenado

Una ventaja de YOLO, es la capacidad de lidiar con objetos que se sobrepongan con otros para realizar la detección, sin embargo, el ángulo de la toma y los objetos que interfieren en la toma, no estuvieron presentes en las imágenes de entrenamiento, ver figura 5.8. Nuevamente, esto podría verse solucionado con un conjunto de datos con más variedad de entornos, y más tomas angulares, pero esto si el sistema observara los objetos de manera terrestre y no aérea.

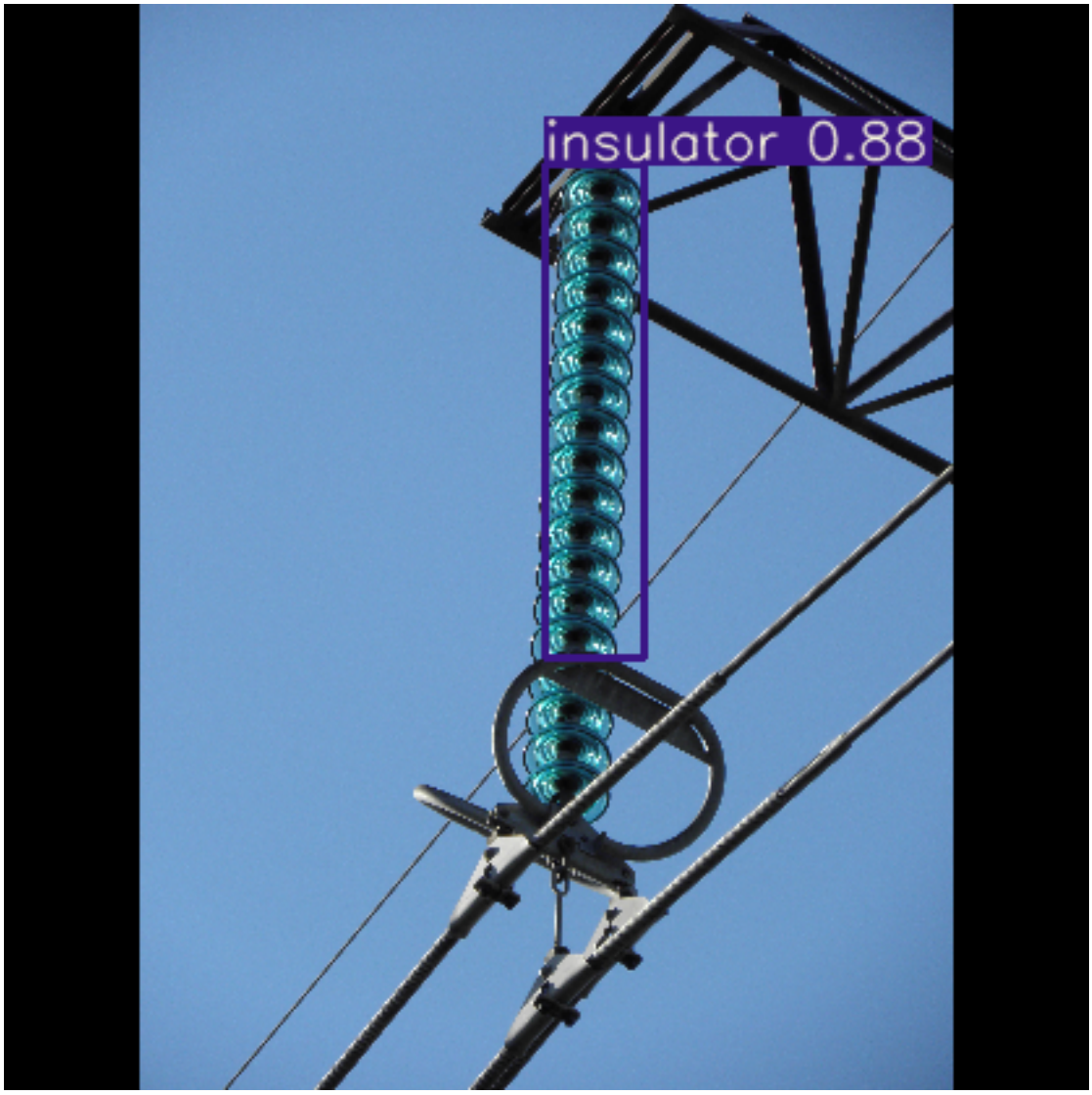


Figura 5.8: Imagen ejemplo

En la figura 5.9 se muestra un caso originado probablemente por la cercanía de la torre, el nivel de detalle y la definición. La segunda torre se encuentra en una distancia más acorde a las imágenes del conjunto de entrenamiento, sin contar que el ángulo de la captura sigue siendo muy bajo en comparación a las tomas aéreas.

Figura 5.9: Imagen ejemplo

5.2.3. Casos especiales

Los casos especiales son situaciones que no fueron previstas o que se encuentran fuera del alcance del proyecto.

La figura 5.10 contiene una cantidad de aisladores muy elevada, y muchos de ellos son cadenas de aisladores dobles. Este tipo de aisladores no se encuentra en el conjunto de entrenamiento y el detalle en los discos que tiene el par de aisladores verticales de la izquierda, es poco notorio.

Figura 5.10: Caso real con cadenas de aisladores dobles

En la figura 5.11 el sistema no logró identificar a las torres por separado, esto es lógico debido a que no existen imágenes en el conjunto de entrenamiento con dos torres alineadas.

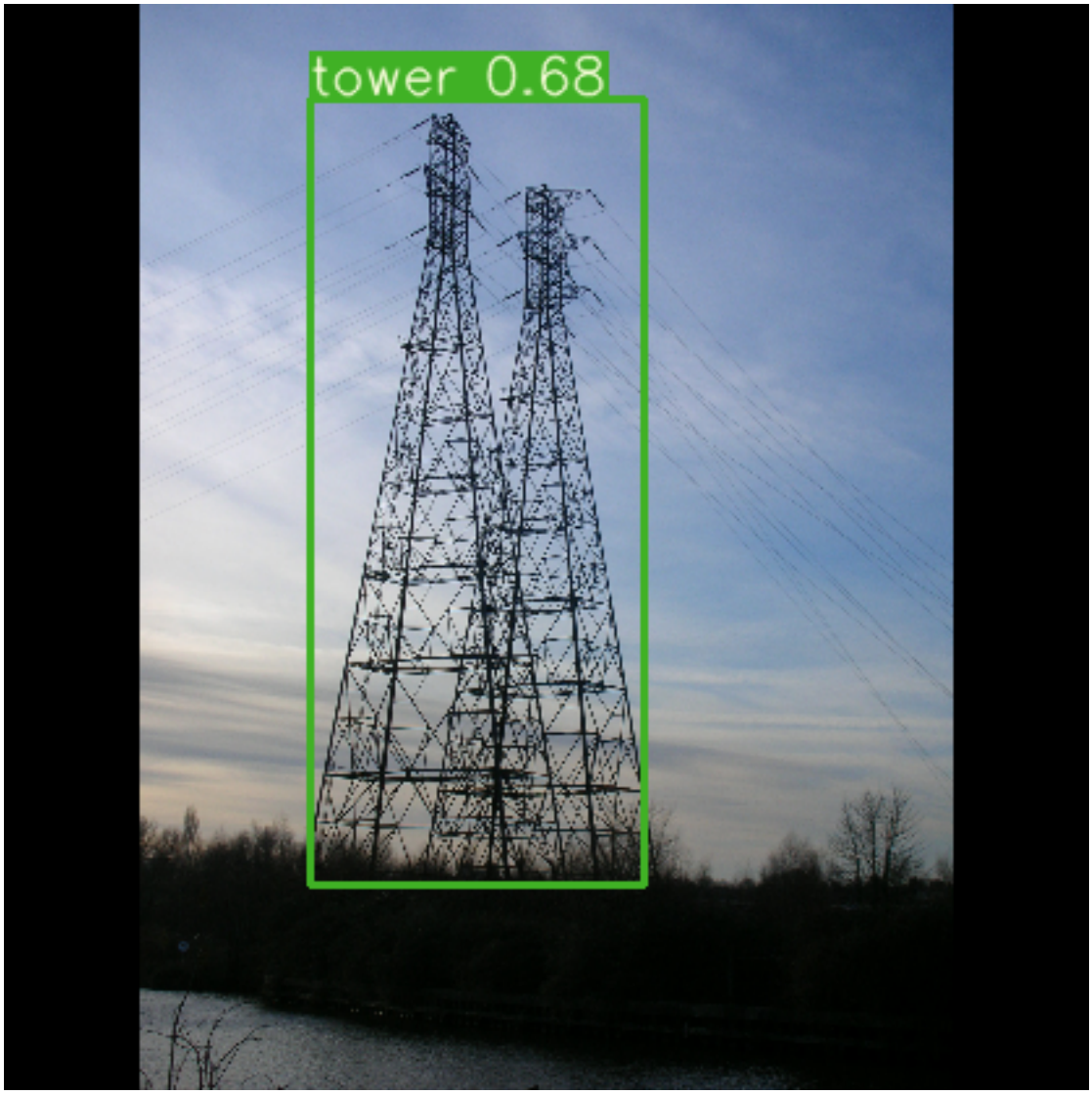


Figura 5.11: Caso real con torres eléctricas cuyas estructuras se encuentran solapadas.

En la figura 5.12 se realizaron dos predicciones correctas para las torres, pero el sistema detectó una torre en las ramas de un árbol con una confianza de 0.5. Las características de las torres pueden llegar a verse similares a las de las ramas de un árbol, pero esto puede ser ocasionado por las imágenes sintéticas, ya que las torres a distancia no logran tener un detalle tan nítido como las que se encuentran a la cercanía.

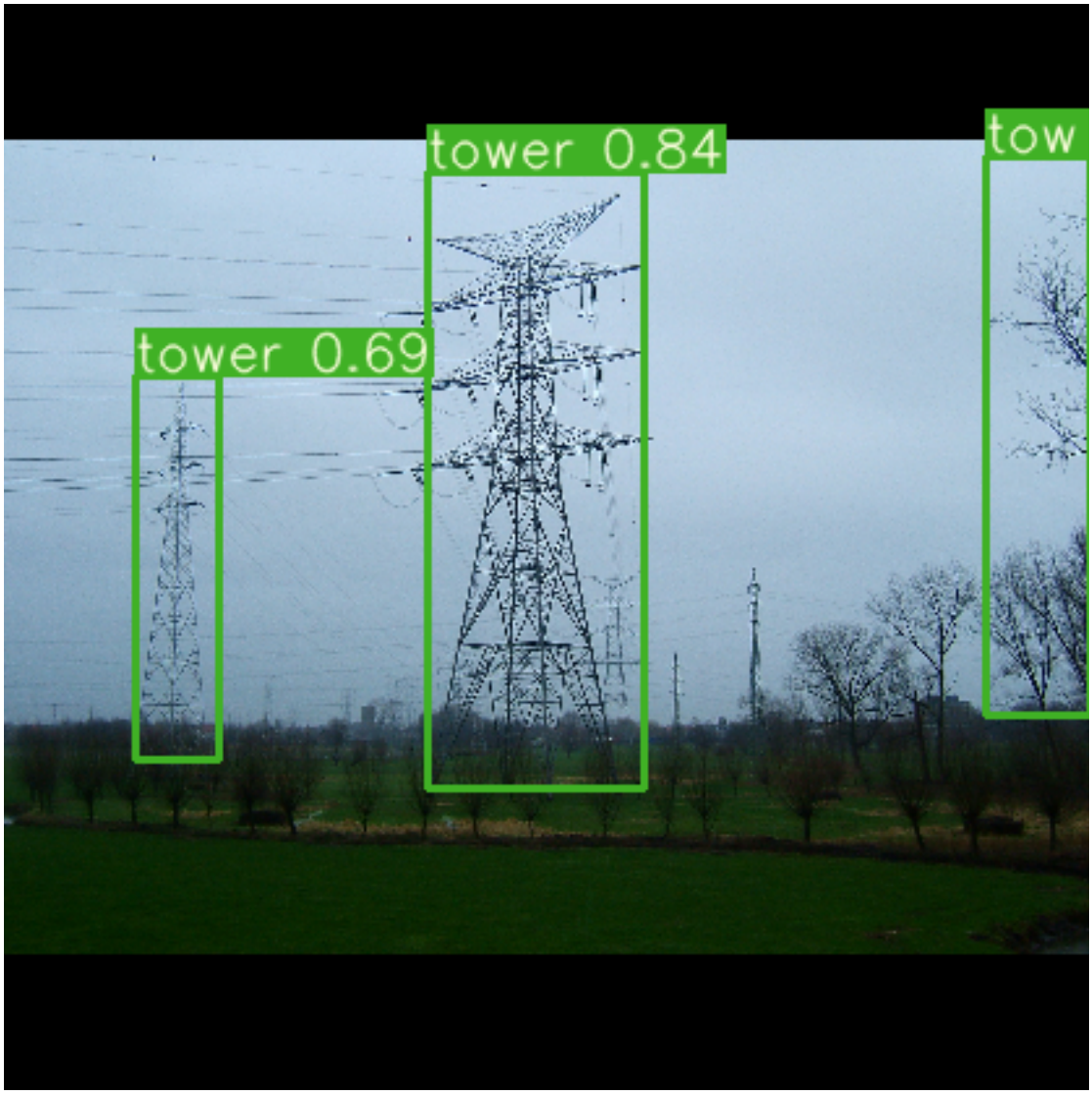


Figura 5.12: Caso real de ramas de árbol confundidas por el modelo como torres eléctricas

En la figura 5.13, se muestra que las primeras dos torres fueron detectadas debido a que se encuentran a una distancia razonable y son claramente identificables, incluso sin respetar los patrones de los modelos de las torres. Las últimas dos se encuentran demasiado lejos y no se logra distinguir bien la forma que tienen, incluso en las secciones medias de las torres hay espacios en blanco entre las partes de la torre, como si estuviesen fragmentadas.

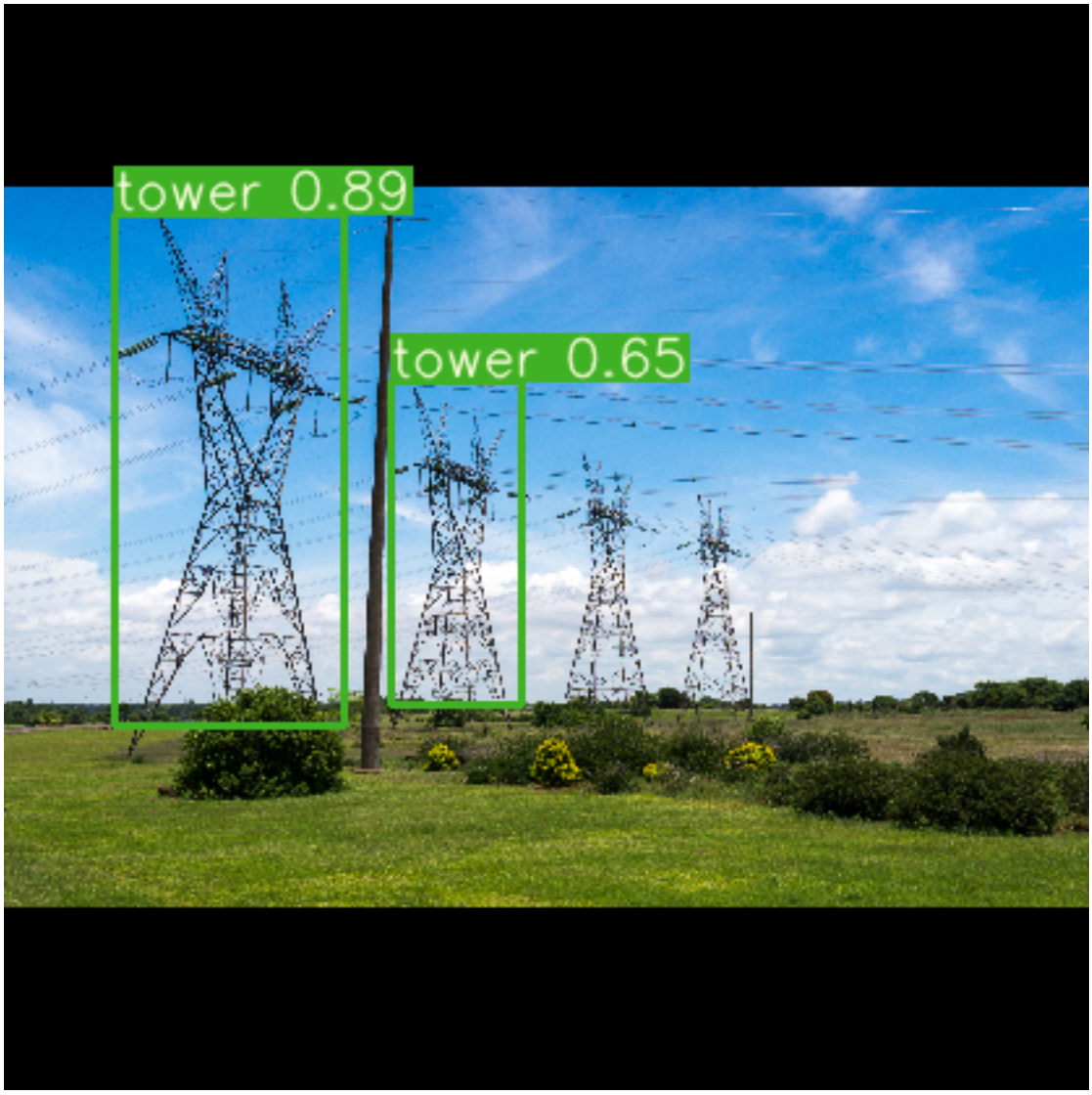


Figura 5.13: Imagen ejemplo. Pérdida de detalle y variaciones en diseño

Capítulo 6

Conclusiones

Se diseño y entrenó un sistema de detección automática basado en una red neuronal convolucional de tipo YOLO, para identificar torres eléctricas y cadenas de aisladores, entrenado a partir de imágenes sintéticas con la finalidad de evaluar su desempeño en imágenes reales.

Las imágenes sintéticas fueron generadas utilizando *Unreal Engine* y *Microsoft Airsim*, usando 4 entornos simulados diferentes, dos modelos de torres eléctricas y dos modelos de cadenas de aisladores.

Para optimizar los tiempos de desarrollo y optimización del sistema, se hizo uso de los pesos pre-entrenados *darknet53* entrenados en el conjunto de datos de *ImageNet* [24].

Este sistema demostró tener un buen desempeño en imágenes reales, con algunos puntos a considerar para futuros proyectos que puedan derivar de este.

Se lograron identificar diversas deficiencias en el modelo, entre ellas: una nula detección de torres eléctricas muy cercanas, problemas detectando torres eléctricas que están fragmentadas debido a la distancia, falta de detecciones individuales en torres eléctricas traslapadas, poca generalización de los aisladores dobles y dificultades para detectar objetos que se encuentran obstruidos por otros.

Para trabajos futuros, las debilidades del sistema podrían ser mitigadas con conjuntos de entrenamiento más grandes y diversos, para mejorar el aprendizaje y la generalización de las características de las imágenes, y el uso de YOLOv5 [27] u otros modelos más recientes [28] para incrementar la eficiencia del modelo.

Bibliografía

[1] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 779-788).

[2] https://www.automate.org/vision

[3] Gao, H. (2018, 16 junio). Object Localization in Overfeat - Towards Data Science. Medium. https://towardsdatascience.com/object-localization-in-overfeat-5bb2f7328b62

[4] colaboradores de Wikipedia. (2021, 16 febrero). Perceptrón. Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Perceptr

[5] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. nature, 323(6088), 533-536.

[6] S. (2020, 12 abril). Stochastic Gradient Descent on your microcontroller. Eloquent Arduino Blog. https://eloquentarduino.github.io/2020/04/stochastic-gradient-descent-on-your-microcontroller/

[7] Liang, J., Jing, T., Niu, H., & Wang, J. (2020). Two-terminal fault location method of distribution network based on adaptive convolution neural network. IEEE Access, 8, 54035-54043.

[8] Kar, K. (2020). Mastering Computer Vision with TensorFlow 2.x: Build advanced computer vision applications using machine learning and deep learning techniques. Packt Publishing.

[9] Redmon, J., & Farhadi, A. (2018). Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767.

[10] Yang, L., Fan, J., Liu, Y., Li, E., Peng, J., & Liang, Z. (2020). A Review on State-of-the-Art Power Line Inspection Techniques. IEEE Transactions on Instrumentation and Measurement, 69(12), 9350-9365.

[11] Jenssen, R., & Roverso, D. (2018). Automatic autonomous vision-based power line inspection: A review of current status and the potential role of deep learning. International Journal of Electrical Power & Energy Systems, 99, 107-120.

[12] Liu, Z., Wang, X., & Liu, Y. (2019). Application of unmanned aerial vehicle hangar in transmission tower inspection considering the risk probabilities of steel towers. IEEE Access, 7, 159048-159057.

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

[14] Felzenszwalb, P. F., & Huttenlocher, D. P. (2004). Efficient graph-based image segmentation. International journal of computer vision, 59(2), 167-181.

[15] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 25, 1097-1105.

[16] Girshick, R. (2015). Fast r-cnn. In Proceedings of the IEEE international conference on computer vision (pp. 1440-1448).

[17] He, K., Zhang, X., Ren, S., & Sun, J. (2015). Spatial pyramid pooling in deep convolutional networks for visual recognition. IEEE transactions on pattern analysis and machine intelligence, 37(9), 1904-1916.

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

[19] Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. Advances in neural information processing systems, 28, 91-99.

[20] AirSim. (s. f.). Microsoft Research. Recuperado 8 de agosto de 2021, de https://microsoft.github.io/AirSim/

[21] Unreal Engine — The most powerful real-time 3D creation platform. (s. f.). Unreal Engine. Recuperado 8 de agosto de 2021, de https://www.unrealengine.com/en-US/

[22] OpenCV. (s. f.). OpenCV. Recuperado 8 de agosto de 2021, de https://opencv.org/

[23] Tzutalin. LabelImg. Git code (2015). https://github.com/tzutalin/labelImg

[24] https://www.image-net.org/index.php

[25] Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I., & Savarese, S. (2019). Generalized intersection over union: A metric and a loss for bounding box regression. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 658-666).

[26] Godoy, D. (2022). Understanding binary cross-entropy / log loss: a visual explanation. Retrieved 24 January 2022, from https://towardsdatascience.com/understanding-binary-cross-entropy-log-loss-a-visual-explanation-a3ac6025181a

[27] https://github.com/ultralytics/yolov5

[28] Liu, L., Ouyang, W., Wang, X., Fieguth, P., Chen, J., Liu, X., & Pietikäinen, M. (2020). Deep learning for generic object detection: A survey. International journal of computer vision, 128(2), 261-318.

VOTOS DE APROBATORIOS

Secretaria Ejecutiva del Instituto de Investigación en Ciencias Básicas Aplicadas de la Universidad Autónoma del Estado de Morelos.
P r e s e n t e.

Por medio de la presente le informamos que después de revisar la versión escrita de la tesis que realizó el C. **REYES RAMÍREZ ANDRÉS ALBERTO** con número de matrícula **10003183** cuyo título es:

"Detección automática de torres eléctricas y cadenas de aisladores en imágenes aéreas".

Consideramos que **SI** reúne los méritos que son necesarios para continuar los trámites para obtener el título de **LICENCIADO EN CIENCIAS ÁREA TERMINAL EN CIENCIAS COMPUTACIONALES Y COMPUTACIÓN CIENTÍFICA.**

Cuernavaca, Mor a 09 de mayo de 2021

Atentamente
Por una universidad culta

Se adiciona página con la e-firma UAEM de los siguientes:

Dr. Juan Manuel Rendón Mancha	(Presidente).
Dr. Jorge Hermosillo Valadez	(Secretario).
Dr. Jorge Alberto Fuentes Pacheco	(Vocal).
Dra. Lorena Díaz González	(Suplente).
Dr. Audberto Reyes Rosas	(Suplente).

Se expide el presente documento firmado electrónicamente de conformidad con el ACUERDO GENERAL PARA LA CONTINUIDAD DEL FUNCIONAMIENTO DE LA UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MORELOS DURANTE LA EMERGENCIA SANITARIA PROVOCADA POR EL VIRUS SARS-COV2 (COVID-19) emitido el 27 de abril del 2020.

El presente documento cuenta con la firma electrónica UAEM del funcionario universitario competente, amparada por un certificado vigente a la fecha de su elaboración y es válido de conformidad con los LINEAMIENTOS EN MATERIA DE FIRMA ELECTRÓNICA PARA LA UNIVERSIDAD AUTÓNOMA DE ESTADO DE MORELOS emitidos el 13 de noviembre del 2019 mediante circular No. 32.

Sello electrónico

JORGE HERMOSILLO VALADEZ | Fecha:2022-05-09 10:41:52 | Firmante

I4CmaB9IfKxMFpX4NgFxeVq3zTH618n1nFJwhuQPh3ptzyI1XclYoASfXJfXZTGgpBzelvmKt0ae/FYozuGfOdglbvoske09gaqJBj2X0/1YAEbC4AwtY9l+8Dea8iVEm0WnFXbXEhu
5pwVdUxrdBUCTWkIxl3+kbsI3x+G7M+MlsLGhVwKPaPLw7FVJBH/L20K/0svy7IpulMB1r536gR5B3NvO1njEwz3oFCBjz5DhL31r2Zf89BXzY0NoRwwnjUE5kKEX9qsRBcnG8t7Z
Bs8XayVFv00I3HEU8DVrhH9l9v/ev9a5DY1su0RCHK76oFw8G6XBeAMxdyoOaPnbRg==

JORGE ALBERTO FUENTES PACHECO | Fecha:2022-05-09 11:29:02 | Firmante

CBZB3Jgj+js/OjtLAuB8EVZ4prFbXBvDyrR0VAbxwZTRVddjixfHjhoezIqBj9vccx1jEJZHAYnFaieeJIde2jQjtNv87eQNRZ4qI81tOP18Eri2BwjpFbH37EkkFQ24YlRgZ48hYDMhy9vVl
8bsEPPvQDwkM02VnL68KuAsl8C+uEm+XEiRJuxwbTQMZFov4bVQqdC9piXGiKUkXi5MqqrIdkQ0jJX6P69cmTcXNK+hLNY0hC5ZUbZ/kODtVAH4UcldJmjdiG6Mc5uROV8Cv8
C6c55L8Hzxh2rZN9UYz1xC8TpbQxcoc+CeXwAxr9yUD5lxXDGHE2dW820j5V4WOg==

JUAN MANUEL RENDON MANCHA | Fecha:2022-05-09 17:31:59 | Firmante

MpBSK32S6VWVofQozBPOFHj9dkwZn2YctF2zzaULYFxVZhIL0n/NWH9tRd8jwuWp4eAP37qhKS+NRAkUzGN/7faye9/skGEbJQ354ehrAA11SbtZEz1Q01DIxaV9VVb9Y9FsOG
6GhkqWGdFfu4b6BocCb+/j8KKtKsx1NleOyy4RogOu5J76BvU4mLRiMzPW9jFJeT92kn83MMoSZT01ykso/+deiSwK+0GNCPeD0N4WlXsV26sZ3s1X6l6M2KAue/JRh0ffg+Bwbcj
ti/3qO1SyKCLhyyzAT/f9plg2etqE7ziCSG9Kwh1SiywYRtn4G/mGf82QtJgVt6XxOoeGFw==

LORENA DIAZ GONZALEZ | Fecha:2022-05-12 19:03:12 | Firmante

RdqjKW+PzGOxTF6CtskIth+UwOFGdPLtWWOuYwogHvgoPjO2gPsHp2RRnY9935S9h5h2spph5oRWfM22uPrPhHyIai2ckQ3XahfArpdSHXBTJUaeWSMwxISkatyULSwivOxWF
/rtobanCX0EqMiuoq9a633GxbTD55sxrOi1U7376CMM3qf1h0b2Z34r88dT6ODxw3b1aTU7xc1Zth7WgCp/3ERnXisgT4jgMm/xMOUGjsG0LsT8ajSmXzzmGw4zxoE5lgY35WuDw
q/mqU+KTDBnm74bSfRCtCOlah8UZM5WiZiopOJLVnq6q54lSoN6WnCAATtEpiATXLBORb7G9g==

AUDBERTO REYES ROSAS | Fecha:2022-05-17 22:32:07 | Firmante

OaM0AqXfFFqk7fipy8RRTWWumiNojnCmJ75y9CyW477J+IJyV0jJAi21CYGnp6lfxETUH2pBvkYoU9nIzQFn5hPval5WGjcW2m/c7fDMnk6XtLtq/glkiLxysKpF9jaQWw3wvPR5oAs
77pa+5SJh77gTQTi5e96rG2Ht7q/9T3VfgMB9PUsUQjhJYgzGxSxf78vRUf34ACcEBqEIliTPefII5KQBc6b9bDHblA7UQqXR9zGlKXAjuvIXoylGHtFqU8K3+VrrwGAR915nTRPWof/
Eo0H+fVKe+pwdB4PMirtN4SwsqgjULlBV/G0nC/8Kv3xrBVRLrOg7fI2lHryCRw==

Puede verificar la autenticidad del documento en la siguiente dirección electrónica o escaneando el código QR ingresando la siguiente clave:

CJubB6mMw

https://efirma.uaem.mx/noRepudio/5huUAg8113KJpQTV4U3S5QNoxCn9UJpB