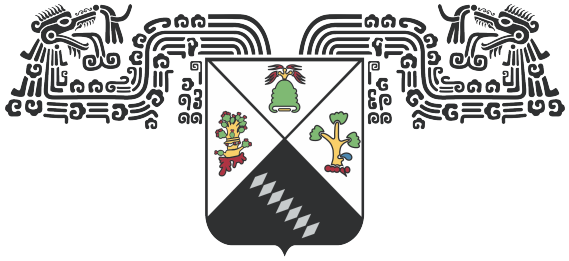


UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MORELOS

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

"Detección visual automática de cadenas de aisladores dañadas en líneas de transmisión a través de imágenes aéreas"

TESIS

QUE PARA OBTENER EL GRADO DE

LICENCIADO EN CIENCIAS

PRESENTA

JOSÉ EDUARDO NÚÑEZ ORTEGA

DIRECTOR DE TESIS:

Dr. Jorge Alberto Fuentes Pacheco

SINODALES:

Dr. Juan Manuel Rendón Mancha (Presidente)

Dr. Jorge Hermosillo Valadez (Secretario)

Dr. Lorena Díaz Gonzales (Suplente)

Dr. Audberto Reyes Rosas (Suplente)

A mi familia

Índice general

Índice de cuadros

Índice de figuras

Capítulo 1

Introducción

1.1. Problemática

Uno de los grandes problemas de la Comisión Federal de Electricidad (CFE) se encuentra en la inspección de líneas de transmisión eléctrica a causa de los altos costes que esta actividad conlleva, ya que las cadenas de aisladores eléctricos son propensos a romperse o cuartearse con facilidad. Esta actividad es realizada por personas, las cuales son propensas a accidentes laborales y otros factores de riesgo. De igual forma, dichas inspecciones se realizan entre lapsos de tiempo muy largos lo que lleva a sufrir desgastes o fallas en los aisladores que al final causan cortes recurrentes en el suministro eléctrico de la población y elevados gastos de mantenimiento. Por lo tanto, contar con otros medios, además del factor humano, para realizar estas inspecciones es de suma importancia.

1.2. Hipótesis de investigación

Es posible detectar fallas en cadenas de aisladores a través de una red neuronal convolucional siamesa usando pocas imágenes, donde el conjunto de imágenes para el entrenamiento será generado a partir de simuladores y el conjunto de imágenes de prueba obtenidas en entornos reales.

1.3. Objetivos

El objetivo de este proyecto es realizar un sistema de visión por computadora que clasifique de manera precisa si una cadena de aisladores de una torre de transmisión eléctrica se encuentra dañada o no. Como objetivos específicos se plantean los siguientes:

- Generación de un conjunto de imágenes aéreas que contendrán diferentes aisladores de subestaciones eléctricas mediante un dron en entornos simulados y reales.
- Modelación e implementación de una red neuronal convolucional siamesa para clasificar si una cadena de aisladores presenta piezas faltantes, además de devolver la probabilidad de dicho resultado.

1.4. Alcances

- La investigación se centra en el problema de la ausencia de discos aislantes.
- Las imágenes se clasificaron en cadenas de aisladores dañadas o sin daño, no se realizó la detección del objeto ni su segmentación.
- La red neuronal puede trabajar con un conjunto limitado de datos para el entrenamiento.

1.5. Limitaciones

- El proceso de entrenamiento no se realiza en tiempo real.
- Las cadenas de aisladores que contienen oclusiones parciales se consideraron como dañadas.
- El conjunto de datos debe contar con la misma cantidad de imágenes de cadenas de aisladores con fallas y sin fallas para evitar los problemas que conlleva realizar un entrenamiento con clases no balanceadas.

Capítulo 2

Marco Teórico

2.1. Conceptos básicos de redes neuronales

La clasificación automática de objetos en categorías específicas es uno de los problemas mayormente estudiados en el campo del aprendizaje de máquina dados sus múltiples usos no solo en la teoría si no también en la vida real. Por este motivo se han desarrollado distintas técnicas de solución y una de las más estudiadas por su precisión y eficacia son las redes neuronales (NN, por sus siglas en inglés), las cuales toman inspiración de los complicados sistemas biológicos cerebrales donde se conectan varias neuronas entre sí para obtener un resultado.

En un inicio, Rosenblatt, F (1961) definió los conceptos básicos como la unidad básica de funcionamiento de las redes: la neurona (también conocida como perceptrón) la cual sigue un mismo camino que su homóloga biológica al momento de operar. Cada neurona recibe señales de entrada en sus dendritas y produce señales de salida a través del axón donde se ramifica nuevamente a otras neuronas a través de la sinapsis propagando la señal, ver figura 2.1.1. En el modelo computacional, una señal propagada (por ejemplo, x0) interactúa multiplicándose (es decir, w0x0) con las dendritas de otra neurona basándose en la fuerza sináptica en esa sinapsis particular (w0). La idea es que esta fuerza sináptica sea aprendida y controlada de una neurona a otra.

En el modelo básico de una neurona artificial las dendritas llevan las señales al cuerpo de la neurona donde se suman y si esta suma sobrepasa cierto umbral, la neurona se puede activar dando un resultado sobre el axón, que en principio sería

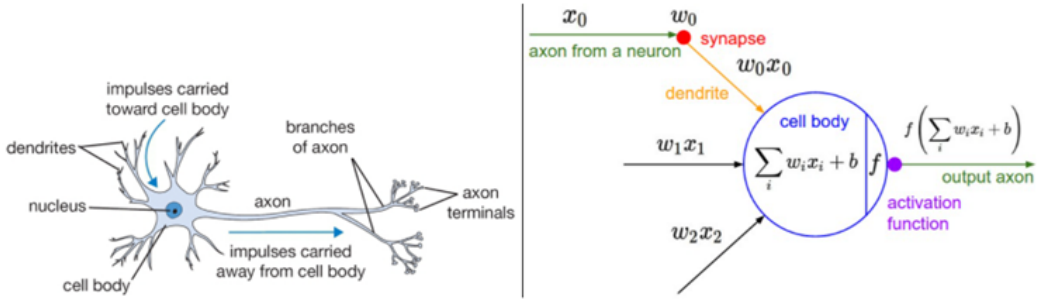


Figura 2.1.1: Ilustración de una neurona (izquierda) y un perceptrón (derecha). Adaptada de Biological motivation and connections[Ilustración], por CS231n:Convolutional Neural Networks for Visual Recognition, 2021, CS231n, (https://cs231n.github.io/neural-networks-1/)

un sí o no. Para modelar dicha activación se hace uso de una función de activación que representa la frecuencia de las activaciones en el axón.

Por sí solo, un perceptrón puede usarse para implementar un clasificador binario (por ejemplo, un clasificador binario o clasificador de máquina de soporte vectorial binario).

2.1.1. Funciones de activación más comunes

Hay distintas funciones matemáticas que determinan la salida de una red neuronal ya que estas mapean un número real a un intervalo entre 0 o 1, o bien entre -1 y 1 dependiendo de la función usada. Estas se dividen en dos grupos: lineales y no lineales. Las funciones de activación lineales están dadas por la formula $f(x) = x$ y por tanto la salida de la red no está limitada a un rango específico por lo que dicha función no es muy usada. Por otro lado, están las funciones no lineales que son las más usadas porque promueven la generalización o adaptación de modelos con diversos datos y la distinción entre sus resultados. En esta categoría se encuentran:

- Sigmoide: De la forma $\sigma(x) = \frac{1}{1+e^x}$. Ver figura 2.1.2.
- Tanh: De la forma $tanh(x)$, pero al ser un escalamiento de la función sigmoide se comprueba que $tanh(x) = 2\sigma(2x) - 1$. Ver figura 2.1.3.
- ReLu: De la forma $f(x) = max(0, x)$. Ver figura 2.1.4.

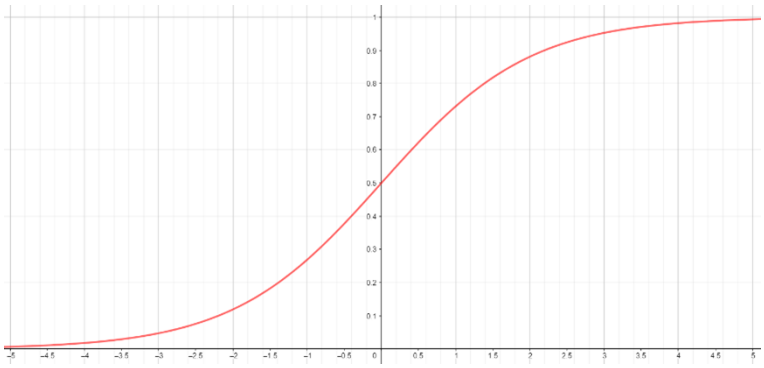


Figura 2.1.2: Función de activación sigmoide

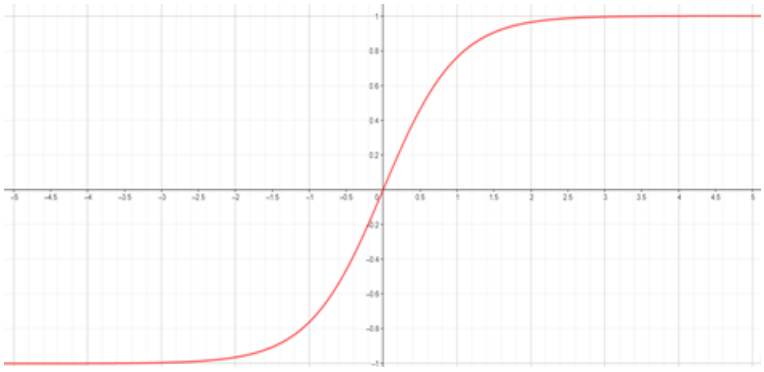


Figura 2.1.3: Función de activación de la tangente hiperbólica

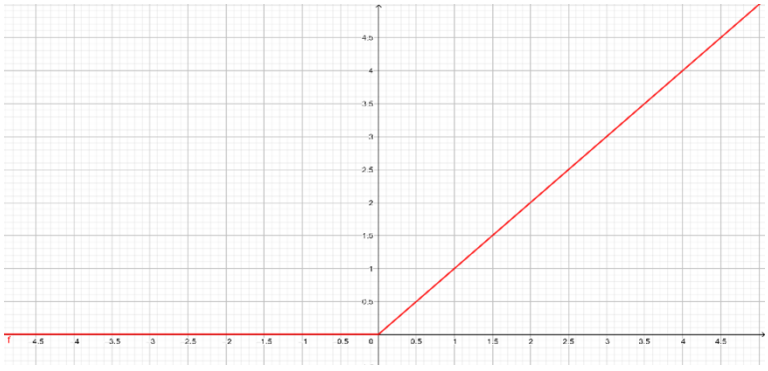


Figura 2.1.4: Función de activación ReLu

2.1.2. Arquitectura de las redes neuronales

Para modelar a las NN se ha hecho uso de grafos ya que, al ser una colección de neuronas conectadas entre sí, los grafos proporcionan una manera sencilla de representarlas. Sin embargo, existen ciertas restricciones: los ciclos no están permitidos, la dirección de las aristas es unidireccional y están organizadas por capas. Las capas reciben tres nombres: la capa de entrada, la capa de salida y las capas ocultas como se muestra en la figura 2.1.5.

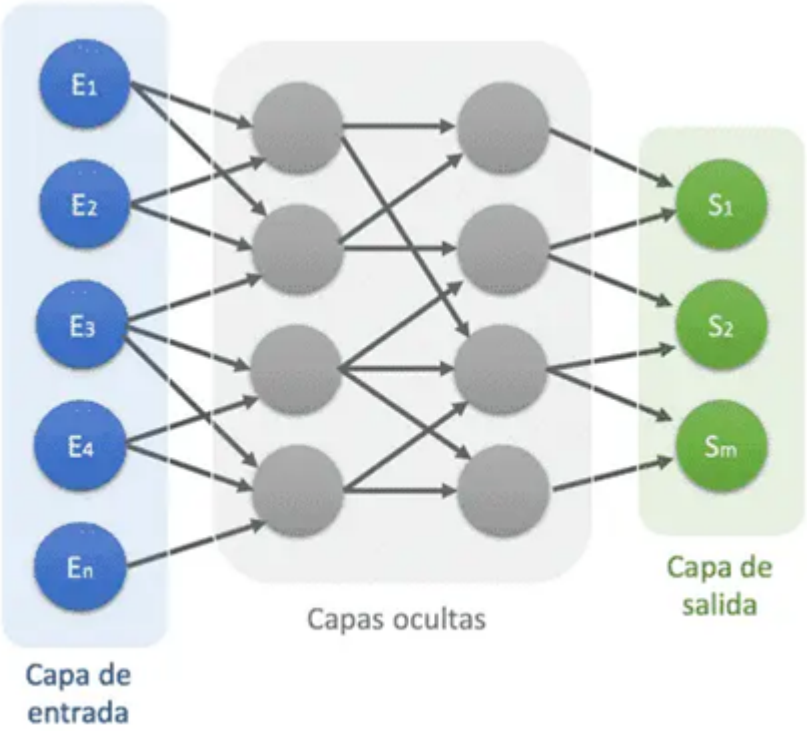


Figura 2.1.5: Ejemplo sencillo de una red neuronal, diferenciando los distintos tipos de capas. Adaptada de Calvo, D.[Imagen], 2017, Diego Calvo (https://www.diegocalvo.es/wp-content/uploads/2017/07/neural-network.png)

Una de las arquitecturas más comunes en las redes neuronales es la conocida como red *feed-forward*, en la cual las conexiones de la red solo se pueden realizar de la capa i a la capa $i+1$. Para describirlas se usa una secuencia de enteros que indican la cantidad de nodos en cada capa. Por ejemplo, la red en la figura 2.1.5 es una red *feed-forward* y se representaría como 5,4,4,3, la capa de salida tiene tantas neuronas como etiquetas de clases, una neurona para cada salida potencial.

2.1.3. Pesos, sesgo y regularización

Los pesos (mayormente conocidos por su traducción al inglés, *weights*) son valores reales asociados a cada característica de una clase dada. Estos transforman los datos de entrada conforme estos avanzan en su procesamiento dentro de las capas ocultas. Al inicio del capítulo se mencionó que las entradas de las neuronas se multiplican por un factor que funge como la fuerza sináptica entre las mismas, este factor son los pesos de la red. Por otro lado, el sesgo (*bias*, por su traducción en inglés) es un valor real que ajusta la función de activación usada por las capas de la red con el propósito de reducir la varianza de las entradas y, por ende, introducir cierta flexibilidad a la red para obtener una mejor generalización en los resultados.

Estos dos parámetros son usados para que la salida sea una predicción Y (en el caso de un clasificador, dicha predicción es si el objeto analizado pertenece a cierta clase o no) calculada de la forma:

$$Y = \sum(weights * inputs) + bias \tag{2.1.1}$$

$$Y = \sum_{i=0}^{n}(w_i * x_i) + bias \tag{2.1.2}$$

donde *weights* es la matriz de pesos de la red e *input* es el vector de entradas de ésta. Existen distintas técnicas de inicialización de estos valores; en el caso de los pesos una de las más usadas es inicializar la matriz con valores aleatorios en una muestra gaussiana con una media y desviación estándar cercana a 0. Con el *bias* es una práctica común inicializarlo con un valor de 0 para lograr que la red active todas las capas en las primeras iteraciones del cálculo.

Al momento de entrenar un modelo de red, los valores de los pesos producen un margen de error, conocido como sobreajuste (*overfitting*, por su traducción al inglés) en el cual el mismo modelo se ajusta a los datos de entrada para una cierta salida y por tanto no se logra una buena generalización del problema. Para evitar este fenómeno, se emplean técnicas de regularización entre las que se encuentran la regularización L2 y L1, aumento automático de los datos y *dropout*.

2.1.4. Funciones de pérdida

Dado que las redes neuronales son algoritmos usados para la optimización de cálculos a problemas complejos, es necesario contar con una función objetivo con la finalidad de minimizar o maximizar el resultado. En las redes neuronales se busca una función donde las posibles soluciones factibles mapeen a una distribución uniforme dada una estimación de máxima similitud entre la clase predicha por la red y la etiqueta verdadera asociada al objeto. Existen distintos tipos de funciones de pérdida, aunque en problemas de clasificación es muy usada la función de entropía cruzada (*cross-entropy*, por su traducción al inglés) que usa una base logarítmica.

2.1.5. Backpropagation

El algoritmo de *Backpropagation* es en la actualidad uno de los algoritmos más importantes en el desarrollo de las redes neuronales, ya que con él se pueden optimizar y obtener mejores resultados. Dicho algoritmo consiste en dos fases, la fase de *forward pass* y la fase de *backward pass*. En la fase de *forward pass* las entradas son propagadas a través de la red y se obtienen las predicciones de las salidas, mientras que en la fase de *backward pass* se calcula el gradiente de la función de costo y este mismo se aplica en toda la red usando la regla de la cadena para actualizar los pesos de ésta. Para poder aplicar este algoritmo, la función de activación debe ser diferenciable ya que debemos calcular las derivadas parciales del error respecto a un peso $w_{i,j}$, el resultado de la función de pérdida E, la salida σ_j y la salida de la red net_j, siguiendo la siguiente fórmula:

$$\frac{\partial E}{\partial w_{i,j}} = \frac{\partial E}{\partial \sigma_j} \frac{\partial \sigma_j}{\partial net_j} \frac{\partial net_j}{\partial w_{i,j}} \tag{2.1.3}$$

Una vez calculado este resultado, los pesos de la red se actualizan con el nuevo valor encontrado para cada entrada.

2.2. Redes neuronales convolucionales

Las redes neuronales convolucionales son casi iguales a las redes neuronales convencionales con la diferencia de que en las redes convolucionales sus entradas

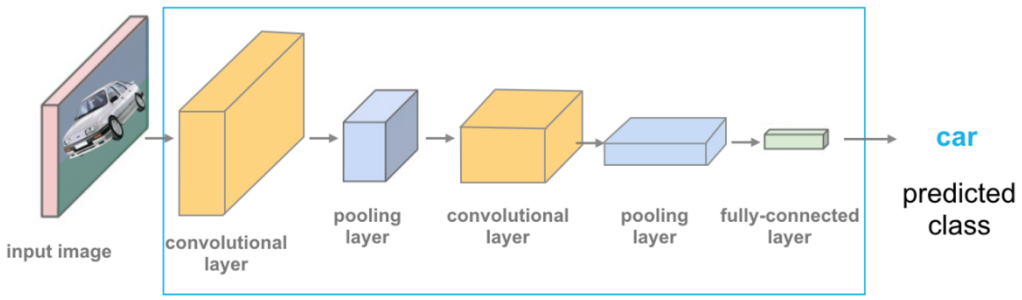


Figura 2.2.1: Ejemplo de una arquitectura de red neuronal convolucional. Adaptada de Camacho, C.[Imagen],2018,Cezaneec,(https://cezannec.github.io/Convolutional_Neural_Networks/)

son imágenes y que su arquitectura difiere en ciertos aspectos. Lo que hace la convolución es tomar dos matrices, obtener el producto punto de ambas y sumar elemento a elemento.

Convencionalmente, las neuronas de entrada a la red están conectadas a una neurona de otra capa anterior (a las capas donde se encuentran estas neuronas se les llama capas completamente conectadas, o *fully-connected* por su traducción al inglés) lo que nos daría capas cuyos pesos sean números demasiado grandes; por ejemplo, una imagen sencilla de $20 \times 20 \times 3$ (20 pixeles de alto, 20 de ancho y 3 canales de color) en una capa *fully-connected* tendría 1200 pesos, que en un inicio no sería problemático. El verdadero inconveniente es cuando se trabaja con imágenes de una resolución más grande.

2.2.1. Núcleos (*kernels*)

En redes convolucionales se define como núcleo (o *kernels*) a una matriz cuadrada de $N \times N$ menor en tamaño de dimensiones a la imagen de entrada de la red; este se desliza de izquierda a derecha y de arriba hacia abajo. En cada coordenada (x, y) de la imagen, el núcleo se centra en dicha coordenada, analiza los vecinos más cercanos a la misma y realiza la operación de convolución para obtener un único valor en esa región que es almacenado por la red, ver figura 2.2.2. Los pasos realizados en las convoluciones son los siguientes:

1. Seleccionar una coordenada (x, y) en la imagen.
2. Posicionar el centro del núcleo en dicha coordenada.
3. Realizar el producto punto entre la región de la imagen acotada por el núcleo

y el mismo.

4. Sumar cada valor obtenido del producto punto.
5. Almacenar en la misma coordenada el valor de la suma del paso 4.

2.2.2. Arquitectura

Las redes convolucionales, al recibir una imagen como entrada, aprovechan esto para formar su arquitectura. Las capas que forman a la red son tridimensionales: ancho, alto y profundidad (cantidad de canales de color en la imagen) y por esta misma razón, las capas no están completamente conectadas unas a otras. Sin embargo, en la capa final si se usa una capa completamente conectada porque la imagen se reduce a un solo vector con el fin de obtener la predicción de la clase, tal y como se muestra en la figura 2.2.1. Existen varios tipos de capas para armar una red convolucional, pero las cuatro principales son: capas de convolución, capas de activación (estas se omiten de los diagramas de las CNN dado que se implican por sí mismas su existencia), capas de pooling y capas *fully-connected*.

Las capas convolucionales (también descritas como CONV layers) son las capas de mayor importancia, sus parámetros que reciben son un numero K finito de filtros (es decir, *kernels*) cada uno con dimensiones ya determinadas. La profundidad en la capa de entrada es el número de canales de color en la imagen y en las capas posteriores se determina por la cantidad de filtros aplicados en las capas anteriores. Esto es porque en las primeras capas la red aprende de los kernels a determinar cierta característica especifica mientras que en las capas más profundas se determinan características de alto nivel: partes del cuerpo humano, partes de un animal, etc. Como se mencionó anteriormente, no todas las neuronas en estas capas están conectadas a la entrada. En vez de eso, se elige una región de la entrada y al tamaño de esta se conoce como campo receptivo (*receptive field*).

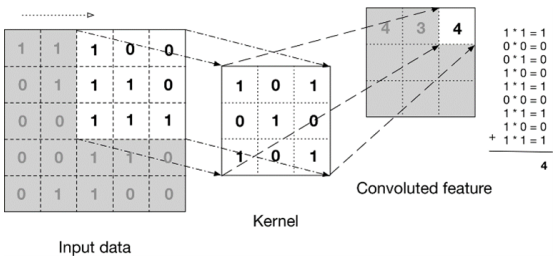


Figura 2.2.2: Ejemplo de una operación de convolución. Adaptada de Batista, D.[Imagen],2018, davidbatista (http://www.davidsbatista.net/blog/2018/03/31/SentenceClassificationConvNets/)

Para propagar la salida de cada capa a la siguiente se deben determinar tres parámetros: profundidad (*depth*), el "paso" (*stride*) y el rellenado de ceros (*zero-padding*). El *depth* determina la cantidad de neuronas conectadas a una región determinada de la entrada; el *stride* determina cuanto movimiento tendrá el núcleo sobre la imagen en cada iteración y el *zero-padding* es la cantidad de rellenado de ceros aplicado a la salida con motivo de mantener un volumen constante. Teniendo estos tres parámetros se tiene que asegurar que la siguiente ecuación sea un entero:

$$((W \check{} F + 2P)/S) + 1 \tag{2.2.1}$$

donde W es el volumen de la entrada, F es el tamaño del campo receptivo, S es el *stride* y P es la cantidad de zero-padding realizado. En caso de no asegurarse que esta ecuación de como resultado un entero, el stride utilizado no cumple que las neuronas sean capaces de encajar simétricamente al volumen de entrada.

Las capas de activación en su mayoría son capas cuya única función es realizar la evaluación de la salida de las capas de convolución en la función de activación ReLu, por esto mismo a estas capas es común denominarlas de esta forma.

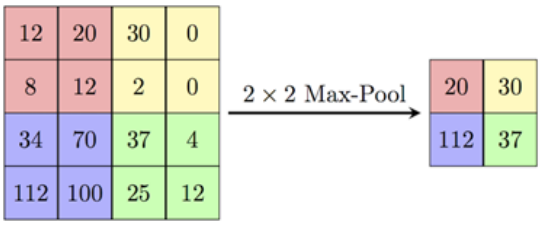


Figura 2.2.3: Ejemplo de aplicar max-pooling a una región de la imagen. Adapatada de FirelordPhoenix, 2018, computersciencewiki (https://computersciencewiki.org/index.php/File:MaxpoolSample2.png)

La razón de la existencia de las capas de *pooling* (denotadas por la palabra en inglés *pool*) se debe a que estas son utilizadas para reducir el tamaño de entrada entre las capas convolucionales permitiendo así reducir la cantidad de parámetros y por ende la cantidad de cálculo realizado, así mismo ayudan a prevenir el *overfitting*. En las capas de *pooling* se encuentran dos variantes: capas de *max-pooling* y capas de *average pooling*. La primera regresa el valor máximo de la porción analizada por el *kernel* en la capa de convolución mientras que la segunda obtiene el promedio entre todos los valores de la región analizada. De parámetros recibidos por capa se usa el tamaño del campo receptivo F y un *stride*, en su mayoría se usa un *max-pooling* de 2×2, como se muestra en la figura 2.2.3

Por último, están las capas *fully-connected* (FC). Estas siempre se encuentran al final de la red y son usadas para obtener el puntaje de clasificación final. Así mismo, existen otras capas que ayudan a la regularización: capas de *batch-normalization* (BN) y capas de *dropout*. Las primeras efectúan una normalización a los datos de entrada haciendo que reduzca de manera efectiva la cantidad de épocas de entrenamiento, sin embargo, tienen la desventaja de aumentar el tiempo de ejecución por época. El segundo tipo de capa, de *dropout*, regulariza los datos previniendo el *overfitting*, ya que dada una probabilidad p se desconectan neuronas de forma aleatoria de la capa anterior a la siguiente.

2.3. Redes convolucionales siamesas

Como su nombre lo indica, este tipo de redes neuronales consisten en dos redes idénticas, cada una con sus entradas correspondientes, que trabajan de manera paralela para al final comparar las similitudes entre ambas entradas. Este algoritmo fue introducido en 1994 por Bromley, J. (1993) con la intención de comparar dos firmas escritas y que dicha red predijera si ambas son originales o falsas, esto se logró al usar dos subredes basadas en la arquitectura de Time Delay Neural Network para al final comparar las salidas de ambas redes mediante la distancia coseno.

Como se mencionó anteriormente, la arquitectura de estos modelos se basa en tener dos redes iguales trabajando a la par para después unir la salida de dichas redes en una función comparativa, que puede ser distancia euclidiana, distancia L2 u otra. (ver figura 2.3.1).

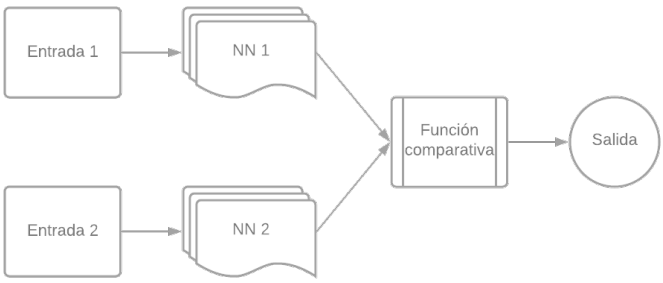


Figura 2.3.1: Arquitectura básica de una red neuronal siamesa

Antes de realizar cualquier entrenamiento a un modelo de este tipo, es necesario aplicar un preprocesamiento al conjunto de datos, el cual consiste en crear pares tanto de entradas afines como de entradas diferentes. A dichos pares se les deben de asignar etiquetas $y = 1$ para

aquellas entradas afines y en caso contrario $y = 0$ para entradas diferentes.

Existen diversas aplicaciones para este tipo de modelos que van desde el campo del procesamiento de audio hasta el biológico pasando por análisis de imágenes, desarrollo de software, química y farmacología, entre otros. Koch, G. (2015) realizó uno de los trabajos más notables usando este tipo de modelo junto a otra técnica conocida como *one-shot learning*; el modelo desarrollado consiste en una red convolucional siamesa a la cual se le pasa un solo ejemplo de imagen de cada clase y con esto obtiene una predicción precisa en la tarea de clasificar alfabetos escritos a mano. Dicho trabajo ha dado hincapié a otros trabajos de clasificación de imágenes usando de manera similar las técnicas anteriormente descritas.

Para las redes siamesas se usa una función de pérdida adaptada para diferenciar de manera eficiente las imágenes de entrada; esta función se conoce como *contrastive loss function* Hadsell, R. (2006) y es dada por la siguiente fórmula:

$$Y * D^2 + (1 - Y) * max(margen - D, 0) \tag{2.3.1}$$

donde:

- Y es el valor de la etiqueta para la entrada de la red, si es 1 significa que ambas imágenes son de la misma clase y en caso contrario es 0.
- D es el valor de la distancia euclídea que hay entre las salidas de las redes siamesas.
- La función max toma el máximo valor entre 0 y un margen m menos la distancia.

Como funciones comparativas de las salidas de los gemelos de la red, es muy común el uso de funciones de "distancia", como es el caso de las siguientes:

- Distancia Manhattan (también conocida como distancia $L1$):

$$L_1 = \sum_{i=1}^{n} |p_i - q_i| \tag{2.3.2}$$

- Distancia euclidiana (también conocida como distancia $L2$):

$$L_2 = \sum_{i=1}^{n} \sqrt{p_i^2} \tag{2.3.3}$$

- Distancia Chi-cuadrada:

$$X^2 = \frac{1}{2} \sum_{i=1}^{n} \frac{(x_i - y_i)^2}{(x_i + y_i)} \tag{2.3.4}$$

2.4. One shot learning

Uno de los grandes problemas a los que se enfrentan los modelos de redes neuronales es que estos necesitan grandes cantidades de datos para las fases de entrenamiento y verificación, por lo que estos son costosos en cuanto a recursos de cómputo. Por esta misma razón, Li,F. (2003) desarrolló en el año 2003 la técnica conocida como one shot learning (en español, aprendizaje de una sola toma); la técnica está basada en la capacidad humana de aprender las características de un objeto con solo ver un ejemplo de él, ver figura 2.4.1.

One shot learning funciona a través de un marco bayesiano el cual calcula una probabilidad que, dada la forma y apariencia de un objeto, este mismo pertenezca a cierta categoría específica. Para realizar esta tarea, se necesita de tres cosas:

- Un modelo que extraiga las características más importantes de las clases de objetos a clasificar.
- Una suposición correcta de la distribución de los datos.
- La forma de clasificación de los datos. Actualmente, esta técnica es muy usada en conjunto con otros modelos de redes neuronales, como puede ser redes neuronales siamesas, para así obtener de forma más eficaz la similitud entre dos imágenes dadas.

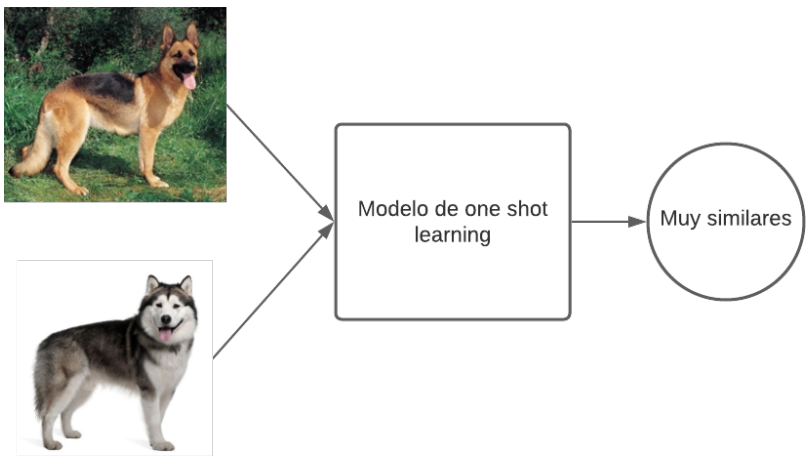


Figura 2.4.1: Ejemplo de cómo se puede utilizar one shot learning a la par de redes neuronales siamesas para obtener una probabilidad de que ambas imágenes pertenezcan a la misma categoría.

Capítulo 3

Estado del arte

La tarea de inspeccionar y detectar fallas en aisladores eléctricos con imágenes aéreas tomadas por UAVs (*unmanned aerial vehicles*) y técnicas de *deep learning* ha dado origen a diversas investigaciones a lo largo de los años, cada una aportando avances significativos a la automatización de dicha tarea. En la mayoría de dichas investigaciones se divide a la tarea en tres subtareas: la primera consiste en extraer de las imágenes la región donde se encuentra el aislador, la segunda tarea en segmentar en su totalidad el aislador del fondo de la imagen y la última tarea es la detección de fallas a esta segmentación, como se ilustra en la figura 3.0.1.

Uno de los trabajos con más impacto en este campo fue el desarrollado por Wang,Y. (2018) el cual desarrollo modelos de redes convolucionales distintos en arquitectura para las tres tareas demostrando buenos resultados con una exactitud superior al 99 %. Para la detección del aislador se usó una red con arquitectura Faster R-CNN la cual tiene la función de clasificar y localizar las cadenas de aisladores en las líneas de transmisión para después realizar una segmentación de estos en una red convolucional.

Al final, la salida de la red anterior es pasada a un clasificador que determina si dicho aislador esta quemado. Al igual, Li,X. (2020) hace uso de una Faster R-CNN, pero introduce a la misma un preprocesamiento de las imágenes de aisladores con la red *ResNeXt-101* para extraer las características a aprender por la red y así obtener una mejor detección de los aisladores y con los resultados de esta red se pasa la información a una red de tipo U-Net (parecida a la usada por Sampedro, C. (2019)) con el propósito de detectar fallas en las cadenas de aisladores logrando resultados en la precisión y recall de 91,9 % y 95,7 % respectivamente. Por su parte Chen, J. (2019) realizó un método de detección de fallas en aisladores a partir de una red convolucional completa de segundo orden (*SOFCN*, por sus siglas en inglés) que mejora la complejidad computacional de otros métodos de *deep learning*, además de ser capaz de suprimir de manera efectiva el ruido de fondo lo que ayuda a mejorar la precisión de la red. Es en Tao, X. (2020) que se describen las mejores técnicas de aumento de datos (como transformaciones afines, difuminado Gaussiano y transformaciones en el brillo y el contraste de las imágenes) para combatir la escasez de datos e imágenes de libre acceso disponibles.

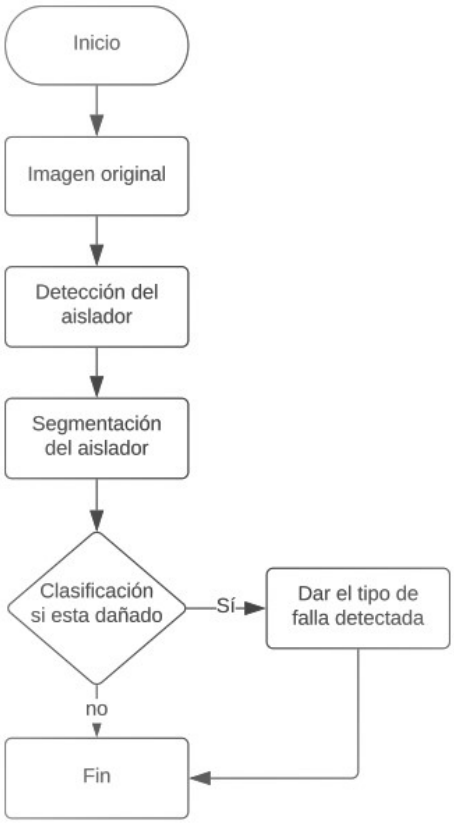


Figura 3.0.1: Estructura de la tarea de detección de fallas en aisladores.

Han, J. (2019) propone una red que parte de una modificación a la red *ResNet50* agregando tres ramas de capas convolucionales conectadas entre sí. Dicha red es funcional no solo para detección de una unica falla si no que es util para multiples tipos de fallas además de presentar un rendimiento en la precisión a la par a otros algoritmos como lo es la red *YOLOv3*.

Otro acercamiento a esta tarea es presentado por Bai, R. (2018) donde se hace

uso de redes tipo *Spatial Pyramid Pooling* networks con técnicas de transferencia de aprendizaje dado el pequeño conjunto de datos con el que se trabajó; dicha red arrojó resultados de precisión de 94,44 % y $F1 - score$ de 0.8333.

Por último, Sampedro, C. (2019) desarrollaron modelos de distintas arquitecturas en redes neuronales para las tres subtareas: para la tarea de detectar y segmentar el aislador se usó una arquitectura llamada *Up-Net* que es una red neuronal convolucional totalmente conectada la cual tiene una configuración tipo *encoding* basada en la arquitectura de *VGG16* que consisten en una combinación de 3×3 capas convolucionales con un *stride* de 1 y activación $ReLU$ seguidas de capas de *max-pooling* de 2×2 con *stride* de 2 al igual que una configuración *decoding* casi simétrica a la red anterior agregando capas de upsampling para aumentar la resolución del mapa de características extraído por la red, dicha operación es seguida por una capa convolucional de 2×2 con el fin de recortar a la mitad el número de canales. La última capa de la red es una capa convolucional de 1×1 usada para reducir el número de características y así dar con una predicción en la etiqueta de salida (ver figura 3.0.3). En el caso de la tarea para la clasificación de si el aislador contiene fallas o tiene discos faltantes, Sampedro realizó dos modelos de redes convolucionales: el primero se trata de una red convolucional cuya arquitectura son 10 capas, 5 de las cuales son capas convolucionales, 3 son capas de *max-pooling* y una capa *fully-connected* para terminar en una única salida, esto con la intención de detectar y clasificar si una cadena de aisladores tiene discos faltantes; mientras que el segundo modelo se trata de una red convolucional siamesa que tiene el objetivo de detectar si un único disco aislador presenta a nivel pixel fallas o ensuciamiento causado por factores naturales. Dicha red esta basada en la arquitectura de *VGG16* hasta la capa 3 donde pasa una única capa convolucional de 1×1 con el objetivo de reducir el volumen de canales para posteriormente pasar a una capa *fully-connected* (ver figura 3.0.3, para ambos modelos).

En cuanto a este trabajo es importante recalcar el resultado obtenido por Sampedro en el modelo de red siamés en contraparte al mejor modelo de red convolucional creado hasta el momento de publicación de su investigación, como se aprecia en la figura 3.0.2

Model arch.	TNR (%)	TPR (%)	G-mean (%)	ROC-AUC (%)	PR-AUC (%)	#param (M)
CNN-c4_vgg16	**86.69 ± 2.38**	86.83 ± 2.77	86.74 ± 1.73	91.84 ± 0.83	89.28 ± 2.16	2.82
SCNN-c4-L1_vgg16	86.56 ± 3.93	**90.42 ± 3.18**	**88.42 ± 1.64**	**93.67 ± 2.03**	**92.75 ± 1.70**	2.82

Figura 3.0.2: Resultados obtenidos por Sampedro, C. (2019).

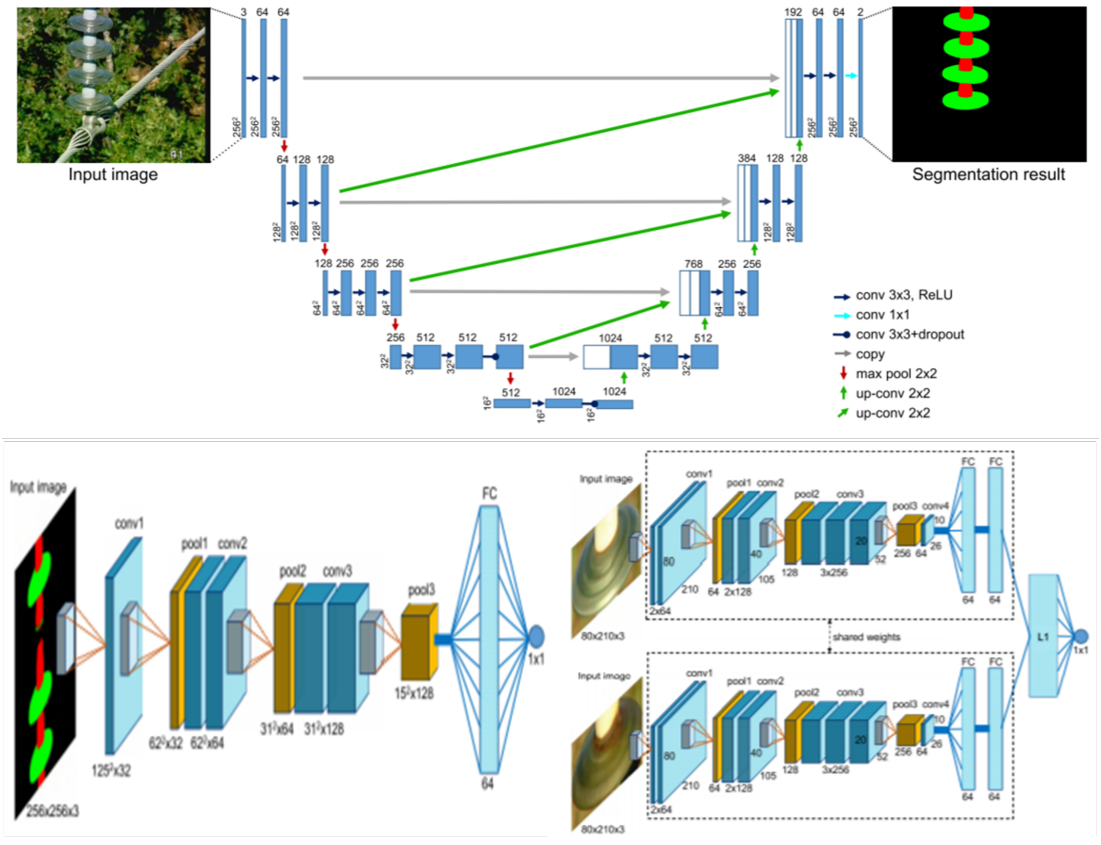


Figura 3.0.3: Modelos de redes convolucionales desarrollados por Sampedro, C. (2019). De arriba abajo e izquierda a derecha: 1) Modelo tipo Up-Net para la detección y segmentación de cadenas de aisladores. 2) Red neuronal convolucional usada para la detección de discos faltantes en la cadena de aisladores. 3) Red convolucional siamesa que tiene como objetivo clasificar si un disco aislante presenta grietas, suciedad u otro tipo de fallas.

Capítulo 4

Metodología de solución

4.1. Creación de un conjunto de datos a partir de imágenes licenciadas bajo *Creative Commons*

Uno de los mayores problemas en la detección de fallas en componentes de líneas de transmisión eléctrica usando técnicas de deep learning radica en la falta de imágenes de acceso libre en el internet; dado que la mayoría de las empresas que proveen servicio eléctrico protegen sus datos e imágenes con recelo. Por esta razón se llevó a cabo la creación de un conjunto de imágenes de aisladores eléctricos, las cuales fueron descargadas desde el portal web Flickr. En dicho portal, usuarios de distintos países suben todo tipo de fotografías las cuales son compartidas bajo distintos tipos de licencias de derechos de autor, una de las cuales se conoce como licencia *Creative Commons* (CC) la cual permite a terceros hacer uso de las imágenes del portal bajo ciertas condiciones.

El conjunto de imágenes consiste en cien imágenes de torres eléctricas y 40 imágenes de aisladores, además de un archivo de texto en donde se enlista la dirección URL de cada imagen y el nombre del usuario que subió dicha imagen a Flickr.

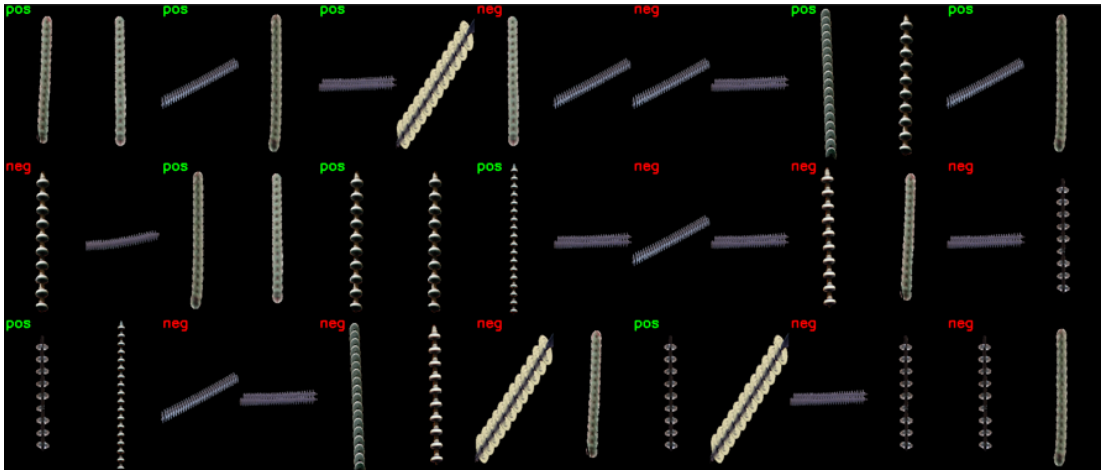


Figura 4.2.1: Visualización de los pares de imágenes con sus respectivas etiquetas generadas por el programa descrito en esta sección.

4.2. Obtención de imágenes sintéticas y el etiquetado manuel de ellas

Para el conjunto de entrenamiento se seleccionaron 138 imágenes segmentadas (es decir, el objeto de interés sobre un fondo negro) obtenidas de simuladores de vuelo de un VANT, de las cuales 69 imágenes son sin fallas y 69 imágenes con fallas, y para el conjunto de prueba se usaron 10 imágenes segmentadas obtenidas de internet (5 sin fallas, 5 con fallas). A dichas imágenes se les asignó mediante su nombre un identificador: un 0 si dicha imagen es de una cadena sin fallas y un 1 en caso contrario.

Como la red usada es siamesa, entonces se deben generar pares de imágenes que serán la entrada a la red; dicho proceso de creación de los pares de imágenes se realizo con un programa hecho en *Python* con las librerías de *Numpy* y *OpenCV*, el cual toma de parámetros las imágenes y la lista de etiquetas de cada una de las imágenes y selecciona una imágen que empareja a otra imágen del conjunto, si ambas imágenes pertenecen a la misma clase, entonces el programa asigna a dicho par la etiqueta de “ *"positive”*. En caso contrario, es decir que las imágenes sean de clases distintas se le asigna la etiqueta *“negative”*.

El programa regresa dos arreglos, uno con los pares de imágenes generados y el otro arreglo contiene la etiqueta asignada a cada par como se muestra en la figura 4.2.1.

4.3. Descripción del modelo de red convolucional siamesa

La red utilizada se basá en la red siamesa propuesta por Sampedro, C. (2019), la cual, a su vez, se basa en la arquitectura de VGG16. Solo se describirá un "gemelo" de la red siamesa dado que ambas redes son exactamente iguales.

Las imágenes de entrada a la red estan fijadas a tener un tamaño de 300×300 píxeles las cuales son pasadas en un primer paso a una capa de aumento de datos la cual realiza un ajuste aleatorio de contraste a las imágenes con el fin de variar el conjunto de datos.

A continuación, las imágenes pasan a tres bloques de dos capas convolucionales cada uno siguiendo una sucesión en la cantidad de filtros de 32, 64 y 128 respectivamente. Seguidamente se pasa a un bloque de tres capas covolucionales con un tamaño de filtro de 256 y a otro con dos capas de 512 filtros. Entre cada uno de estos bloques se hace uso de una capa de max-pooling con el fin de reducir el volumen de salida entre cada bloque.

Cada capa convolucional usa un tamaño de núcleo de 3×3, *stride* de 1×1 y activación tipo $ReLU$, mientras que las capas de *max-pooling* usan un tamaño de núcleo y *stride* de 2×2.

Terminando las capas convolucionales, se realiza un aplanamiento y se pasa este volumen a dos capas densas de 1024 unidades cada una con activación sigmoidal.

Ambos gemelos se unen en una capa tipo Lambda que calcula la distancia L_1 entre ambas salidas. La predicción final se calcula utilizando una capa densa de solo una unidad con activación sigmoidal.

Como función de pérdida se usa la función de *Contrastive Loss* iniciando con un margen de 1. La red final se ilustra en la figura 4.3.1

4.4. Métricas de evaluación

Para evaluar el rendimiento de la red se usarán las métricas de evaluación para redes convolucionales calculadas por la librería de *scikit-learn*. Estas son:

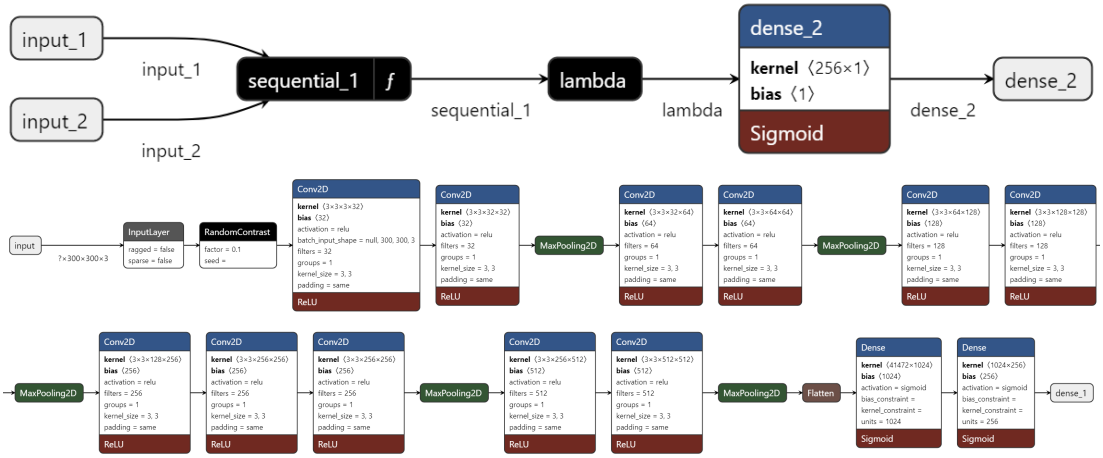


Figura 4.3.1: Visualización del modelo de red usando Netron https://github.com/lutzroeder/netron. En la parte superior es el modelo completo mientras en la parte inferior se presenta solamente un gemelo.

- Exactitud:

$$\frac{tp + tn}{tp + fp + tn + fn} \tag{4.4.1}$$

- Precisión:

$$\frac{tp}{tp + fp} \tag{4.4.2}$$

- Recuerdo:

$$\frac{tp}{tp + fn} \tag{4.4.3}$$

donde las siglas tp significas *true positive* (verdadero positivo, en inglés), tn es *true negative* (verdadero negativo), fp es *false positive* (verdadero positivo) y fn es *false negative* (falso negativo).

Capítulo 5

Resultados

En este proyecto se realizaron varias pruebas para asegurar que se obtuvieron los mejores resultados al ejecutar la red convolucional. Enseguida se describe las actividades llevadas a cabo para este fin.

5.1. Analisis del desempeño de la red siamesa.

Para combatir la escacez de datos disponibles, lo primero con lo que se tiene que experimentar son las técnicas de aumento de datos. Como plantea Tao, X. (2020), las técnicas que presentan mejores resultados son las transformaciones afines y ajustes en el brillo y contraste de la imagen; por lo que se probaron las transformaciones aleatorias de rotación, volteado y zoom, al igual que ajustes aleatorios al contraste.

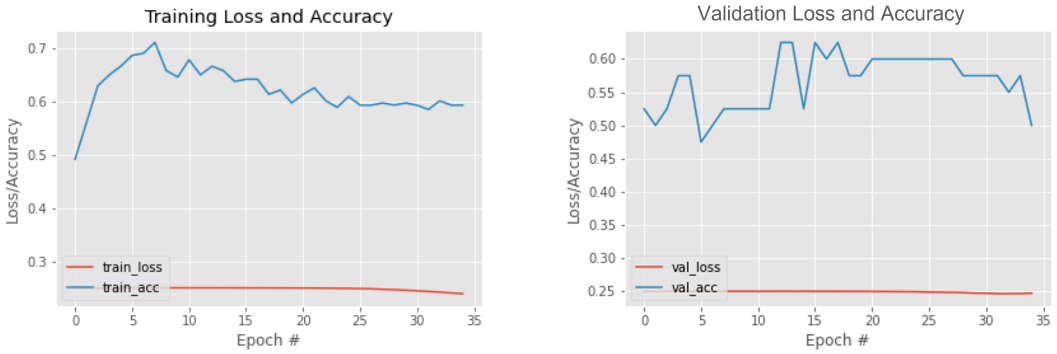
5.1.1. Pruebas sin aumento de datos

En primer lugar, se analizó el desempeño de la red sin ningún tipo de técnica de aumento de datos, esto con el fin de tener una base sobre el compartamiento de la misma.

Para dicha prueba, se entrenó el modelo con 118 ejemplos de aisladores capturados en el simulador empleando un *learning rate* de $1e^{-5}$ durante 35 épocas, validandose con otros 20 ejemplos distintos de aisladores del simulador.

En la figura 5.1.1 se muestra el comportamiento del modelo en las 35 épocas de

entrenamiento y validación. Se puede notar que en la mejor época de entrenamiento se logro obtener una exactitud cercana al 72 % sin embargo, en la validación solo se logró obtener una exactitud cercana al 62 % en su mejor caso.



(a) Entrenamiento **(b)** Validación

Figura 5.1.1: Gráfico que muestra los valores de exactitud y pérdida de la red sin hacer uso de técnicas de aumento de datos.

Esto demuestra que la información contenida en las imágenes con las que se cuenta es insuficiente, por tanto es necesario el uso de las técnicas de aumento de datos.

5.1.2. Pruebas con distintas capas de aumento de datos

Como se menciona anteriormente, se probaron distintas técnicas de aumento de datos a las imágenes, que de acuerdo con Tao, X. (2020) son:

- Rotación aleatoria.
- Giro horizontal aleatorio.
- Zoom aleatorio.
- Ajuste aleatorio al contraste.

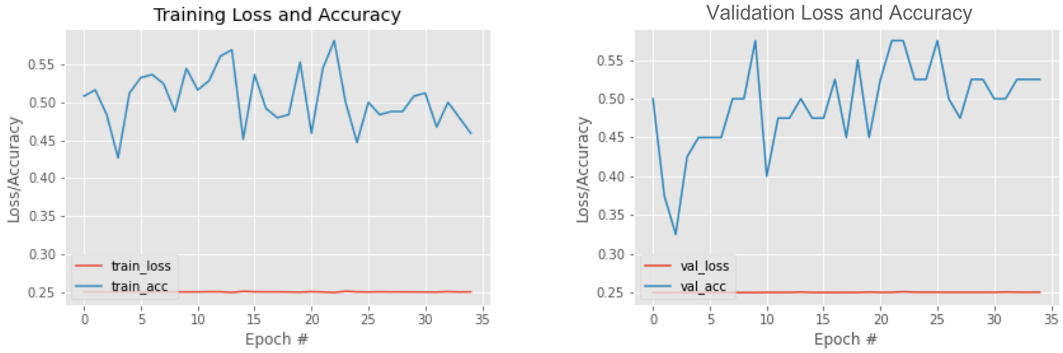
Para cada una de estas pruebas,se entrenó el modelo con capas de aumento de datos halladas en las líbrerias de *Tensorflow* las cuales se ejecutan en cada época generando distintos resultados.

Todos los entrenamientos se hicieron con 35 épocas y con un *learning rate* que comienza con un valor de $1e^{-5}$ y se optimiza usando el algoritmo *Adam*. Asimismo se hace uso de las 128 imágenes de aisladores obtenidas del simulador para el entrenamiento validandose sobre 20 imágenes igualmente del simulador.

5.1.2.1. Rotación aleatoria

En esta prueba se usa la capa llamada *RandomRotation* aplicando una rotación aleatoria a las imágenes bajo un factor de 0.10, es decir, se rotan las imágenes en sentido horario y anti-horario un 10 %.

Como demuestra la figura 5.1.2, la técnica de rotación aleatoria a las imágenes no mejoró los resultados en ninguna de las dos etapas, por lo que se descartó su uso para este proyecto.



(a) Entrenamiento **(b)** Validación

Figura 5.1.2: Gráfico que muestra los valores de exactitud y pérdida de la red con el uso de la rotación aleatoria.

5.1.2.2. Giro horizontal

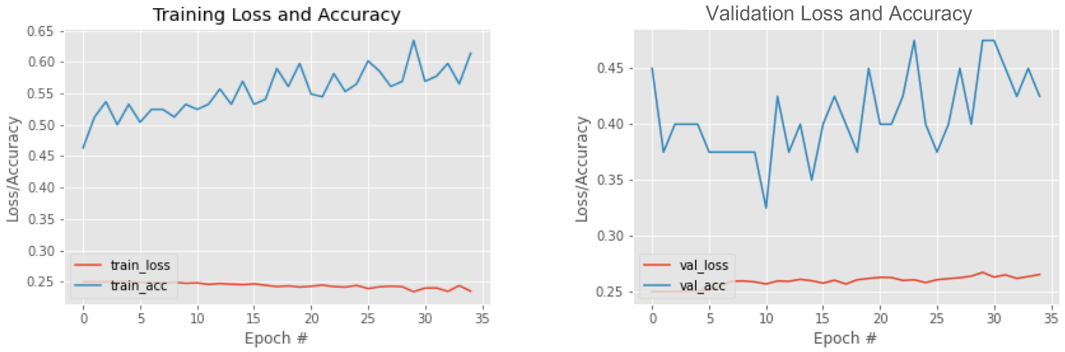
Para esta prueba, se usa la capa *RandomFlip* la cual aplica un giro horizontal aleatorio a las imágenes.

Los resultados (figura 5.1.3) obtenidos en el entrenamiento mejoraron considerablemente; sin embargo la validación empeoró al mostrar grandes saltos en los valores de exactitud y un incremento en el valor de pérdida aunque mínimo, no es deseable. Por tanto, también se descartó el uso de esta técnica.

5.1.2.3. Zoom aleatorio

La misma situación ocurre con la capa de *RandomZoom*, la cual realiza un zoom aleatorio a las imágenes en un factor de 10 %.

En la figura 5.1.4 se nota que se obtuvo una mejora significativa tanto en el entrenamiento como en la validación, haciendo de está técnica una buena solución



(a) Entrenamiento **(b)** Validación

Figura 5.1.3: Gráfico que muestra los valores de exactitud y pérdida de la red con el uso del giro aleatorio.

(a) Entrenamiento **(b)** Validación

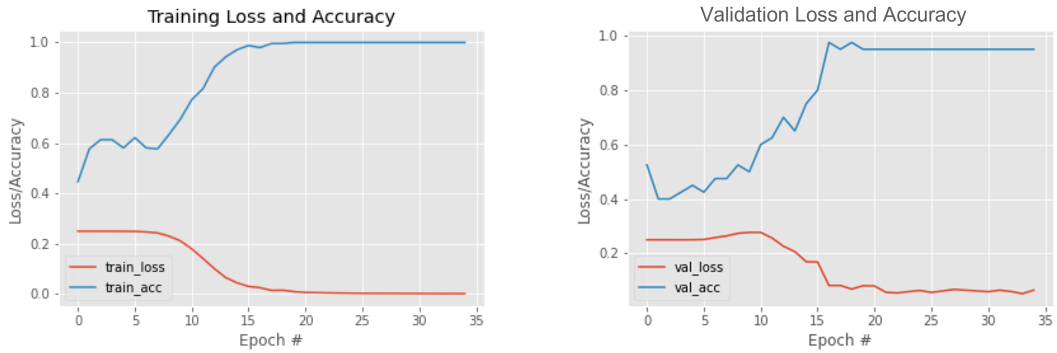
Figura 5.1.4: Gráfico que muestra los valores de exactitud y pérdida de la red con el uso de zoom aleatorio.

al problema de falta de imágenes, sin embargo, la exactitud en el conjunto de validación se mantuvo en un valor de 82 %.

5.1.2.4. Contraste aleatorio

Por último, se realizó la prueba con la capa de *RandomContrast* que efectúa en la imágenes un ajuste al contraste de manera aleatoria usando un factor del 10 %.

Esta técnica mostró gran comportamiento en el entrenamiento como en la validación, obteniendo puntajes en la exactitud de 100 % y 97 % además de obtener valores de pérdida menores a 0.05 puntos en ambas instancias como se muestra en la figura 5.1.5. Por tanto, se optó que el modelo de red incluyera este tipo de capa de aumento de datos.



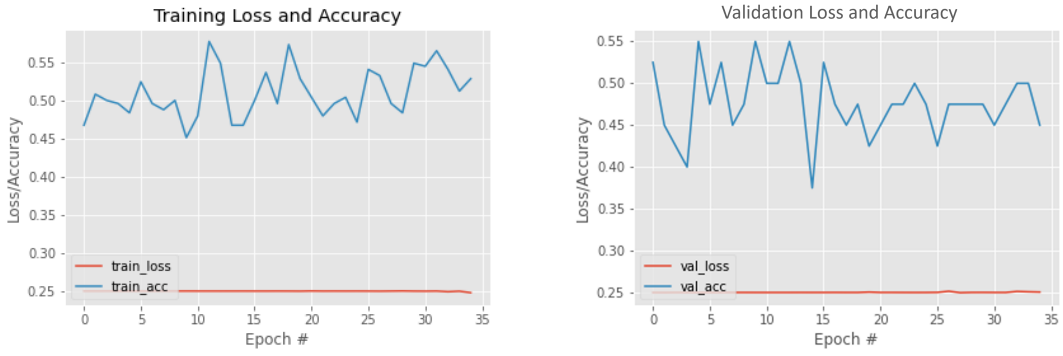
(a) Entrenamiento **(b)** Validación

Figura 5.1.5: Gráfico que muestra los valores de exactitud y pérdida de la red con el uso del contraste aleatorio.

5.1.2.5. Combinaciones entre las técnicas de aumento de datos

Del mismo modo, se realizaron pruebas con las distintas combinaciones de las técnicas anteriormente mencionadas, empezando por aquellas combinaciones que involucraron la técnica de contraste aleatorio.

En la figura 5.1.6 se observa que la combinación entre rotación aleatoria y contraste aleatoria no mejora el resultado obtenido si solo se utiliza el contraste aleatorio, por lo que esta y las demás combinaciones que involucren estas técnicas quedan descartadas.

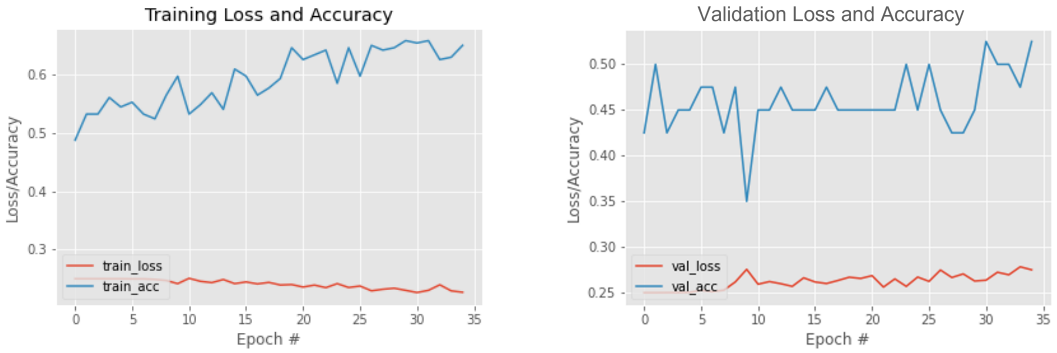


(a) Entrenamiento **(b)** Validación

Figura 5.1.6: Gráfico que muestra los valores de exactitud y pérdida de la red con el uso de rotación aleatoria y contraste aleatorio.

De igual forma, la combinación de volteado y contraste aleatorio (figura 5.1.7) no ofrece resultados que superen a los resultados obtenidos con la técnica de contraste

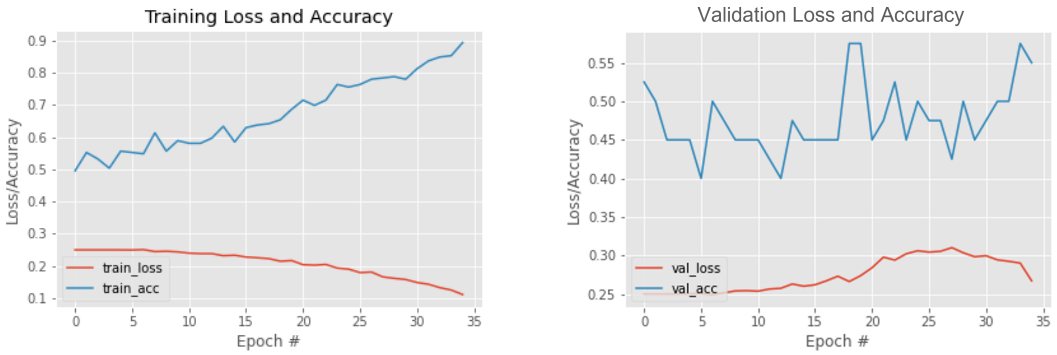
aleatorio.



(a) Entrenamiento **(b)** Validación

Figura 5.1.7: Gráfico que muestra los valores de exactitud y pérdida de la red con el uso de volteado y contraste aleatorio.

En el caso de la combinación entre las técnicas de contraste y zoom aleatorio, se encontró un comportamiento errático, a causa de una validación con saltos importantes en la exactitud y un incremento sustancial en la pérdida entre las épocas 20 y 35, expuesto en la figura 5.1.8.



(a) Entrenamiento **(b)** Validación

Figura 5.1.8: Gráfico que muestra los valores de exactitud y pérdida de la red con el uso de contraste y zoom aleatorio.

Por lo tanto, ninguna posible combinación da mejores resultados que si solo se utiliza la técnica de contraste aleatorio, por lo cual solo se utilizó esta técnica en las siguientes pruebas.

5.2. Validación cruzada

Una consecuecia directa de tener tan pocos datos para realizar el entrenamiento y validación de un modelo clasificador radica que el resultado obtenido en la evaluación de las métricas cambien drásticamente dependiendo de que conjunto de datos de entrenamiento se elija usar para el entrenamiento.

Por tal motivo, se llevo a cabo un proceso de validación en k-iteraciones, que en el caso de este proyecto $k = 5$ para obtener el mejor promedio de cada métrica de evaluación. Cada fracción (referida en la tabla 5.3.2 como *split*) es usada 3 veces para entrenar y validar el modelo.

Cada partición esta conformada por 128 imágenes para el conjunto de entrenamiento y 20 imágenes para el conjunto de validación, ambos con imágenes del simulador.

La tabla 5.3.2 muestra los resultados de cada métrica evaluada más la desviación estandar en las 3 ejecuciones de cada partición. Estos datos corresponden unicamente a la etapa de validación.

Cuadro 5.2.1: Resultados obtenidos en la validación cruzada.

	Accuracy (%)	Precision (%)	Recall (%)	Loss
Split 1	87.5±5.40	88.2±6.14	96.7±2.36	0.120±0.045
Split 2	83.3±1.18	93.9±4.82	85.0±7.07	0.145±0.003
Split 3	87.5±3.54	91.3±6.36	86.7±2.36	0.106±0.027
Split 4	90.8±12.96	91.7±11.78	93.3±9.42	0.093±0.108
Split 5	80.8±20.45	84.4±22.00	93.3±6.23	0.131±0.087
Promedio	85.6±7.69	89.9±10.22	91.0±5.49	0.119±0.042

El *split* 1 muestra un buen desempeño en el recuerdo, sin ambargo las demás métricas obtienen un resultado muy por debajo del esperado; igualmente, el *split* 2 y 3 presentan una precisión mayor a los demás *splits* pero sacrificando recuerdo y exactitud.

Es en el *split* 4 que se obtienen los mejores resultados en todas las categorías, aun cuando la desviación estandar de estas es la mayor por una mala ejecución en la última instancia. Por ultimo, el *split* 5 repite el mismo patrón mostrado por los *splits* 1,2 y 3.

5.3. Validación con imágenes reales

Como última prueba al modelo y para validar la hipotesis de este proyecto se realizó una validación cualitativa que consiste en una evaluación de la red con los pesos aprendidos con el split 3 de la validación cruzada.

Cuadro 5.3.1: Resultados obtenidos en la validación con imágenes reales (Split 3).

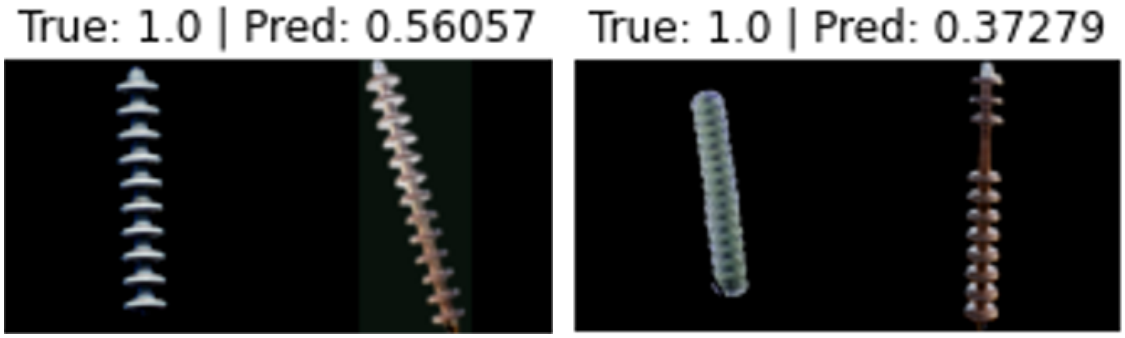
Split 3	Accuracy(%)	Precision(%)	Recall(%)	Loss
Total	60.0	62.5	50.0	0.329

Cuadro 5.3.2: Resultados obtenidos en la validación con imágenes reales (Split 4).

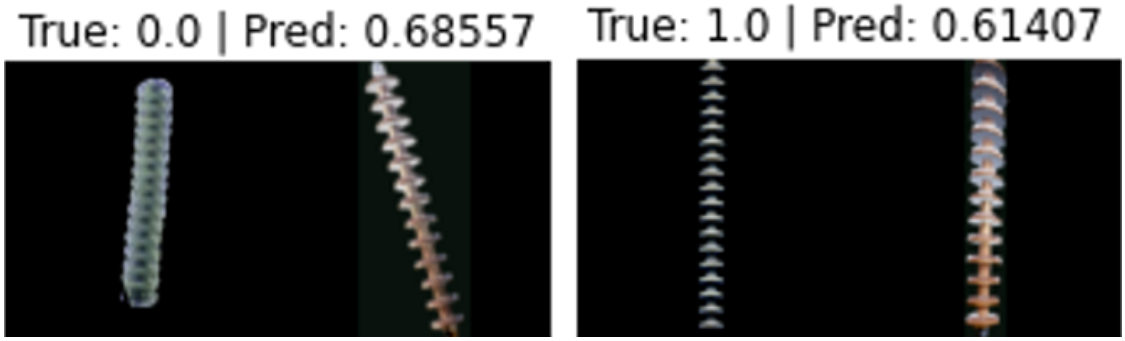
Split 4	Accuracy(%)	Precision(%)	Recall(%)	Loss
Total	67.5	66.7	70.0	0.235

En las figuras 5.3.1 y 5.3.2 se presenta los resultados de esta evaluación, la primera figura muestra aquellas predicciones malas mientras la segunda muestra las mejores predicciones hechas por el modelo.

Cabe destacar que la información que regresa el modelo es un puntaje que representa que tan similares son las dos imágenes comparadas entre si. Entonces, si la predicción del modelo es un valor cercano a la etiqueta real del par de imágenes, este par se considera que esta bien clasificado.

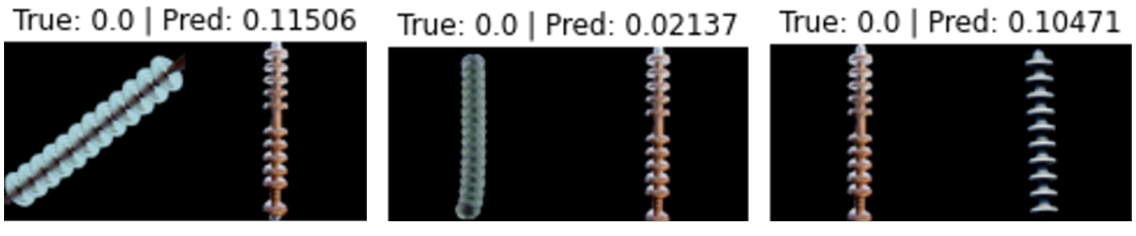


(a) 1 **(b)** 2

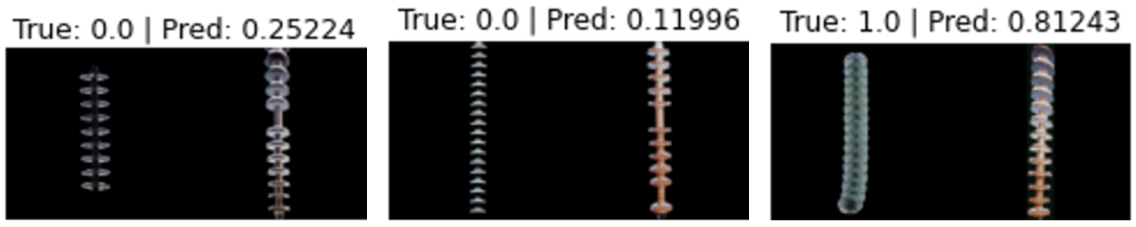


(c) 3 **(d)** 4

Figura 5.3.1: Imágenes que fueron clasificadas incorrectamente.



(a) 1 **(b)** 2 **(c)** 3



(d) 4 **(e)** 5 **(f)** 6

Figura 5.3.2: Imágenes que fueron clasificadas correctamente.

Capítulo 6

Conclusión

En conclusión, en este proyecto se logró:

- La creación exitosa de un conjunto de imágenes de aisladores, tanto simulados como de entornos reales.
- El modelamiento y ejecución de una red convolucional siamesa con los mejores parámetros posibles dada la escasa cantidad de imágenes de aisladores.
- La validación de la red usando diversas técnicas para esta tarea.

Gracias a la validación cualitativa, se puede ratificar parcialmente la hipótesis de este proyecto, siendo posible mejorar el rendimiento del modelo con una mayor variedad en las imágenes de aisladores simulados (forma, tipo de material y perspectiva del aislador).

Bibliografía

Bai, R. , Cao, H. , Y. W. D. C. (2018). Insulator fault recognition based on spatial pyramid pooling networks with transfer learning (match 2018). *2018 3rd International Conference on Advanced Robotics and Mechatronics (ICARM)*, pages 824–828.

Bromley, J. , Bentz, J. W. , B. G. L. M. S. S. (1993). Signature verification using a "siamese" time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, 07(04):669–688.

Chen, J. , Xu, X. , D. (2019). Fault detection of insulators using second-order fully convolutional network model. *Mathematical Problems in Engineering*, 2019.

Hadsell, R. , Chopra, S. , L. (2006). Dimensionality reduction by learning an invariant mapping. *IEEE Computer Society on Computer Vision and Pattern Recogniton (CVPR'06)*, 2:1735–1742.

Han, J. , Yang, Z. , Z. C. L. L. H. X. X. W. C. (2019). A method of insulator faults detection in aerial images for high-voltage transmission lines inspection. *MDPI Applied Sciences*, 9.

Koch, G. , Zemel R. , S. (2015). Siamese neural networks for one-shot image recognition. *ICML Deep Learning Workshop.*, 2.

Li,F. , Fergus, R. , P. (2003). A bayesian approach to unsupervised one-shot learning of object categories. *IEEE Ninth International Conference on Computer Vision. 2*, pages 1134–1141.

Li,X. , Su, H. , L. (2020). Insulator defect recognition based on global detection and local segmentation. *IEEE Access*, 8:59934–59946.

Rosenblatt, F (1961). *Principles of neurodynamics. perceptrons and the theory of brain mechanisms.*

Sampedro, C. , Rodriguez-Vazquez, J. , R. C. C. (2019). Deep learning-based system for automatic recognition and diagnosis of electrical insulator strings. *IEEE Access*, 7:101283—-101308.

Tao, X. , Zhang, D. , W. L. Z. X. (2020). Detection of power line insulator defects using aerial images analyzed with convolutional neural networks. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 50(04):1486–1498.

Wang,Y. , Wang, J. , G. H. X. Z. Y. X. L. (2018). Detection and recognition for fault insulator based on deep learning. *11th International Congress on Image and Signal Processing, Biomedical Engineering and Informatics (CISP-BMEI)*, pages 1–6.

VOTOS DE APROBATORIOS

SECRETARIA EJECUTIVA DEL
INSTITUTO DE INVESTIGACIÓN EN CIENCIAS BÁSICAS APLICADAS
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. JOSÉ EDUARDO NUÑEZ ORTEGA** con número de matrícula **10003181** cuyo título es:

"Detección visual automática de cadenas de aisladores dañadas en líneas de transmisión a través de 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 6 de septiembre de 2023

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)**

VRRC/EAE

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:2023-09-06 16:01:05 | Firmante

EdY4z272mJMs3VBTevL6sdEyoTejo4kicKOlQrA1S4RE+OWtbU8MWhrePf5f+ULNaOK8LajtMyN/iDv0uMn7GDbwW+BuOOYDSqtWBP26F2M1NkwRka5j4dITOYySxYDZ16rnpnAH6Kn1V9jI97+H3UWuF7hD6LoenFBoDkZVsQUthETXeBpuEB6f1gWYTR9YuSQcjw7jxSTZvHSXpYof1D3XlRSrzFcJBXVPPY5yXbiWvAm9gjmesIAM3bXBcZjHQ32TBOhwq03rnaw24+fM6KnQYL15xWa9TizL2WMSA6PfxnsgmGJ2bDvCRg3SgbeDVE8sO34QZ7QgIsrJHinvRQ==

LORENA DIAZ GONZALEZ | Fecha:2023-09-06 16:46:26 | Firmante

KyxsQMCG/VFebA05K5627AY7Za8OGiq7BUZEov5KYj2zOcgLp4rejn2dJANtCUyb5Ye1Mxka2ozFdMdVAie/gugQCIp2mFvH1FF3nd9/1c4V/JJXLukOyw5bP7l6p+kBJRsrNWpZL5vL4+jQR1t2ebaK/r6r7qFIXXEaKKm9UvhMAvfvPKdRKXRyUCzsWZG4onUB1TXuA1SPZYABccSykDouXDycfVJHWW3Cqs+qOiPyA3nfS0OscbiWDqTEVnodBVBvrpbxQadBPO4SmLRtQec2GI8m4/Rg5qnqswcv+cbxR68BK5s62eQJvMORlkJAdoUufe84PJMaVayK9qBzDw==

JORGE ALBERTO FUENTES PACHECO | Fecha:2023-09-07 11:18:41 | Firmante

zxnqqQuBrBfDFr2UoTlBfDyeQAltuFwnsJvm0Z+/Dimwd+7k2jqhKevQ2G9UTS6ZG3aIOX+XX1208HXdpxKBOzv+3tSTzUHHEuHxRgWhsBXTo7MhhE5sjEomoyqjaBBXnzDNquwMJ/fDrOCy+ZFqorVTSeGHa8lGdBC3EJFGMMG5eagIt7WTPLagB80OLRsB8Jat2Ogns+mnQYhtmopWgFOvuhCsGF3VZTyvRSMLTf/XKmvrCAj0gVD0M5oJOI22OqFCOhPSsyG3WTSNlX+isvCnY+yFUm9FUNx+ZaKiPbRW8fkcRiC5uYBbWyjPyjcHMiHzcYetVSRxvb2oRnxTzQ==

JUAN MANUEL RENDON MANCHA | Fecha:2023-09-07 15:31:42 | Firmante

oX9zu2DwqLw695hCnB++LDTkwzjWgzYcWLwixkYEdcxoPAj3hefdoPZV6WD8q8uLVha3TFkKn0drAcPJQbqaK6IM4dxWzEWa9qyLJTAynjNCn/wXBBEFurh7GKZU3PTKHKzn1nLWpaIp57aCv3MfWs/NyxkASf0XGVYdwrJJIiCYD0DxPKJSBq4RDK28nkMqXHRjq0Stv6NQB2JA7nUmrJwxAt9WVa2dPMjlz4UYtNF50Fo6s3eZWDjEt5fP/IkmbyD0i3KQYnmhWQAPYvOFN5hOX4+3XVTSAfYReWNspbAj4iyMMyeNlLdqxHh4wKnWIk28HlIcrDwpNOQab1DcEw==

AUDBERTO REYES ROSAS | Fecha:2023-09-07 23:08:15 | Firmante

MetweGWYCG+WCrxQzRYBSpXxyz9DbMaRkVwvvWFadwm+QpzMgWMPrNyQ/xkjzujTtvSJLpnbfSSu55mZpoAy7NWTnc5vRxfq/4azVc9KpmPP3J/g3vROOo86gmXbad838Z42dhwRX5h0DInoqBkTBk+4xE1wvN1xCwdUZ3qXx1fqK932m9iYe/DjAmlwX3VbBjFeLRPsK+OZcXPN4QM7QsbVBUVJzDqRqOFGJ8TWBjj1G7/85EWEneFpoSDdO3gkv5ovEchxU7rYXc7wdDjVSe51/rdhN4mHwREksITUK/QClcHIz50uNSxdLweR5s+hZT9OjfA3LKjykwLFvy0cBQ==

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

OjngTLdUo

https://efirma.uaem.mx/noRepudio/N6gKwF58QuEe6nYeazNAlepTV5l8EvQa