RESUMEN TRABAJO DE

GRADO

AUTOR(ES): NOMBRES Y APELLIDOS COMPLETOS

NOMBRE(S): Luis Carlos APELLIDOS: Torres Vega

FACULTAD: Ingeniería

PLAN DE ESTUDIOS: Ingeniería electrónica

DIRECTOR:

NOMBRE(S): Byron APELLIDOS: Medina Delgado

CODIRECTOR:

NOMBRE(S): Ghiordy Ferney APELLIDOS: Contreras Contreras

TÍTULO DEL TRABAJO DE GRADO: Sistema De Corrección De Ruido En Imágenes De Conducción Asistida Implementando Redes Neuronales De Tipo Autoencoder

Resumen:

Esta investigación tuvo como finalidad desarrollar un sistema que permitió corregir el ruido presente en imágenes de conducción asistida, esto por medio del uso de redes neuronales de tipo autoencoder. Para dar cumplimiento a los objetivos se planteó una metodología estructurada en 5 fases, lo cual permitió llevar a cabo actividades orientadas a cada uno de los objetivos específicos establecidos. Como resultado se pudo apreciar una mejora en la calidad de las imágenes procesadas, esto en función de métricas objetivas de calidad como la proporción máxima de señal a ruido (PSNR) o la medida del índice de similitud estructural (SSIM), incrementando hasta 17 dB el valor de PSNR respecto al obtenido por la imagen con ruido y obteniendo imágenes de 99% de similitud estructural, lo anterior en las pruebas realizadas con ruido Sal & Pimienta; en cuanto a los resultados con ruido Gaussiano, se pudieron obtener ganancias de 9 dB en el PSNR y hasta un 83% de similitud estructural, respuesta que se debe a la forma como el ruido Gaussiano se manifiesta en la imagen, dificultando su modelado y correspondiente procesamiento.

PALABRAS CLAVES: Corrección de ruido, deep learning, visión

CARACTERÍSTICAS:

PÁGINAS: 118 PLANOS:____ ILUSTRACIONES: _____ CD ROOM: ____

Sistema De Corrección De Ruido En Imágenes De Conducción Asistida Implementando Redes Neuronales De Tipo Autoencoder

Luis Carlos Torres Vega

Universidad Francisco De Paula Santander

Facultad De Ingeniería

Ingeniería Electrónica

San José de Cúcuta, Colombia

2023

Sistema De Corrección De Ruido En Imágenes De Conducción Asistida Implementando Redes Neuronales De Tipo Autoencoder

Luis Carlos Torres Vega

Director: PhD. Byron Medina Delgado

Codirector: Ghiordy Ferney Contreras Contreras

Trabajo de grado presentado para optar por el título de Ingeniero Electrónico

Universidad Francisco De Paula Santander

Facultad De Ingeniería

Ingeniería Electrónica

San José de Cúcuta, Colombia

2023

Universidad Francisco de Paula Santander
Vigilada Mineducación
NIT. 890500622 - 6

ACTA DE SUSTENTACIÓN DE UN TRABAJO DE GRADO

Fecha: CÚCUTA, 01 DE MARZO DE 2023

Hora: 10:00

Lugar: SALON SC 301

Plan de Estudios: INGENIERÍA ELECTRÓNICA

Título del trabajo de grado: "SISTEMA DE CORRECCIÓN DE RUIDO EN IMÁGENES DE CONDUCCIÓN ASISTIDA IMPLEMENTANDO REDES NEURONALES DE TIPO AUTOENCODER".

Jurados: IE PhD. FRANCISCO MORENO GARCIA
IE MSc. ANDRES EDUARDO PAEZ PEÑA

Director: IE. PhD. BYRON MEDINA DELGADO

Codirector: IE. GHIORDY FERNEY CONTRERAS CONTRERAS

Nombre del Estudiante:	**Código:**	**Calificación:**	
		Número	Letra
LUIS CARLOS TORRES VEGA	1161637	5,0	Cinco,cero

LAUREADA

FRANCISCO MORENO GARCIA

ANDRES EDUARDO PAEZ PEÑA

SERGIO SEPÚLVEDA MORA
Coordinador Comité Curricular
Ingeniería Electrónica

CARTA DE AUTORIZACIÓN DE LOS AUTORES PARA LA CONSULTA, LA REPRODUCCIÓN PARCIAL O TOTAL Y LA PUBLICACIÓN ELECTRÓNICA DEL TEXTO COMPLETO

Cúcuta,

Señores
BIBLIOTECA EDUARDO COTE LAMUS
Ciudad

Cordial saludo:

Yo, **Luis Carlos Torres Vega**, identificado con la C.C. Nº **1005073089**, autor de la tesis y/o trabajo de grado titulado **SISTEMA DE CORRECCIÓN DE RUIDO EN IMÁGENES DE CONDUCCIÓN ASISTIDA IMPLEMENTANDO REDES NEURONALES DE TIPO AUTOENCODER** presentado y aprobado en el año **2023** como requisito para optar al título de **ingeniero electrónico**; **SÍ** autorizo a la biblioteca de la Universidad Francisco de Paula Santander, Eduardo Cote Lamus, para que con fines académicos, muestre a la comunidad en general a la producción intelectual de esta institución educativa, a través de la visibilidad de su contenido de la siguiente manera:

- los usuarios pueden consultar el contenido de este trabajo de grado en la página web de la Biblioteca Eduardo Cote Lamus y en las redes de información del país y el exterior, con las cuales tenga convenio la Universidad Francisco de Paula Santander.

- Permita la consulta, la reproducción, a los usuarios interesados en el contenido de este trabajo, para todos los usos que tengan finalidad académica, ya sea en formato CD-ROM o digital desde Internet, Intranet etc.; y en general para cualquier formato conocido o por conocer.

Lo anterior, de conformidad con lo establecido en el artículo 30 de la ley 1982 y el artículo 11 de la decisión andina 351 de 1993, que establece que "**los derechos morales del trabajo son propiedad de los autores"**, los cuales son irrenunciables, imprescriptibles, inembargables e inalienables.

Luis Carlos Torres

Luis Carlos Torres Vega
C.C. 1005073089
Código.

Agradecimientos

El autor expresa sus agradecimientos a:

- Mis orientadores, el Doctor Byron Medina Delgado y el candidato a Magíster Ghiordy Ferney Contreras Contreras, su enseñanza, confianza, motivación y apoyo fue vital durante este proceso de formación personal y profesional.
- Al Grupo de Investigación en Tecnología, Innovación y Sociedad (GITecInSo) y el Grupo de Investigación y Desarrollo en Electrónica y Telecomunicaciones (GIDET), por brindarme el espacio y las herramientas necesarias durante este proceso de investigación.
- A mis compañeros de estudio que me acompañaron durante esta etapa de mi vida, brindándome espacios de crecimiento profesional y personal.
- A toda mi familia, especialmente a mi madre por todo el sacrificio hecho para que cumpliese mis metas, sin sus consejos nada de esto habría sido posible; a mi padre y a mis hermanos por el apoyo brindado en cada etapa de mi vida.

Contenido

Pág.

Introducción **17**

1. Descripción del Problema **19**

1.1. Planteamiento del Problema 19

1.2. Justificación 22

1.2.1. Beneficios Tecnológicos 23

1.2.2. Beneficios Científicos 23

1.3. Alcance 24

1.3.1. Resultados Esperados 24

1.3.2. Impacto Social 24

1.4. Limitaciones 25

1.4.1. Hardware 25

1.4.2. Software 25

1.5. Delimitaciones 26

1.5.1. Conceptual 26

1.5.2. Espacial 26

1.5.3. Temporal 26

1.6. Objetivos 27

1.6.1. General 27

1.6.2. Específicos 27

2. Marco Referencial **28**

2.1. Antecedentes 28

2.1.1. Deep Learning on Image Denoising: An Overview 29

2.1.2. Autoencoders Based Deep Learner for Image Denoising 29

2.1.3. Devdan: Deep Evolving Denoising Autoencoder 29

2.1.4. Llnet: A Deep Autoencoder Approach to Natural Low-Hight Image Enhancement 30

2.1.5. Underwater Color Restoration Using U-Net Denoising Autoencoder 30

2.2. Marco teórico 31

2.2.1. Conducción Asistida 31

2.2.2. Obtención de Información del Entorno 32

2.2.2.1. Radar. 32

2.2.2.2. Cámaras. 32

2.2.2.3. LASER Imaging Detection and Ranging (LiDAR). 33

2.2.2.4. Sensores de Proximidad (Ultrasonido). 33

2.2.3. Tipos de Imágenes y Métodos de Denoising 33

2.2.3.1. Eliminación de Ruido Blanco Aditivo. 34

2.2.3.2. Eliminación de Ruido en Imágenes Realmente Ruidosas. 34

2.2.3.3. Eliminación de Ruido a Ciegas. 34

2.2.3.4. Eliminación de Ruido Híbrido. 35

2.2.4. Redes Neuronales 35

2.2.5. Autoencoders 36

2.2.6. Métricas Aplicadas en el Procesamiento de Imágenes 37

2.2.6.1. Relación de Compresión (CR). 38

2.2.6.2. Mean Squared Error (MSE). 38
2.2.6.3. Peak Signal-to-Noise Ratio (PSNR). 39
2.2.6.4. Structural Similarity Index (SSIM). 39
2.3. Marco legal 40
2.3.1. Reglamentación en el Ámbito de la Conducción Asistida 41
2.3.2. Uso de Software Libre 42
2.3.3. Tratamiento de Datos 43
3. Metodología 45
3.1. Análisis 45
3.2. Diseño 46
3.3. Implementación 47
3.4. Evaluación 48
3.5. Divulgación 49
4. Resultados 50
4.1. Métodos de Denoising y Requerimientos Técnicos 50
4.1.1. Revisión del Estado del Arte 50
4.1.2. Identificación de Métodos de Denoising 51
4.1.2.1. Tipos de Ruido. 52
4.1.2.2. Métodos de Denoising. 54
4.1.3. Requerimientos de la Problemática 55
4.1.3.1. Costo Computacional. 55
4.1.3.2. Métricas. 58
4.2. Diseño del Algoritmo DCAEAD 60

4.2.1. Fuentes de Libre Acceso 60
4.2.1.1. Lenguaje de Programación. 60
4.2.1.2. Entorno de Desarrollo Integrado (IDE). 61
4.2.1.3. Dataset. 62
4.2.2. Asignación de Recursos 64
4.2.2.1. Selección del Lenguaje de Programación. 65
4.2.2.2. Selección del Entorno de Desarrollo Integrado. 69
4.2.2.3. Selección del Dataset. 69
4.2.3. Estructura del Software 69
4.2.3.1. Parámetros de la Red. 71
4.3. Implementación y Pruebas de Funcionamiento 72
4.3.1. Programación en Python 72
4.3.1.1. Lectura y Procesamiento de los Datos. 72
4.3.1.2. Insertar Ruido. 74
4.3.1.3. Modelo del Sistema DCAEAD. 75
4.3.1.4. Entrenamiendo del Sistema. 77
4.3.2. Ajuste del Software a los Requerimientos 77
4.3.3. Pruebas Unitarias 80
4.3.3.1. Pruebas con Ruido Gaussiano. 81
4.3.3.2. Pruebas con Ruido Salt & Pepper. 82
4.4. Evaluación del Sistema 84
4.4.1. Análisis Estadístico 85
4.4.1.1. Resultados con AWGN. 85

4.4.1.2. Resultados con Ruido Salt & Pepper. 87

4.4.1.3. Ganancia del Sistema en Función de PSNR y SSIM. 89

4.4.1.4. Contrastar Resultados. 91

4.4.2. Análisis de Costo Computacional 91

4.4.3. Retroalimentación 93

4.5. Divulgación de Resultados 94

5. Conclusiones 96

Referencias Bibliográficas 98

Anexos 106

Lista de Figuras

Pág.

Figura 1. Imagen tomada de dataset adicionando ruido de tipo Salt & Pepper a $\mathrm{Prob} = 0.202$. 20

Figura 2. Representación gráfica del proceso de reducción de ruido con autoencoders. Imagen tomada del dataset de prueba con adición de ruido AWGN a $\sigma = 0.3$. 21

Figura 3. Teoría necesaria para comprender el desarrollo y funcionamiento de esta investigación. 31

Figura 4. Breve historia de las redes neuronales. 36

Figura 5. Relación del PSNR con la calidad de la imagen. 39

Figura 6. Esquema de la metodología implementada. 45

Figura 7. Clasificación de las investigaciones revisadas en el estado del arte. 51

Figura 8. Representación gráfica de la función de activación ReLU. 56

Figura 9. Muestra de los datos aportados por cada dataset. 63

Figura 10. Diagrama de bloques correspondiente a la fase de entrenamiento. 70

Figura 11. Esquema representativo de la estructura de la red. 72

Figura 12. Representación gráfica de la probabilidad de insertar ruido Salt & Pepper. 75

Figura 13. Perdidas obtenidas para los entrenamientos realizados. 79

Figura 14. Pruebas realizadas con el modelo entrenado para corregir AWGN. 82

Figura 15. Pruebas realizadas con el modelo entrenado para corregir Salt & Pepper. 84

Figura 16. Curvas correspondientes a los valores promedio de PSNR obtenidos a partir de las imágenes estimadas y las imágenes con AWGN. 86

Figura 17. Curvas correspondientes a los valores promedio de SSIM obtenidos a partir de las imágenes estimadas y las imágenes con AWGN. 87

Figura 18. Curvas correspondientes a los valores promedio de PSNR obtenidos a partir de las imágenes estimadas y las imágenes con ruido Salt & Pepper. 88

Figura 19. Curvas correspondientes a los valores promedio de SSIM obtenidos a partir de las imágenes estimadas y las imágenes con ruido Salt & Pepper. 89

Figura 20. Ganancia del sistema DCAEAD en términos de PSNR. 90

Figura 21. Ganancia del sistema DCAEAD en términos de SSIM. 90

Figura 22. Tiempo de ejecución del sistema en función de la cantidad de datos a procesar. 92

Lista de Tablas

Pág.

Tabla 1. Descripción de los tipos de ruido más comunes. 53

Tabla 2. Recomendaciones para disminuir el costo computacional. 57

Tabla 3. Valores promedio de PSNR y SSIM correspondientes a la estimación respecto a la imagen ruidosa, para pruebas realizadas con el dataset Urban100. 59

Tabla 4. Valores promedios de PSNR y SSIM calculados a partir de las investigaciones revisadas. 59

Tabla 5. Comparativa de lenguajes de programación. 61

Tabla 6. Comparativa de los entornos de desarrollo integrado. 61

Tabla 7. Comparativa de los dataset de libre acceso disponibles. 62

Tabla 8. Cantidad de tiempo de video aportado por los dataset. 64

Tabla 9. Ponderación para los criterios de selección del lenguaje de programación. 66

Tabla 10. Ponderación de los lenguajes de programación según el nivel de abstracción. 66

Tabla 11. Ponderación de los lenguajes de programación según la manera de ejecución. 67

Tabla 12. Ponderación de los lenguajes de programación según el tipado. 67

Tabla 13. Ponderación de los lenguajes de programación según el paradigma de programación. 67

Tabla 14. Cálculo de la ponderación final para los lenguajes de programación. 68

Tabla 15. Parámetros seleccionados para el entrenamiento de la red. 78

Tabla 16. Equivalencias entre σ y Prob en función del PSNR producido por la imagen ruidosa respecto a la original. 80

Tabla 17. Evaluación de los resultados promedio obtenidos con base en los requerimientos. 91

Tabla 18. Resumen de participaciones en eventos de divulgación científica. 95

Lista de Anexos

Pág.

Anexo 1. Síntesis de la Revisión del Estado del Arte. 106

Anexo 2. Niveles de Automatización de la Conducción (SAE J3016). 111

Anexo 3. Comparación entre adición de ruido a imágenes normalizadas y no normalizadas. 112

Anexo 4. Valores promedio de PSNR y SSIM obtenidos a partir del modelo entrenado con AWGN. 113

Anexo 5. Valores promedio de PSNR y SSIM obtenidos a partir del modelo entrenado con Salt & Pepper. 114

Anexo 6. Certificado de participación en el evento IEEE ICA-ACCA 2022. 115

Anexo 7. Artículo publicado en IEEE Xplore producto de la participación en el evento IEEE ICA-ACCA 2022. 116

Anexo 8. Certificado de participación en la IX Semana Internacional de Ciencia, Tecnología e Innovación. 117

Resumen

Esta investigación planteó el desarrollo de un sistema de corrección de ruido en imágenes de conducción asistida implementando redes neuronales de tipo autoencoder. En este documento se muestra toda la información recolectada durante el proceso de desarrollo del sistema antes mencionado, con la finalidad de facilitar su comprensión se dividió el contenido en 5 capítulos, tal como se menciona a continuación:

El capítulo 1 aborda la descripción del problema, identificando de manera clara la problemática abordada, junto con la justificación de la investigación, de modo que permitió establecer un alcance, límites y delimitaciones del proyecto, para finalmente plantear los objetivos a cumplir.

El capítulo 2 aborda un análisis del campo de investigación, identificando los principales antecedentes, junto con el aporte y/o recomendación expresada por los autores; en segundo lugar, se plantea el marco teórico, donde se establece la teoría a tener en cuenta durante el desarrollo de esta investigación además de la normativa identificada en el marco legal.

El capítulo 3 plantea la metodología establecida para llevar a cabo el desarrollo de la investigación, estructurada en fases, donde cada una de estas se encuentra dirigida al cumplimiento de cada uno de los objetivos planteados, de modo que los resultados obtenidos reflejen lo estipulado en el capítulo 1 de este documento.

El capítulo 4 muestra los resultados concernientes a cada uno de los objetivos planteados, de modo que se muestre de manera clara el cumplimiento de cada uno de estos.

Finalmente, el capítulo 5 muestra las conclusiones de la investigación, resaltando los principales aspectos vistos durante el desarrollo del proyecto planteado.

Abstract

This research proposed the development of a noise correction system for assisted driving images by implementing autoencoder neural networks. This document contains all the information collected during the development process of the aforementioned system. To facilitate its understanding, the content was divided into 5 chapters, as mentioned below.

Chapter 1 deals with the description of the problem, clearly identifying the problem addressed, together with the justification of the research, to establish the scope, limits, and delimitations of the project, and finally establish the objectives to be met.

Chapter 2 deals with an analysis of the research field, identifying the main antecedents, together with the contribution and/or recommendation expressed by the authors; secondly, the theoretical framework is presented, where the theory to be taken into account during the development of this research is established, in addition to the regulations identified in the legal framework.

Chapter 3 presents the methodology established to carry out the development of the research, structured in phases, where each one of these is directed to the fulfillment of each one of the proposed objectives so that the results obtained reflect what was stipulated in Chapter 1 of this document.

Chapter 4 shows the results concerning each of the objectives, to clearly show the fulfillment of each one of them.

Finally, Chapter 5 shows the conclusions of the research, highlighting the main aspects seen during the development of the project.

Introducción

Un factor constante a lo largo de la historia de la humanidad es la dedicación dada a innovar todos los procesos involucrados en el diario vivir, con la finalidad de facilitar la realización de las actividades habituales. Inicialmente, esto condujo hacia a la evolución con la creación de las primeras herramientas y actualmente, este factor aún prevalece en la mentalidad humana (Benavides, 2004), buscando siempre alternativas que faciliten las acciones cotidianas. Es así como surge el concepto de sistemas inteligentes de transporte (ITS por sus siglas en inglés) (Florez, 2001), el cual busca mejorar la calidad y seguridad del transporte terrestre, puesto que comprende el medio más usado según datos otorgados por el ministerio de transporte (mintransporte) (Ministerio de transporte , 2021).

Uno de los métodos implementados actualmente en los sistemas inteligentes de transporte es la conducción asistida, permitiendo obtener información del exterior del vehículo para brindar mayor seguridad y comodidad al conductor, apoyándose en una variedad de sensores y módulos, obteniendo datos visuales por medio de cámaras (Leal, 2021). Un ejemplo práctico de conducción asistida son los asistentes de ángulo muerto (Blind Spot Detection BSD), el cual, por medio de imágenes obtenidas de los laterales y parte trasera del vehículo informa al conductor sobre la presencia de autos cercanos y de esta forma ayuda a prevenir posibles accidentes (Continental, 2020).

Durante la conducción, el cuerpo humano requiere de sus 5 sentidos, pero, el sentido de la visión resalta en esta tarea (Caparrós, 1999), pues a través de este se pueden identificar situaciones de riesgo y reaccionar a tiempo de prevenirlas, sin embargo, la tasa de accidentes viales es considerablemente alta (Organización mundial de la salud (OMS), 2017), esto a causa de distracciones presentes en el ambiente. Por esta razón los sistemas de conducción asistida

representan una ayuda sustancial al conductor, sin embargo, para hacer que estos sistemas sean funcionales se requiere la implementación de computer vision, la cual consiste en una disciplina científica que pretende emular el sentido humano de la visión por medio de la tecnología (Doulamis, Doulamis, Voulodimos, & Protopapadakis, 2018), permitiendo tomar imágenes, analizarlas e interpretarlas con la finalidad de obtener datos numéricos que puedan ser tratados por computadora (Szeliski, 2010). Durante este proceso es necesario tener en cuenta que una imagen capturada por una cámara normalmente presenta cierto grado de contaminación, debido a diferentes factores como las condiciones del ambiente, la incertidumbre presente en los dispositivos de captura de video (Contreras, Pabón, García, Rojas, & Arguello, 2021), entre otros. Dicha contaminación suele presentarse en forma de ruido y dificulta obtener la mayor cantidad de información a partir de la imagen dada.

Por otra parte, los autoencoders se entienden como un tipo de red neuronal artificial (RNA) que permite construir una señal de salida a partir de una señal de entrada por medio de un proceso de mapeo de las características principales de la señal con ruido (Bank, Koenigstein, & Giryes, 2021). Esto supone una gran ayuda en el campo del procesamiento digital de imágenes, puesto que contribuye a reducir el margen de error en la información obtenida, como se ha demostrado en diferentes investigaciones (Yapici & Akcayol, 2021) (Tian, Fei, Zheng, Xu, & Lin, 2020).

Teniendo en cuenta lo anterior, esta investigación planteó el desarrollo de un sistema de corrección de ruido en imágenes de conducción asistida, esto mediante la implementación de Deep learning y redes neuronales de tipo autoencoder, brindando una alternativa en el procesamiento digital de imágenes que contribuye a mejorar la calidad de los sistemas basados en este tipo de datos.

1. Descripción del Problema

Actualmente, los avances tecnológicos son cada vez más acelerados, buscando siempre un objetivo en común, facilitar la vida de los seres humanos (Benavides, 2004). Por esto que surgieron los sistemas de conducción asistida, los cuales, se basan en la obtención de imágenes del entorno para posteriormente procesarlas y obtener la información necesaria que permita realizar una toma de decisiones y contribuya al bienestar del conductor y los peatones (Talero, 2015). Por esta razón se implementan diferentes métodos que permitan disminuir el factor de ruido en la imagen y de esta forma obtener información más valiosa a partir de estas (Parmar & Patil, 2013). Debido a lo anterior, esta investigación se centró en desarrollar un sistema capaz de responder a la problemática planteada mediante la implementación de redes neuronales de tipo autoencoder.

1.1. Planteamiento del Problema

En el proceso de adquisición de imágenes del entorno se deben superar una gran cantidad de retos para poder obtener imágenes que permiten extraer información útil, uno de estos retos es el ruido presente en las imágenes, el cual impide o dificulta el correcto procesamiento de estos datos y por ende disminuye la eficiencia del sistema (Esqueda, 2002).

Dicho ruido se debe a diversas circunstancias, tanto de factores externos, como las condiciones ambientales o ruido blanco, o de factores internos, como los fallos del sistema de adquisición o del sistema óptico (Fernández, 1996), este último representa un inconveniente al momento de procesar la señal, puesto que no se puede modelar, lo cual limita el proceso de predecir su generación y plantear una forma de contrarrestarlo.

Como se explicó anteriormente, los sistemas de conducción asistida se basan en el procesamiento de las imágenes tomadas del entorno, por lo que se hace necesario que estas

cuenten con la menor interferencia o presencia de ruido posible, si se desea obtener un análisis más exacto. Sin embargo, los sistemas implementados para dicha labor no son del todo eficientes para evitar el ruido, produciendo que se obtengan resultados erróneos o imprecisos. Con la finalidad de ejemplificar lo anteriormente expuesto, se presenta en la Figura 1 una imagen aparentemente sin ruido tomada de uno de los datasets que se implementó para el desarrollo del sistema (Reddy S. , Mathew, Gomez, Rusinol, & Karatzas, 2020), a dicha imagen se le adiciona un nivel de ruido, en este caso en específico ruido de tipo Salt & Pepper, con la finalidad de evidenciar de esta forma la distorsión que se produce en la información.

Figura 1. Imagen tomada de dataset adicionando ruido de tipo Salt & Pepper a $\mathrm{Prob} = 0.202$.

Por lo anterior, se hace importante aumentar la calidad de las imágenes obtenidas por estos sistemas, esto mediante la implementación de autoencoders, los cuales permitieron reducir la tasa de ruido y por ende disminuir el margen de error en la información, corrigiendo las alteraciones presentes en la señal a causa de factores como el medio ambiente, exceso luz, entre otros.

Este proceso de reducción de ruido por medio de autoencoders se representa de manera gráfica en el esquema de la Figura 2, notando como a partir de una imagen de entrada con presencia de ruido se obtiene a la salida la imagen con mayor calidad, de la cual se puede obtener mayor información. Para conseguirlo, el autoencoder realiza internamente dos procesos, inicialmente, en el Enconder se comprime la entrada respaldando solo los valores de mayor

relevancia, representándose mediante la función $h = f(x)$, posteriormente, en el Decoder, se reconstruye la información recolectada previamente, expresándose como $r = g(h)$, con base en esto, el autoencoder se puede representar de la forma $d(f(x)) = r$, donde la salida r es similar a la entrada original x (Charte, 2021).

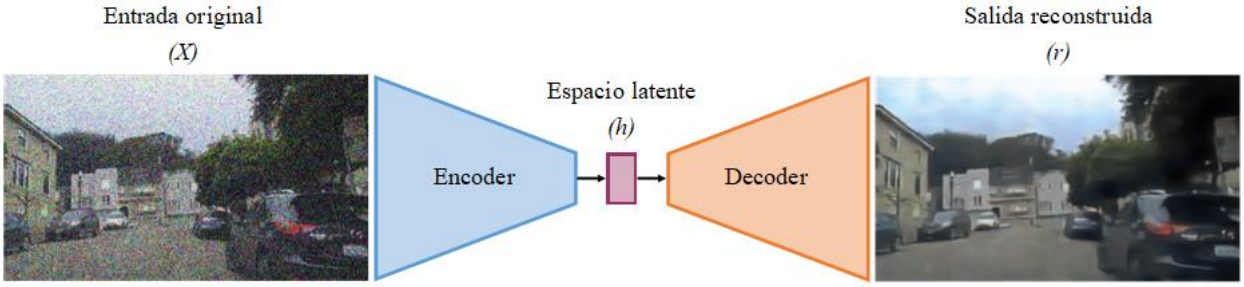


Figura 2. Representación gráfica del proceso de reducción de ruido con autoencoders. Imagen tomada del dataset de prueba *(Reddy S. , Mathew, Gomez, Rusinol, & Karatzas, 2020)* con adición de ruido AWGN a $\sigma = 0.3$.

Otro factor a tener en cuenta a la hora de procesar este tipo de imágenes es la corrección gamma, esto debido a la naturaleza de la investigación, puesto que las imágenes serán obtenidas al aire libre, se darán circunstancias donde es necesario corregir la luminancia, para el caso de aumentar o disminuir el factor gamma, dependiendo de la circunstancia en específico, dicha corrección gamma se encuentra representada por la ecuación (Poynton, 2012)

$$V_{out} = AV_{in}^{\gamma}, \tag{1}$$

donde A es una constante, V_{out} corresponde a la salida, V_{in} es la entrada y deben ser reales no negativos, mientras que el exponente γ para valores de $\gamma > 1$ produce sombras mas oscuras y para valores de $\gamma < 1$ produce que las zonas oscuras sean más claras (Poynton, 2012).

Teniendo en cuenta lo anteriormente expresado, se plantea el siguiente interrogante, ¿Cómo desarrollar un algoritmo basado en Deep learning capaz de compensar el ruido para apoyar sistemas autónomos de conducción?

1.2. Justificación

Esta investigación propuso el desarrollo de un sistema que permite la corrección de ruido en imágenes aplicando autoencoders, facilitando de esta forma la obtención de la información deseada. Dicho sistema se orientó específicamente en el campo de la conducción asistida de vehículos, donde las estadísticas de accidentalidad vial muestran una gran problemática de seguridad (Agencia Nacional de Seguridad Vial, s.f.).

Inicialmente, se introdujeron al mundo de la tecnología varios sistemas que contribuían a mejorar la seguridad vial, esto basándose en sistemas ópticos de captura de imagen y el procesamiento de estos datos para obtener información, permitiendo reconocer vehículos u objetos cercanos (Burguillos, 2016), entre otras aplicaciones. Sin embargo, dichos sistemas presentan fallas o se ven limitados debido al ruido presente en las imágenes, el cual, dificulta obtener la mayor información útil reduciendo así su eficiencia, es por esto que diferentes investigaciones se han centrado en buscar un método que permita corregir el ruido (Ashfahani, Pratama, Lughofer, & Ong, 2020) (Lv & Li, 2021) (Parmar & Patil, 2013), obteniendo soluciones válidas para este problema, donde los métodos implementados con autoencoders han demostrado tener una alta eficiencia (Rohrer, 2020), es por eso que este proyecto se centró en la aplicación de autoencoders para corregir el ruido presente en imágenes implementadas durante la conducción asistida.

1.2.1. Beneficios Tecnológicos

El procesamiento digital de señales permite a los dispositivos percibir el entorno, sin embargo, al corregir el ruido presente en estas imágenes se facilita la identificación de los factores que influyen en él, otorgando a los sistemas la capacidad de reconocer los diferentes parámetros y de esta forma contribuir en el ámbito de la conducción asistida, aumentando la seguridad vial (Burguillos, 2016).

Las técnicas de Machine learning brindan la posibilidad del aprendizaje automático a partir de un conjunto de datos de entrenamiento (dataset) (Reddy S. , Mathew, Gomez, Rusinol, & Karatzas, 2020), el cual, hace posible disponer de un sistema altamente eficaz que permita identificar las correspondientes situaciones y modificar parámetros de forma automática.

Actualmente, los softwares de código abierto han brindado una gran posibilidad para el desarrollo de investigaciones, puesto que contribuyen a reducir costos de desarrollo y otorgan su fácil acceso, permitiendo impulsar sistemas basados en estos sin la necesidad de un alto costo de inversión.

1.2.2. Beneficios Científicos

Los medios visuales están en la base de muchas de las investigaciones o trabajos que se realizan, como trabajos de laboratorio, videovigilancia, medicina, aeroespaciales, entre otros (Riveros, 2013) (Mérida, 2012), debido a que la obtención de información a partir de imágenes se hace más práctica, pero, en muchos casos este proceso se ve limitado debido al ruido presente en las imágenes (Esqueda, 2002), lo cual, impide su correcto análisis, por esto, se hace necesario buscar la forma de disminuir la cantidad de ruido y así aumentar la eficiencia en estos procesos.

La corrección de ruido implementando autoencoders otorga a los investigadores mayor libertad en el alcance de sus investigaciones, puesto que ayuda a obtener información más precisa de los datos, disminuyendo el margen de error presente.

1.3. Alcance

Esta investigación es de tipo aplicada tecnológica cuantitativa donde se buscaron estrategias para abordar un problema en específico, como lo es la corrección de ruido en imágenes utilizadas para la conducción asistida. Teniendo en cuenta lo anterior, esta investigación planteó un sistema que soluciona este problema, abriendo las puertas para su implementación en futuros trabajos investigativos e impulsando un impacto positivo en la vida cotidiana.

1.3.1. Resultados Esperados

Disminuir el ruido presente en las imágenes de conducción asistida por medio de sistemas basados en Deep learning y facilitar el análisis e interpretación de la información allí contenida.

Generar conciencia acerca de los métodos y herramientas computacionales para la reducción de ruido en imágenes.

Evaluar la exactitud de los resultados obtenidos con respecto a métricas existentes como la Proporción Máxima de Señal a Ruido (PSNR por sus siglas en inglés) o el Índice de Similitud Estructural (SSIM por sus siglas en inglés) (Huynh-Thu & Mohammed, 2008).

1.3.2. Impacto Social

Según informes brindados por la organización mundial de la salud (OMS), cada año mueren cerca de 1.3 millones de personas a nivel mundial a causa de accidentes viales, y entre 20 y 50 millones padecen traumatismos no mortales, estos traumatismos por accidentes de tránsito llegan a tal punto de ser considerados problema de salud pública a nivel mundial (Organización mundial de la salud (OMS), 2017).

Por lo anterior, se hace importante realizar investigaciones que permitan mejorar la seguridad vial y por ende ayuden a disminuir la tasa de accidentalidad, es así como surgió este proyecto, el cual, buscó generar un sistema capaz de corregir el ruido presente en imágenes usadas durante la conducción asistida, permitiendo de esta forma desarrollar tecnologías con mayor precisión que contribuyan a aumentar la seguridad no solo de los conductores, sino de los peatones, la infraestructura, y la población en general.

1.4. Limitaciones

Dado el tipo de problema abordado, se presentaron ciertas limitaciones de carácter externo que limitaron el alcance pretendido por la investigación, estas comprendieron recursos tecnológicos y aspectos materiales, como lo son el hardware y software al que se tuvo acceso.

1.4.1. Hardware

Para el desarrollo de este proyecto se contó con un computador Lenovo IdeaPad S145, con procesador CORE i7 de 8th generación a 1.8GHz, sin tarjeta gráfica, lo cual representa muchas limitantes al momento de trabajar con metadatos puesto que la capacidad de cómputo es bastante limitada, repercutiendo directamente en la complejidad computacional del proyecto.

1.4.2. Software

En lo concerniente al software se contó únicamente con servicios open source como el lenguaje de Python para la programación y codificación necesaria, mientras que para implementar el código se usaron entornos de libre acceso como PyCharm, VSCode o Colab, además de solo contar con datasets de libre acceso (Visual Data), lo cual dificultó el proceso de entrenamiento de las redes neuronales artificiales.

1.5. Delimitaciones

Las delimitaciones se plantearon con la finalidad de brindar mayor precisión y definir la extensión que abarca el tema de investigación, teniendo en cuenta aspectos geográficos y temporales, como también la profundidad que se deseaba adquirir en el tema, de esta forma se establecieron ciertos límites que permitieron conservar el tema central de estudio, siguiendo los lineamientos planteados en la metodología.

1.5.1. Conceptual

Se abordaron imágenes tomadas del entorno con fines de aplicación en conducción asistida. El sistema está diseñado para tomar imágenes de muestra en tiempo real y procesarlas, reduciendo el ruido presente en ellas y facilitando la obtención de los parámetros deseados, una vez se obtengan las imágenes limpias de ruido, estas se pueden implementar en gran cantidad de aplicaciones, como reconocimiento de peatones, reconocimiento de señales de tráfico, detección de obstáculos, entre otros, sin embargo, esta investigación no profundizará en dichas aplicaciones.

1.5.2. Espacial

Esta investigación se llevó a cabo en la Universidad Francisco de Paula Santander, en colaboración con el Grupo de Investigación y Desarrollo en Electrónica y Telecomunicaciones (GIDET) y el Grupo de Investigación en Tecnología, Innovación y Sociedad (GITecInSo).

1.5.3. Temporal

Inicialmente, se estimó que el desarrollo del proyecto tomaría veintiséis (26) semanas a media jornada laboral (20 horas/semana) después de aprobado el anteproyecto, sin embargo, este fue culminado en un plazo de diecisiete (17) semanas.

1.6. Objetivos

Teniendo clara la problemática que pretende afrontar esta investigación, las limitaciones inherentes al proyecto y las delimitaciones impuestas, se define el objetivo general y a su vez los objetivos específicos, los cuales representan las tareas a seguir para llevar a cabalidad el proyecto.

1.6.1. General

Desarrollar un sistema de corrección de ruido aplicado a imágenes de conducción asistida, mediante la implementación de autoencoders.

1.6.2. Específicos

Identificar métodos de corrección de ruido y requerimientos técnicos de usuario en imágenes de conducción asistida.

Diseñar el algoritmo de corrección de ruido en imágenes de conducción asistida de acuerdo con las necesidades de usuario.

Implementar el algoritmo en lenguaje Python y validar el funcionamiento aplicando pruebas unitarias.

Evaluar el sistema aplicando análisis asintótico de complejidad computacional y sus resultados aplicando las métricas PSNR y SSIM.

Divulgar los resultados obtenidos mediante artículos publicados en revistas especializadas o en eventos científicos.

2. Marco Referencial

En este capítulo se aborda el estado del arte del tema de investigación, teniendo en cuenta trabajos e investigaciones científicas divulgadas tanto en revistas como en universidades a nivel mundial, analizando la metodología implementada para afrontar el problema, los resultados obtenidos y el conocimiento aportado al campo de la investigación. Además de lo anterior, también se plantean brevemente algunos conceptos teóricos que se hacen necesarios para comprender el tema de investigación. Así mismo, se tienen en cuenta los aspectos legales que intervienen en el debido desarrollo del proyecto.

2.1. Antecedentes

A continuación, se presenta la revisión de artículos de investigación y publicaciones científicas acordes al tema, esto con la finalidad identificar las dificultades y ventajas que se presentaron (ver Anexo 1), de manera que sirva de base o guía en este trabajo de investigación, permitiendo enfocar específicamente el eje investigativo, de esta manera, se pretende obtener resultados de forma más rápida y efectiva.

En el proceso de consulta de antecedentes se encontraron varios artículos relacionados con el tema central de esta investigación, sin embargo, se hace necesario poner en práctica las delimitaciones planteadas anteriormente, en las cuales, se definieron unos límites con la finalidad de conservar el tema central de estudio, teniendo esto en cuenta, se seleccionaron los artículos más acordes al eje de investigación, los cuales se analizan en los posteriores ítems. Por otra parte, para obtener una mayor lucidez en el tema, se plantea la Anexo 1, en el cual, se sintetizan los aportes y recomendaciones vistas en los artículos consultados que no han sido incluidos en la selección de antecedentes.

2.1.1. Deep Learning on Image Denoising: An Overview

El 6 de agosto de 2020, se publicó un artículo de investigación realizado por Chunwei Tian, Lunke Fei, Wenxian Zheng, Yong Xu, Wangmeng Zuo y Chia-Wen Lin, en el cual aplican diferentes métodos de denoising basados en redes neuronales convolucionales (CNN), con la finalidad de comparar los resultados obtenidos a través de pruebas para imágenes con ruido blanco, realmente ruidosa, con ruido híbrido o eliminación de ruido a ciegas, otorgando una comparativa de las ventajas y desventajas de cada técnica implementada, concluyendo que aún existen desafíos por superar en este campo, una de las recomendaciones dadas para futuras investigaciones es buscar métricas más precisas para evaluar el proceso de corrección de ruido en las imágenes (Tian, Fei, Zheng, Xu, & Lin, 2020).

2.1.2. Autoencoders Based Deep Learner for Image Denoising

En la 3ª conferencia internacional sobre computación y comunicaciones en red (CoCoNet'19) se presentó esta investigación realizada por Komal Bajaj, Dushyant Kumar Singh, Mohd. Aquib Ansari, donde implementaron autoencoders como un método de denoising para el ruido de tipo gaussiano, demostrando el buen desempeño que tienen los autoencoders en términos de PSNR en comparación con las tecnologías convencionales, abriendo la posibilidad a futuras investigaciones para expandir esta técnica de denoising a otros tipos de ruido (Bajaj, Kumar, & Aquib, 2020).

2.1.3. Devdan: Deep Evolving Denoising Autoencoder

En el volumen 390 de la revista Neurocomputing se publicó este artículo de investigación desarrollado por Andri Ashfahani, Mahardhika Pratamaa, Edwin Lughofer y Yew-Soon Ong, en el cual plantearon que el método de denoising con autoencoders era poco viable debido a que no posee la capacidad de adaptarse a entornos que cambian rápidamente, es así como proponen un

Deep evolving denoising autoencoder (DEVDAN) (codificador automático de eliminación de ruido de evolución profunda), un sistema que demostró ser altamente eficiente en comparación con los métodos usuales implementados (Ashfahani, Pratama, Lughofer, & Ong, 2020).

2.1.4. Llnet: A Deep Autoencoder Approach to Natural Low-Hight Image Enhancement

Este artículo de investigación publicado en junio del año 2016, por los autores Kin Gwn Lore, Adedotun Akintayo y Soumik Sarkar, pretende dar solución al inconveniente generado por poca iluminación en una imagen, lo cual dificulta el proceso de obtención de información. Este tipo de inconveniente se presenta en gran medida en las imágenes tomadas del entorno, en específico imágenes nocturnas, donde la luz natural es leve. Los autores presentan un método similar a la corrección gamma, sin embargo, cuenta con la ventaja de realizar una corrección de ruido simultáneamente a la corrección de iluminación, esto representa una mayor ventaja sobre los métodos usualmente usados en estos procesos. Todo lo anterior mediante la implementación de autoencoders (Lore, Akintayo, & Sarkar, 2017).

2.1.5. Underwater Color Restoration Using U-Net Denoising Autoencoder

En el año 2019, en el XI Simposio Internacional sobre Procesamiento y Análisis de Imágenes y Señales (ISPA), se presentó este artículo de investigación desarrollado por Yousif Hashisho, Mohamad Albadawi, Tom Krause y Uwe Freiherr von Lukas, en el cual se desarrolla un autoencoder que permite mejorar la calidad obtenida en imágenes submarinas, en otras palabras, reducir la interferencia presente en los videos a causa del agua, mostrando eficiencia en la realización de esta tarea, permitiendo procesar videos en tiempo real (Hashisho, Albadawi, Krause, & Freiherr, 2019). Lo anterior es de gran ayuda para la investigación presentada en este proyecto, dado que las imágenes son tomadas directamente del entorno y debido a condiciones

meteorológicas la presencia de agua en las imágenes puede dificultar el procesamiento y obtención de información.

2.2. Marco teórico

En esta sección se abordan brevemente los conceptos teóricos en los cuales, está fundamentada la investigación, esto con la finalidad de comprender en mejor medida el funcionamiento y desarrollo del proyecto, teniendo en cuenta datos abordados por libros y proyectos de investigación anteriormente desarrollados. Adicionalmente, en la Figura 3 se presenta un mapa conceptual donde se sintetiza la teoría tomada en cuenta, a su vez, dicho mapa conceptual propone el orden lógico de ideas que se recomienda seguir para comprender completamente la teoría planteada en este marco teórico.

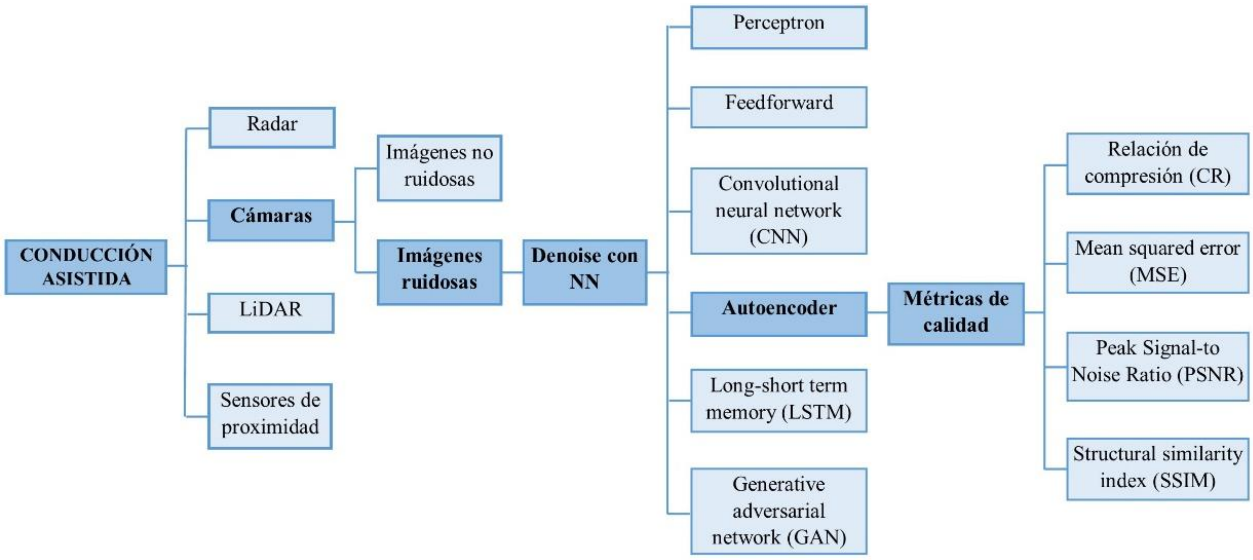


Figura 3. Teoría necesaria para comprender el desarrollo y funcionamiento de esta investigación.

2.2.1. Conducción Asistida

Actualmente se están implementando diferentes alternativas que permitan brindar mayor seguridad a los conductores y peatones (Talero, 2015), una de estas prácticas implementadas es la conducción asistida, la cual consiste en brindar apoyo al usuario durante la conducción, esto

por medio de sensores, cámaras, entre otros dispositivos que permiten obtener datos del entorno y complementar la información que el usuario obtiene a través de sus sentidos (Leal, 2021), de esta forma se pretende alertar de posibles incidentes o accidentes, y de ser posible ayudar a prevenirlos a tiempo. Un ejemplo de este tipo de prácticas son los asistentes de ángulo muerto (Blind spot detection), el cual ayuda a supervisar aquellas zonas que se salen del rango de visión del conductor (Continental, 2020).

2.2.2. Obtención de Información del Entorno

La conducción asistida se basa en los datos que el vehículo pueda recolectar del entorno, para esto se implementan diferentes métodos o sistemas de recolección de información, siendo los más comunes los que se describen a continuación.

2.2.2.1. Radar.

El radar es un tipo de tecnología basada en la energía microondas y el efecto Doppler, para su funcionamiento lo que hace es focalizar la energía en una zona en específico, cuando un obstáculo se interpone el haz de ondas emitido es reflejado y percibido por una antena, de esta forma el vehículo puede identificar la presencia de objetos cercanos (Leal, 2021), dada la naturaleza del funcionamiento de este sistema, se obtiene un rango de detección largo, aunque estrecho.

2.2.2.2. Cámaras.

Las cámaras contribuyen a complementar el rango de visión del conductor, pues al momento de conducir se cuenta con gran cantidad de distractores que impiden percibir el entorno (Caparrós, 1999), esto además de los denominados “puntos ciegos” los cuales comprenden zonas a las que el usuario no tiene acceso visual mientras conduce, es por esto que los sistemas de

conducción asistida registran, procesan e interpretan las imágenes en busca de información útil para el conductor, como la presencia de vehículos u objetos cercanos (Leal, 2021).

2.2.2.3. LASER Imaging Detection and Ranging (LiDAR).

El LiDAR consiste en una técnica que mide el entorno que rodea al vehículo en tres dimensiones, esto mediante la emisión de impulsos luminosos y la media del tiempo que tardan en volver, se basa en una fuente activa de luz, por lo que no necesita una fuerte iluminación natural (Leal, 2021), haciéndolo ideal para entornos de iluminación escasa donde se dificulta la identificación de obstáculos por medio de otros métodos como las cámaras.

2.2.2.4. Sensores de Proximidad (Ultrasonido).

Los sensores de ultrasonido tienen un funcionamiento similar al del radar, con la diferencia de que estos usan ondas ultrasónicas para medir la distancia que existe en el vehículo y posibles obstáculos cercanos.

2.2.3. Tipos de Imágenes y Métodos de Denoising

Actualmente las cámaras son comúnmente implementadas como método de obtención de información del entorno (Leal, 2021), sin embargo, esta información presenta distorsión o ruido, algo que se hace imposible de evitar pues se debe no solo a condiciones técnicas del sistema (Fernández, 1996) sino también a condiciones externas como el ambiente (Mohd & Jayant, 2013), es por esto que las imágenes obtenidas normalmente contienen ruido ya sea en mayor o menor medida, de allí surge la necesidad de implementar métodos de denoising que permitan reducir la cantidad de ruido presente en las imágenes y de esta forma poder obtener la mayor información posible a partir de estas. A continuación, se plantean algunos de los tipos de ruido más comunes y métodos de denoising basados en neural networks (NN).

2.2.3.1. Eliminación de Ruido Blanco Aditivo.

Las imágenes con ruido blanco aditivo son muy comunes en los entornos de investigación, debido a que son usadas para entrenar sistemas de denoising basados en redes neuronales, los cuales son más fáciles de producir que una imagen con ruido real, pues el ruido blanco aditivo se agrega por medio de software. Las imágenes de ruido blanco aditivo más usadas son Gaussianas, Poisson, Salt & Pepper, entre otras. Existen diferentes técnicas de Deep learning para la eliminación de este tipo de ruido, las cuales incluyen, redes neuronales convolucionales (CNN), extracción de características principales, y la combinación de métodos de optimización (Tian, Fei, Zheng, Xu, & Lin, 2020).

2.2.3.2. Eliminación de Ruido en Imágenes Realmente Ruidosas.

Para la eliminación de ruido en este tipo de imágenes, algunos autores proponen dos técnicas, con CNN única de extremo a extremo y la combinación de conocimiento previo, esto representaría modificar la arquitectura de la red (Tian, Fei, Zheng, Xu, & Lin, 2020). Por otra parte, otros autores proponen que para esta tarea es más viable una red de eliminación de ruido a ciegas, implementado arquitectura modular (Anwar & Barnes, 2019).

2.2.3.3. Eliminación de Ruido a Ciegas.

Las técnicas de eliminación de ruido a ciegas suponen un cambio en comparación con los métodos de denoising convencionales, puesto que, en este tipo de proceso, el sistema aprende la base a utilizar en la eliminación de ruido a partir de la misma muestra ruidosa (Majumdar, 2019), esto supone una mayor complejidad al querer abordar este proceso por medio de autoencoders, dado que estos normalmente son entrenados con un tipo en particular de imágenes, y al cambiar el tipo de imagen de entrada, el autoencoder no funciona correctamente, sin embargo, algunos

autores han planteado métodos de reducción de ruido a ciegas que aparentan ser eficientes en esta tarea (Majumdar, 2019).

2.2.3.4. Eliminación de Ruido Híbrido.

Dada la complejidad que se presenta en el mundo real en lo concerniente a la presencia de ruido en imágenes, es común que una misma imagen suele verse afectada por diferentes tipos de ruido, dando pie a investigaciones que pretendan encontrar un modelo capaz de eliminar el ruido híbrido, es así como se han presentado diferentes técnicas que pretenden dar solución a este inconveniente, como son la combinación de CNN y orientación deformada, CNN con algoritmos iterativos, redes en cascada, entre otros (Tian, Fei, Zheng, Xu, & Lin, 2020).

2.2.4. Redes Neuronales

Las bases de lo que hoy conocemos como redes neuronales artificiales surgieron el siglo anterior, con la realización de investigaciones en torno al "*data science"*. En el año de 1943 Warren S. McCulloch y Walter Pitts definieron formalmente el concepto de neurona en su artículo titulado *A logical calculus of the ideas immanent in nervous activity* (McCulloch & Pitts, 1943), en el cual se refirieron a las neuronas artificiales como una maquina binaria que contaba con varias entradas y salidas, a raíz de esta idea, Frank Rosenblatt postulo el modelo de perceptrón en 1958 (Rosenblatt, 1958), el cual añadía a la red neuronal un mecanismo de aprendizaje.

Consecuentemente se realizaron investigaciones a lo largo de los años, planteando diferentes modelos de redes neuronales, las cuales cada vez eran más complejas. En la Figura 4, se presenta la evolución que han tenido las redes neuronales a lo largo de la historia, abordando algunos de los principales modelos.

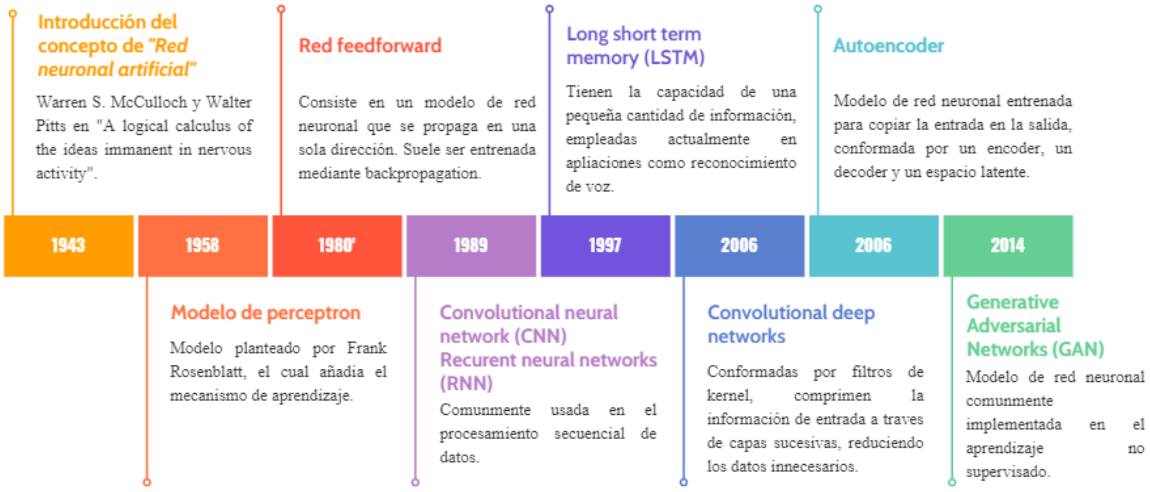


Figura 4. Breve historia de las redes neuronales *(McCulloch & Pitts, 1943)*, *(Rosenblatt, 1958)*, *(de los Reyes, 2019)*, *(Barradas, 2020)*.

2.2.5. Autoencoders

En pocas palabras, los autoencoders consisten en una arquitectura de red neuronal que busca encontrar una representación comprimida a partir de una entrada de datos y en función de esto, trata de reconstruir la entrada en la salida.

Normalmente se suele tomar al autoencoder como una sola red neuronal, sin embargo, si se desea ser un poco más explícito, se podría afirmar que este consiste en una composición de dos redes neuronales, las cuales son el Encoder y Decoder, y están unidas por un nexo en común denominado espacio latente (Barradas, 2020).

Cada una de las partes del autoencoder realiza una función específica que contribuye a su correcto funcionamiento, a continuación, se da una breve descripción de cada una de sus partes.

- Encoder: conforma la primera parte del autoencoder y se encarga de comprimir la entrada x en un vector $z = f(x)$, dado que este vector debe ser de baja dimensión, el codificador se ve obligado a extraer solo los datos de mayor relevancia (Atienza, 2018).

- Espacio latente: Hace referencia a la presentación más reducida del Encoder, en esta capa se encuentran los datos de mayor relevancia necesarios para reconstruir la señal de entrada.
- Decoder: Es la última parte del autoencoder y en esta se trata de reconstruir la señal de entrada a partir de los datos salvados en el vector latente, se puede representar de la siguiente forma $g(z) = \tilde{x}$. La finalidad del Decoder es hacer que la salida $(\tilde{x})$ sea lo más parecido a la entrada (x) (Atienza, 2018).

Teniendo en cuenta lo anterior, el autoencoder en términos generales se puede representar como $g(f(x)\,) = \tilde{x}$, donde $\tilde{x}$ debe ser aproximadamente igual a x.

Cabe aclarar que el autoencoder tiene un funcionamiento secuencial (Barradas, 2020), es decir, la entrada de datos ingresa al Encoder, este se encarga de comprimir la señal y procede a enviarla al espacio latente, el cual a su vez consiste en la entrada del Decoder, este proceso se puede apreciar de mejor manera en el esquema de la Figura 2 (Capítulo 1).

Los autoencoders tienen una gran cantidad de aplicaciones, tanto en su forma más simple como parte de redes neuronales más complejas, consisten en una herramienta clave para entender el aprendizaje profundo. Algunas de las aplicaciones más comunes son, aritmética a nivel de caracteres, detección, seguimiento, segmentación, coloración y por supuesto, reducción de ruido.

2.2.6. Métricas Aplicadas en el Procesamiento de Imágenes

La calidad de una imagen puede degradarse o disminuir debido a diferentes factores, tanto durante la adquisición como en el procesamiento de las mismas, algunas de las formas más comunes en las cuales se degrada la calidad de una imagen son el desenfoque, el timbre, el ruido, entre otros (MathWorks, s.f.).

A causa de lo anterior se han realizado esfuerzos con la finalidad de encontrar medidas objetivas de calidad, debido a que las subjetivas involucran la percepción por parte de un observador humano, lo cual genera que la calidad sea relativa en función de las consideraciones tenidas en cuenta por el observador.

En la actualidad se cuenta con diferentes métricas objetivas que permiten evaluar la calidad de una imagen, en función de la compresión, el ruido presente, la similitud, entre otros factores (León, Bermeo, Paredes, & Torres, 2020). A continuación, se presenta una breve explicación de algunas de las métricas objetivas más usadas en el procesamiento de imágenes.

2.2.6.1. Relación de Compresión (CR).

La compresión de imágenes comprende un proceso por medio del cual se reduce el tamaño de bytes en un archivo, pero, sin degradar la calidad a un nivel inaceptable, esto permite almacenar más archivos en un mismo espacio de memoria (León, Bermeo, Paredes, & Torres, 2020). La relación de compresión se define como

$$CR = \frac{n_1}{n_2}, \tag{1}$$

donde n_1 es el numero de bytes de la imagen original y n_2 el de la imagen comprimida.

2.2.6.2. Mean Squared Error (MSE).

También conocido como error medio cuadrático, mide la diferencia cuadrada promedio entre los valores reales e ideales (MathWorks, s.f.), el MSE está descrito como

$$MSE = \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}[I(x,y) - K(x,y)]^2 , \tag{2}$$

donde I y K son imágenes a comparar de tamaño $m \times n$. Cabe aclarar que al calcular este valor podría no concordar con la percepción humana de calidad.

2.2.6.3. Peak Signal-to-Noise Ratio (PSNR).

En español conocida como proporción máxima de señal a ruido, es una derivación del MSE y relaciona la intensidad máxima de pixeles con la potencia de la distorsión (MathWorks, s.f.). Es usado comúnmente para medir la calidad en imágenes reconstruidas, donde la señal corresponde a los datos originales de entrada, y el ruido es el agregado a causa de la reconstrucción (León, Bermeo, Paredes, & Torres, 2020).

Matemáticamente se puede expresar el PSNR como

$$PSNR = 10 \log_{10} \frac{(max_I)^2}{MSE}, \quad (3)$$

donde max_I es la fluctuación máxima en la imagen y MSE es el error cuadrático medio. En la Figura 5, se puede apreciar gráficamente en que consiste el PSNR, mostrando dos imágenes, una con ruido Salt & Pepper a $Prob = 0.15$ (Figura 5a) y otra aparentemente sin ruido (Figura 5b), notando que el PSNR de la primera imagen respecto a la original es relativamente bajo, degradando la calidad de la imagen y dificultando la extracción de información.

a) Imagen con ruido (PSNR=12.54 dB). b) Imagen original.

Figura 5. Relación del PSNR con la calidad de la imagen.

2.2.6.4.Structural Similarity Index (SSIM).

Conocido en español como índice de similitud estructural, combina tres de los principales parámetros de la imagen dada (luminancia, contraste y estructura) en una única

puntuación de calidad (Horé & Ziou, 2010). Este índice se puede representar matemáticamente de la siguiente manera

$$SSIM(I,K) = l(I,K)\, c(I,K)\, s(I,K)\,, \tag{4}$$

donde el primer término corresponde a la luminancia, el segundo está en función del contraste y el tercero depende de la saturación, a su vez cada termino se puede representar como

$$\left.\begin{aligned} l(I,K) &= \frac{2\mu_I\mu_K + c_1}{{\mu_I}^2 + {\mu_K}^2 + c_1} \\ c(I,K) &= \frac{2\sigma_I\sigma_K + c_2}{{\sigma_I}^2 + {\sigma_K}^2 + c_2} \\ s(I,K) &= \frac{\sigma_{IK} + c_3}{\sigma_I\sigma_K + c_3} \end{aligned}\right\}, \tag{5}$$

el primer termino $l(I,K)$ mide la cercanía de la media de la luminancia de ambas imágenes ($\mu_I,\ \mu_K$), tomando un valor máximo de 1 solo cuando $\mu_I = \mu_K$; el segundo término $c(I,K)$ comprende la comparación del contraste, medido por la desviación estándar ($\sigma_I,\ \sigma_K$), análogamente a la luminancia, este término tendrá un valor máximo de 1 cuando $\sigma_I = \sigma_K$; el tercer término $s(I,K)$ es función de la comparación estructural de ambas imágenes (*I*, *K*), donde σ_{IK} es la covarianza entre K e I.

El SSIM puede tomar valores entre 0 y 1, donde cero significa que no hay relación entre las imágenes dadas y 1 significa que ambas imágenes son iguales (*I*=*K*). Por último, las constantes c_1, c_2 y c_3 se implementan con la finalidad de evitar obtener un denominador nulo.

2.3. Marco legal

Para el correcto desarrollo de este proyecto fue necesario tener en consideración los aspectos legales que aplican debido al campo de aplicación de la investigación, es así como para la programación necesaria, se seleccionó en primera instancia un lenguaje de programación que se adaptara a la naturaleza del mismo, en específico Python, seguidamente se buscó software de código abierto que permitiera llevar a cabo el proyecto. Por otra parte, se hizo necesario tener en

cuenta algunas normas que reglamentan el ámbito de la conducción asistida y todo lo que esto conlleva, así como lo relacionado con el tratamiento de datos. Conforme a lo mencionado anteriormente, se presentan a continuación algunas de las normas establecidas que se tuvieron en consideración durante el desarrollo de la investigación.

2.3.1. Reglamentación en el Ámbito de la Conducción Asistida

El campo de la conducción asistida, al igual que muchos otros campos de investigación, cuenta con normas que se encargan de garantizar la mayor calidad, es así como se presenta la norma ISO 26262, la cual vigila el desarrollo de sistemas de alta integridad en la industria de la automatización. Dicha norma clasifica las funciones en niveles de integridad de seguridad automotriz (ASIL), dichos niveles van del A al D, donde el A es el nivel de integridad más bajo y D el más riguroso, es decir el más exigente con la mayoría de los requisitos (MathWorks, s.f.).

Por otra parte, la Society of Automotive Engineers (SAE) por medio del estándar J3016 categoriza la automatización de la conducción asistida en seis niveles los cuales van desde la ausencia de automatización (Nivel 0), hasta la automatización completa de la conducción (Nivel 5), la distribución de dichos niveles se presenta a continuación (SAE International, 2021):

- Nivel 0: Sin automatización de conducción.
- Nivel 1: Asistencia al conductor.
- Nivel 2: Automatización de conducción parcial.
- Nivel 3: Automatización de conducción condicional.
- Nivel 4: Alta automatización de conducción.
- Nivel 5: Automatización de conducción completa.

Los sistemas de conducción asistida clasificados como nivel 3 o superiores son aquellos que se caracterizan por recolectar datos a través de sensores con la finalidad de crear un modelo del

entorno y en función de esto asistir al conductor o controlar el vehículo, en estos niveles es donde se encuentran los sistemas de conducción asistida contemplados en esta investigación.

Si se desea profundizar en la clasificación de la conducción asistida planteada por la SAE J3016, se puede consultar el Anexo 2, el cual muestra de manera gráfica la relación entre los niveles con sus respectivos parámetros o condiciones.

2.3.2. Uso de Software Libre

A lo largo de los años y debido al avance tecnológico se han presentado diferentes proyectos de ley a nivel nacional con la finalidad de regular el uso de software libre, para de esta forma contribuir a la investigación en el país, es así como surgen los siguientes proyectos de ley:

- Proyecto de ley de 2002: El proyecto de ley 83 de 2002 referente a la regulación del uso de software de Colombia, “por medio del cual se incentiva el uso del software libre como mecanismo para fomentar el respeto a los derechos constitucionales de los ciudadanos e incentivar el desarrollo tecnológico de la Nación” (Pereira, 2017).
- Proyecto de ley de 2007: Dado el fracaso del proyecto de ley de 2002, se generaron nuevas estrategias con la finalidad de presentar otra propuesta, siendo está el proyecto de ley 021 de 2007, “por la cual se implementa la utilización del software libre en las entidades del Estado” (Pereira, 2017).
- Proyecto de ley 2009: Dos años después se presenta el proyecto de ley 017 de 2009, el cual pretendía “incentivar el desarrollo tecnológico en todo el país, lo cual va a permitir garantizar un control efectivo de métodos que las mismas entidades utilicen, permitiendo que se pueda organizar sus sistemas informáticos de manera eficiente y productiva”, este proyecto de ley fue archivado en segundo debate debido a la sentencia de la corte

suprema de justicia bajo el argumento de que “el concepto de neutralidad tecnológica va en pro del libre mercado y la competencia” (Pereira, 2017).

Sin embargo, debido al fracaso de esos proyectos, estas leyes no se tendrán en consideración, pues no fueron sancionadas. Por lo anterior, se tendrá en cuenta la ley 44 de 1993 (Congreso de Colombia, 1993), la cual regula el tratamiento de los derechos de autor, y penaliza el uso indebido de software licenciado, fomentando de esta forma el uso de software correctamente licenciado, o en su defecto, el uso de software de libre acceso.

Por otra parte, internacionalmente cuando se hace referencia al uso de software libre, entre varias opciones de código abierto, se tiene el licenciamiento MIT, la cual otorga a los usuarios permiso expreso para reutilizar el código en cualquier propósito, esta licencia es reconocida como uno de los acuerdos de licencia de código abierto más simples. En el cuerpo de la licencia se menciona que “el software se proporciona tal “cual”, sin garantía de ningún tipo, expresa o implícita, incluyendo, pero no limitado a, las garantías de comerciabilidad, aptitud para un propósito particular y no infracción. en ningún caso los autores o titulares de los derechos de autor serán responsables de cualquier reclamo, daños u otra responsabilidad, ya sea en una acción de contrato, agravio o de otro modo, que surja de, fuera de o en relación con el software o el uso u otras negociaciones en el software” (Open source initiative , s.f.).

2.3.3. Tratamiento de Datos

El acceso, distribución y modificación de la información de terceros es un tema muy delicado legalmente hablando, pues la información se considera de carácter personal o privado, es por esto que se cuenta con diferentes leyes que regulan estos aspectos (Cano & Bautista, 2019).

Teniendo en cuenta lo anterior se considera para esta investigación el “Habeas data”, el cual es una garantía constitucional que protege a las personas contra el uso abusivo de la información personal. A través de la Ley 1581 de 2012 y el Decreto 1377 de 2013 se da el derecho constitucional que tienen todas las personas a conocer, suprimir, actualizar y rectificar todo tipo de datos personales recolectados, almacenados o que hayan sido objeto de tratamiento en bases de datos ya sea por entidades públicas o privadas, además también de otorgar al titular de los datos personales el poder para limitar las posibilidades de la divulgación, publicación o cesión la información recolectada (Ministerio de educación, 2020).

3. Metodología

Teniendo en cuenta los objetivos, lineamientos y delimitaciones propuestas anteriormente, y con la finalidad de estructurar las actividades realizadas en el desarrollo de este proyecto, se planteó una metodología dividida en cinco fases, como se aprecia en la Figura 6. Cada una de estas fases abarca las actividades que se realizaron para el cumplimiento de los objetivos planteados en el primer capítulo de este documento.

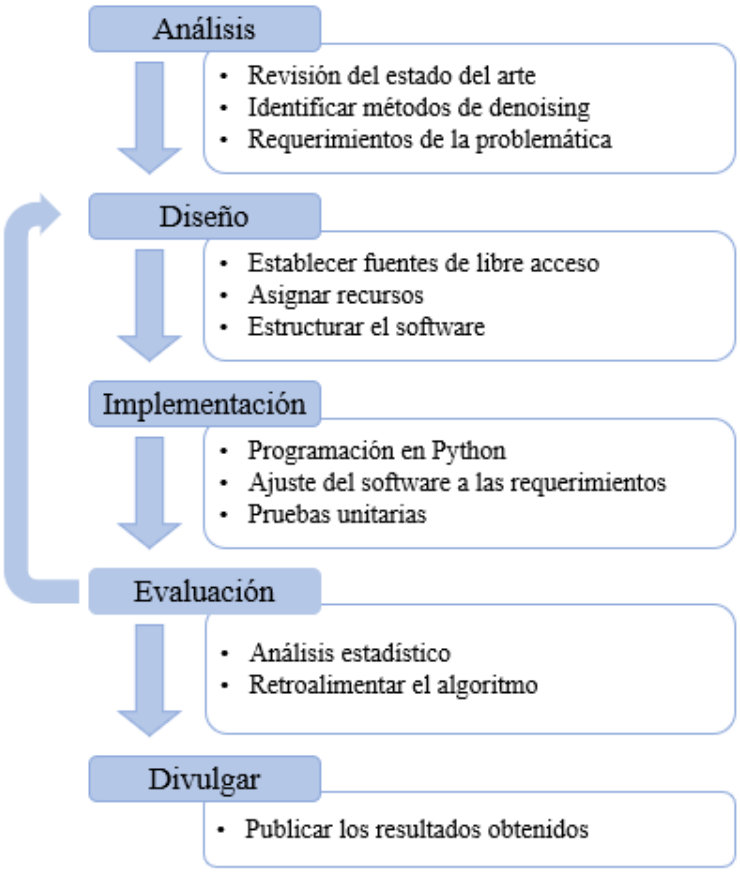


Figura 6. Esquema de la metodología implementada.

3.1. Análisis

La fase inicial de este proyecto se centró en realizar un estudio detallado del campo de investigación, esto con la finalidad de identificar las diferentes alternativas que existen para abordar el proceso de corrección de ruido en imágenes para sistemas de conducción asistida, y de

este modo contribuir en este campo de investigación y cumplir con los objetivos del proyecto. Teniendo en cuenta lo anterior, la fase de análisis se encuentra conformada por tres actividades, las cuales son, revisión del estado del arte, identificación de método de denoising y requerimientos de la problemática.

En la primera actividad de la fase de análisis se consultaron buscadores científicos como Springer o Google Académico, con la finalidad de identificar investigaciones acordes al tema abordado en este documento, teniendo en cuenta los aportes realizados por cada investigación.

Como segunda actividad de la fase de análisis, se plateó una identificación de los métodos de denoising, proceso que fue necesario para conocer las diferentes alternativas existentes en la tarea de corrección de ruido en imágenes. Además de permitir identificar los principales tipos de y su naturaleza o procedencia.

Finalmente, la tercera actividad de la fase de análisis consistió en plantear o identificar los requerimientos de la problemática abordada, teniendo en cuenta los lineamientos establecidos para el desarrollo de la investigación y con la finalidad de dar cumplimiento a los objetivos planteados. Para esto se parte de la información consultada en la revisión del estado del arte y la identificación de los diferentes tipos de ruido y métodos existentes para la corrección del mismo.

3.2. Diseño

Como segunda fase de esta investigación se planteó el diseño del algoritmo que permite realizar el proceso de denoising en imágenes de conducción asistida, para esto se tuvieron en cuenta las recomendaciones vistas en el estado del arte, así como los requerimientos establecidos en la primera fase de la metodología. Con la finalidad de cumplir lo antes mencionado, la fase de diseño está conformada por tres actividades, las cuales son, identificación de fuentes de libre acceso, asignación de recursos y estructuración del software.

La primera actividad de la fase de diseño, teniendo en cuenta que una de las limitaciones del proyecto se centró en desarrollarse únicamente con recursos de código abierto o de libre acceso, se plantearon diferentes alternativas en cuanto al lenguaje de programación, entorno de desarrollo integrado y dataset.

En la segunda actividad de la fase de diseño se llevó a cabo la asignación de recursos, esto a partir de las alternativas planteadas en la actividad anterior, y teniendo en cuenta las limitaciones y delimitaciones establecidas en el primer capítulo de este documento, de este modo se pudieron seleccionar las herramientas o recursos que permitieron llevar a cabo el desarrollo del sistema de corrección de ruido en imágenes de conducción asistida.

Por último, en la tercera actividad de la fase de diseño se planteó la estructura del software que permite llevar a cabo el proceso de corrección de ruido. Para ello se estableció la arquitectura de red neuronal artificial a implementar y los parámetros que esta posee, teniendo en cuenta los recursos y herramientas seleccionados en la actividad anterior.

3.3. Implementación

Continuando con el desarrollo del proyecto y en base a las decisiones tomadas en las fases de análisis y diseño, se procede a desarrollar o implementar el algoritmo de corrección de ruido en imágenes de conducción asistida. Para esto se plantearon tres actividades, las cuales son, programación en Python, ajuste del software a los requerimientos y la realización de pruebas unitarias.

En la primera actividad de la fase de implementación fue necesario el uso de librerías que proporcionan las herramientas necesarias para trabajar con procesamiento de imágenes. Para desarrollar la programación se partió de la estructura antes definida, llevando a cabo una programación estructural que permitió desarrollar y ejecutar cada uno de los algoritmos.

Seguidamente, se llevó a cabo la segunda actividad de la fase de implementación, la cual, consistió en ajustar el software a los requerimientos con la finalidad de cumplir los objetivos del proyecto. Durante esta actividad fue necesario tener en cuenta los recursos seleccionados y en función de esto se modificaron algunos parámetros de la red como lo son el tamaño y cantidad de datos, épocas de entrenamiento, entre otros.

Finalmente, la última actividad de la fase de implementación consistió en llevar a cabo pruebas unitarias del sistema, esto con la finalidad de comprobar la funcionalidad del mismo en el proceso de corrección de ruido en imágenes. Para validar los resultados obtenidos, fue necesario aplicar métricas objetivas de calidad que permitieron cuantificar la respuesta del sistema.

3.4. Evaluación

La cuarta fase de la metodología implementada consiste en la evaluación del sistema de corrección de ruido, en términos del costo computacional y el proceso de corrección de ruido. se aplicaron una serie de pruebas que permitieron identificar la respuesta del sistema frente a ruido AWGN y Salt & Pepper, llevando a cabo las actividades de análisis estadístico y retroalimentación del algoritmo.

En la primera actividad de la fase de evaluación, fue necesario evaluar la respuesta del sistema frente a una mayor cantidad de datos de entrada, de esta forma se pudieron obtener datos estadísticos que proporcionan mayor información acerca del funcionamiento del sistema de corrección de ruido. Además, se llevó a cabo una evaluación del costo computacional, esto en función del tiempo de procesamiento.

Como segunda actividad de la fase de evaluación, se realizó una retroalimentación del algoritmo, con base en los resultados obtenidos en la anterior actividad. Esto permitió identificar

las principales falencias del sistema y llevar a cabo modificaciones que permitieron mejorar el proceso de corrección de ruido y, por consiguiente, incrementar la ganancia de PSNR y SSIM.

3.5. Divulgación

El objetivo final de toda investigación debe ser la transmisión del conocimiento, esto con el fin de facilitar el acceso a la información y permitir que se desarrollen futuras investigación en ese mismo eje temático. Teniendo esto en cuenta, la única actividad de esta última fase fue publicar los resultados obtenidos. Para esto, se presentaron los resultados obtenidos a través de participaciones en eventos de divulgación científica como ponencias, y publicaciones de artículos en revistas indexadas.

4. Resultados

A continuación, se presentan y analizan los resultados obtenidos en esta investigación, abordando cada uno de los objetivos planteados, evaluando la respuesta del sistema frente a ruido AWGN y Salt & Pepper, esto en términos de PSNR y SSIM. Permitiendo identificar las circunstancias que permiten obtener la mejor respuesta del sistema de corrección de ruido.

4.1. Métodos de Denoising y Requerimientos Técnicos

El primer objetivo planteado en esta investigación se centró en llevar a cabo una revisión del estado del arte que permitiera identificar las diferentes alternativas existentes para el proceso de corrección de ruido en imágenes, y posteriormente poder identificar los requerimientos de la problemática en función de la aplicación del sistema con imágenes de conducción asistida.

4.1.1. Revisión del Estado del Arte

Para llevar a cabo la revisión del estado del arte se consultaron diferentes proyectos investigativos, de los cuales, se tuvo en cuenta las dificultades y/o sugerencias planteadas por los autores, resaltando sus aportes y permitiendo obtener una visión más amplia en lo relacionado a las diferentes formas de afrontar la problemática.

Los principales factores que se tuvieron presente en la revisión del estado del arte fueron los aportes y el tema de investigación, abordando cuatro ejes temáticos relacionados con el sistema de corrección de ruido desarrollado en este proyecto, los cuales son self-driving, video processing, denoising methods y Autoencoders. Planteando el esquema mostrado en la Figura 7.

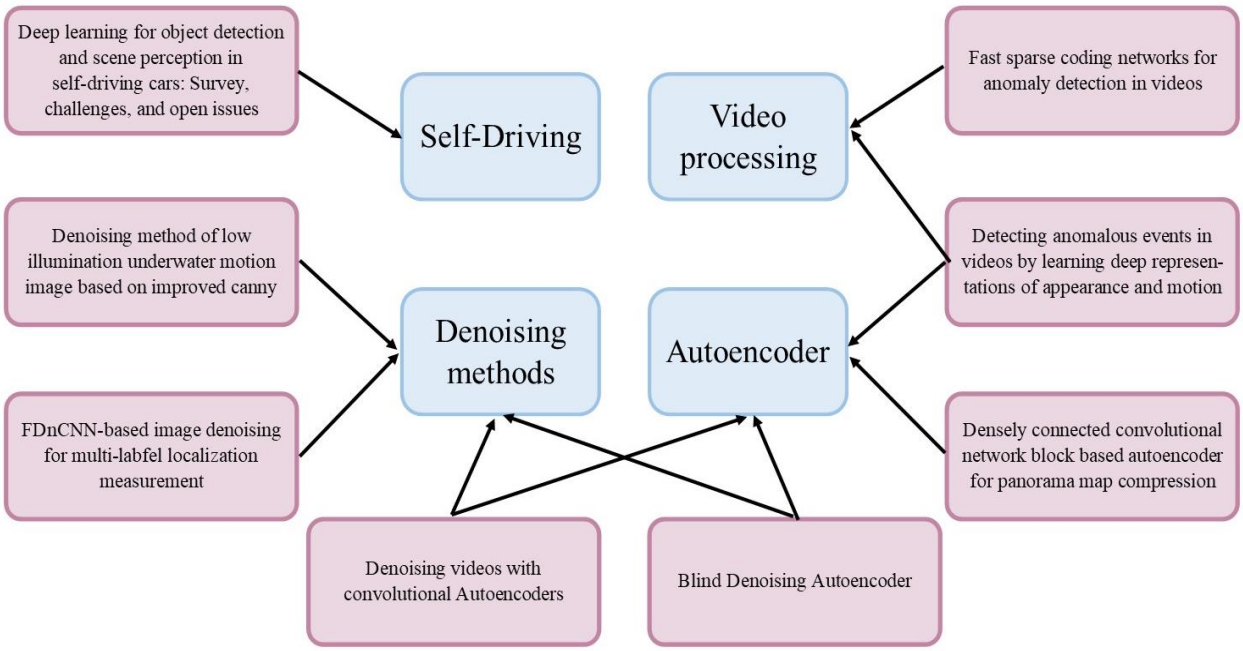


Figura 7. Clasificación de las investigaciones revisadas en el estado del arte *(Majumdar, 2019) (Gupta, Anpalagan, Guan, & Shaharyar, 2021) (Wang, Wang, Xiang, & Yu, 2020) (Xu, Yan, Ricci, & Sebe, 2017) (Wu, Liu, Li, Sun, & Shen, 2020) (Larrue, Li, Meng, & Han, 2018) (Lv & Li, 2021) (Li, y otros, 2020)*.

Además de lo anterior, también se realizó una tabla donde se sintetizaron los aportes realizados y las recomendaciones o dificultades presentes en cada investigación, dicha tabla se puede apreciar en el Anexo 2.

4.1.2. Identificación de Métodos de Denoising

Con el propósito de conocer las diferentes alternativas en el proceso de corrección de ruido en imágenes aplicadas a la conducción asistida, se identificaron las técnicas, abordando específicamente aquellos métodos que involucran la aplicación de redes neuronales, para esto se partió de las investigaciones analizadas en el estado del arte.

4.1.2.1. Tipos de Ruido.

Antes de conocer los diferentes métodos de denoising que se implementan en el procesamiento de imágenes, es necesario identificar los tipos de ruido que suelen afectar la calidad de las mismas, siendo que, el método más eficiente se encuentra directamente relacionado al tipo de ruido que se desea corregir.

En entornos reales, el ruido que afecta las imágenes es de carácter aleatorio por lo que se dificulta predecir su origen, debido a que este puede provenir de factores externos como el ambiente o internos pertenecientes al dispositivo de captura (Contreras, Pabón, García, Rojas, & Arguello, 2021), por esto mismo se dificulta su generación de manera sintética para entrenar sistemas.

Por otra parte, se cuenta con el ruido blanco aditivo, el cual es ampliamente utilizado para el entrenamiento de sistemas que requieren imágenes ruidosas como entrada, claro ejemplo de esto son los sistemas de denoising. Dentro de esta categoría se encuentra el ruido gaussiano, de Poisson, Salt & Pepper, entre otros (Tian, Fei, Zheng, Xu, & Lin, 2020).

Con la finalidad de comprender en mayor medida lo mencionado anteriormente, al mismo tiempo que identificar los tipos de ruido, se revisaron diferentes investigaciones relacionadas a esta área, permitiendo seleccionar los tipos de ruido más comunes, los cuales se describen brevemente en la Tabla 1.

Tabla 1. Descripción de los tipos de ruido más comunes *(Mohd & Jayant, 2013)*.

Tipos de ruido	Descripción
Additive white Gaussian noise (AWGN)	Surge principalmente durante el proceso de adquisición, por fallos del sensor debido a poca iluminación, elevadas temperaturas, o durante el proceso de transmisión a causa del ruido electrónico de los circuitos.
Poisson	Se presenta cuando el número de partículas que transportan la energía (fotones), es muy pequeño, por lo que surgen fluctuaciones durante la medición.
Salt & Pepper	Denominado también como ruido de pico, es un tipo de ruido que genera pixeles oscuros en zonas brillantes y pixeles brillantes en zonas oscuras. Una de sus causas es debido a errores en el convertidor analógico-digital, errores de bits en la transmisión, entre otros factores.
Disparo	Corresponde al ruido dominante en las partes más claras de una imagen, suele ser causado por la variación del número de fotones en un determinado nivel de exposición. Este tipo de ruido sigue una distribución de Poisson similar a la Gaussiana.
Speckle	Es un tipo de ruido granular que afecta especialmente a las imágenes de radar activo y de radar de apertura sintética. Caracterizado por aumentar el nivel medio de gris en un área local.
Cuantificación (Ruido uniforme)	Causado por la cuantificación de los pixeles de una imagen detectada en un número de niveles discretos. Se caracteriza por poseer una distribución uniforme, así mismo puede depender de la señal siempre y cuando no exista otra fuente de ruido lo suficientemente grande.

4.1.2.2. Métodos de Denoising.

Una vez que abordados los diferentes tipos de ruido que afectan la calidad de las imágenes, se procedió a revisar los métodos de denoising aplicados por investigadores para el procesamiento digital de imágenes, permitiendo conocer tanto las dificultas como ventajas aportadas por cada método, así mismo, fue necesario identificar los resultados obtenidos en la aplicación de cada uno de estos. Para obtener esta información se consultaron únicamente aquellos métodos de denoising basados en Deep learning, pues en este campo de investigación es donde se desempeña este proyecto.

La mayor diferencia presente en los sistemas de denoising basados en Deep learning, es la arquitectura implementada para la interconexión de las redes neuronales, en otras palabras, su mayor diferencia es el tipo o modelo de red neuronal que implementa, con base en esto, surgen diferentes sistemas de denoising basados en redes neuronales, los cuales a su vez varían según sus aplicaciones, complejidad y resultados.

En términos generales, los métodos de denoising parten del diseño y codificación del sistema, posteriormente se realiza el entrenamiento del mismo a partir de imágenes con y sin ruido, de tal manera que la red neuronal aprenda a realizar el proceso de denoising. Sin embargo, este tipo de sistemas trae algunas limitaciones consigo, una de estas se debe a que durante el entrenamiento se usan imágenes con un tipo de ruido especifico, causando que, al momento de implementar el sistema, este solo corregirá satisfactoriamente el ruido correspondiente al usado durante el entrenamiento, perdiendo funcionalidad si se procesa otro tipo de ruido (Torres, y otros, 2022).

Una forma de eliminar la limitación mencionada anteriormente, es implementar métodos de reducción de ruido a ciegas (blind denoising) (Majumdar, 2019), esta técnica se caracteriza por no poseer una etapa de entrenamiento, sino que el proceso de aprendizaje se realiza durante la implementación, permitiendo que la red aprenda a partir de la misma señal de entrada (imagen con ruido), de esta manera se asegura que el sistema pueda procesar ese tipo de ruido en específico.

Además de lo anterior, también cabe resaltar que la arquitectura de red neuronal convolucional permite reducir el costo computacional durante el proceso de corrección de ruido (Tian, Fei, Zheng, Xu, & Lin, 2020), esto debido a la extracción automática de características. Por tal motivo, es uno de los tipos más usados en la literatura, al igual que el modelo de autoencoder, el cual permite codificar una imagen ruidosa seleccionando los parámetros de mayor relevancia para posteriormente reconstruirla con base en esos datos (Yapici & Akcayol, 2021), obteniendo de esta forma una imagen sin ruido.

4.1.3. Requerimientos de la Problemática

Teniendo en cuenta el campo de aplicación en el cual se desarrolló esta investigación, y con la finalidad de cumplir satisfactoriamente lo propuesto, se identificaron los requerimientos necesarios para afrontar la problemática, teniendo en cuenta los siguientes ítems:

4.1.3.1. Costo Computacional.

Debido a que el proyecto se enfoca a implementaciones en vehículos, se hace necesario que este cuente con un reducido costo computacional, así mismo, se requiere que el procesamiento de las imágenes sea en tiempo real, pues el sistema se aplica en momentos de conducción, donde el tiempo de procesamiento es un factor clave.

Con base en lo anterior se revisó la literatura en busca de investigaciones acordes a la problemática, de tal forma que permita conocer los aportes y las recomendaciones realizadas por diferentes investigadores. Como punto de partida se tomó la función de activación a implementar, debido a que toda red neuronal se encuentra compuesta por diferentes neuronas y estas a su vez requieren de una función de activación que permita definir la salida en función de los datos de entrada, algunos investigadores demostraron que la función de activación sigmoidal (Marreiros, Daunizeau, Kiebel, & Friston, 2008) y la función de activación tanh (Jarrett, Kavukcuoglu, Ranzato, & LeCun, 2009) requieren un alto costo computacional, debido a esto se opta por trabajar con la función de activación ReLU (Rectified Linear Unit), pues esta permite acortar el tiempo de entrenamiento (Lau & Hann, 2018). Dicha función se puede representar gráficamente como se aprecia en la Figura 8, y matemáticamente se encuentra descrita por la ecuación

$$f(x) = \begin{cases} x & x > 0 \\ 0 & x \leq 0 \end{cases}, \tag{6}$$

Tomando el valor de cero para valores negativos de entrada, mientras que para valores positivos adopta el mismo valor de entrada.

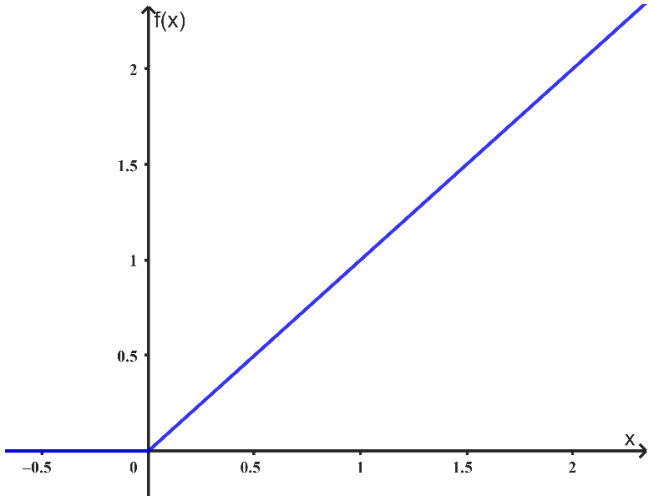


Figura 8. Representación gráfica de la función de activación ReLU.

De igual forma se encuentran en la literatura diferentes recomendaciones orientadas a reducir el costo computacional en los sistemas de Deep learning, y en específico sistemas de denoising. Para fines de este proyecto, en la Tabla 2 se sintetizan de forma breve algunas de dichas recomendaciones, esto con la finalidad de tenerlas en cuenta al momento de realizar el diseño.

Tabla 2. Recomendaciones para disminuir el costo computacional.

Investigación	Recomendación
Review of Adaptive Activation Function in Deep Neural Network (Lau & Hann, 2018)	Demostró que la función de activación ReLU permite acortar el tiempo de entrenamiento, según sus resultados, la función de activación ReLU obtuvo una tasa de error del 1,6%, siendo el porcentaje más bajo en comparación con las demás funciones de activación.
Image Denoising Via Sparse and Redundant Representations Over Learned Dictionaries (Elad & Aharon, 2006)	En esta investigación implementan un método de aprendizaje basado en diccionarios, usando el algoritmo K-SVD, lo cual permitió filtrar rápidamente el ruido y reducir el costo computacional.
Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising (Zhang, Zuo, Chen, Meng, & Zhang, 2017)	Los investigadores implementaron redes neuronales convolucionales feed-forward (DnCNN) para eliminación de ruido de imágenes, demostrando que con esta arquitectura se acelera

	el proceso de entrenamiento y mejora el rendimiento del sistema.
Going Deeper with Convolutions (Christian, y otros, 2015)	A través de este artículo los investigadores presentan una de red neuronal convolucional, cuyo mayor distintivo es mejorar la utilización de los recursos informáticos de la red, transformando un gran núcleo convolucional en dos más pequeños, reduciendo de esta manera la cantidad de parámetros y el costo computacional.
Block-Matching Convolutional Neural Network for Image Denoising (Ahn, 2017)	En esta investigación implementaron una red neuronal convolucional con auto-similitud no local (NSS), esto permitió acelerar el procesamiento de datos, así como reducir los costos computacionales.

4.1.3.2. Métricas.

Además de considerar el costo computacional dentro de los requerimientos, también se hace necesario revisar lo concerniente a las métricas. Para determinar esto se hace uso de métricas que permitan cuantificar la calidad, como lo son PSNR y SSIM. Teniendo en cuenta el tipo de imágenes abordadas en esta investigación, se revisaron únicamente modelos orientados al procesamiento de imágenes a color, teniendo en cuenta los resultados obtenidos por los modelos BM3D (Dabov, Foi, Katkovnik, & Egiazarian, 2007), WNNM (Gu, Zhang, Zuo, & Feng, 2014) y MemNet (Tai, Yang, Liu, & Xu, 2017), frente a imágenes con adición de ruido AWGN para tres valores de desviación estándar (σ), tal como se puede apreciar en la Tabla 3.

Tabla 3. Valores promedio de PSNR y SSIM correspondientes a la estimación respecto a la imagen ruidosa, para pruebas realizadas con el dataset Urban100.

	Nivel de ruido (σ)					
	30		50		70	
Métodos	PSNR (dB)	SSIM	PSNR (dB)	SSIM	PSNR (dB)	SSIM
BM3D	28.75	0.8567	25.95	0.7791	24.27	0.7163
WNNM	29.47	0.8697	26.83	0.8047	25.11	0.7501
MemNet	29.10	0.8631	26.65	0.8030	25.01	0.7496

Cabe resaltar que algunos investigadores luego de realizar diferentes pruebas, llegaron a la conclusión de que la métrica PSNR no es lo suficientemente acertada o valida como medida objetiva para cuantificar la calidad de videos, en su lugar recomiendan utilizar SSIM (Kotevski & Mitrevski, 2009).

Teniendo en cuenta lo anterior, se plantan en la Tabla 4 algunos valores de PSNR y SSIM a partir de los cuales se consideran aceptables los resultados obtenidos. Dichos valores se calcularon con base en el promedio de las investigaciones revisadas (Tabla 3). Sin embargo, cabe aclarar que estos resultados se obtuvieron dejando de lado otros factores como el costo computacional o el tiempo de procesamiento.

Tabla 4. Valores promedios de PSNR y SSIM calculados a partir de las investigaciones revisadas.

Nivel de ruido (σ)					
30		50		70	
PSNR (dB)	SSIM	PSNR (dB)	SSIM	PSNR (dB)	SSIM

29.11	0.8632	26.48	0.7956	24.80	0.7487

4.2. Diseño del Algoritmo DCAEAD

El segundo objetivo de esta investigación se centró el diseñar el algoritmo denominado DCAEAD (Denoising Convolutional Autoencoder for Assisted Driving). Para esto fue necesario identificar las fuentes de libre acceso que suministraron las herramientas o recursos necesarios durante el desarrollo del proyecto, y posteriormente, se llevó a cabo la estructura del software, esto teniendo en cuenta las recomendaciones vistas en el estado del arte, así como los requerimientos anteriormente planteados.

4.2.1. Fuentes de Libre Acceso

Inicialmente, se abordaron los componentes o herramientas necesarias para desarrollar el proyecto, esto desde el punto de vista del software de libre acceso. Para esto, se planteó una comparativa entre algunas opciones que permiten afrontar la problemática, identificando el aporte o beneficio que estas ofrecen. Como principales componentes o herramientas se tiene el lenguaje de programación, el entorno de desarrollo integrado y el conjunto de datos.

4.2.1.1. Lenguaje de Programación.

Actualmente, existe una gran variedad de lenguajes de programación, los cuales, proporcionan herramientas o librerías que facilitan ejecutar procesos complejos como lo es el procesamiento de imágenes, además de esto, cada lenguaje posee características propias, tal como se muestra en la Tabla 5, donde se realizó una comparativa de los tres lenguajes de programación contemplados como opciones para desarrollar el sistema de corrección de ruido planteado en esta investigación.

Tabla 5. Comparativa de lenguajes de programación.

Parámetros	C++ (C++ Foundation, s.f.)	Java (Oracle, s.f.)	Python (Python Software Foundation, s.f.)
Nivel de abstracción	Alto	Alto	Alto
Manera de ejecución	Compilado	Intermedio	Interpretado
Tipado	Si	Si	No
Paradigma de programación	Orientado a objetos	Orientado a objetos	Multiparadigma

4.2.1.2. Entorno de Desarrollo Integrado (IDE).

Dado que el desarrollo de este proyecto se basa completamente en codificación, se requiere un IDE que permita realizar la programación. Para ello existen diferentes programas que ofrecen una variedad de herramientas útiles para el programador, sin embargo, fue necesario realizar una comparativa que permitiera identificar las principales características y ventajas que ofrece cada uno. A raíz de estos se plantea en la Tabla 6 una comparativa de los IDE más comunes y que se ajustan en mayor medida a la investigación.

Tabla 6. Comparativa de los entornos de desarrollo integrado.

Parámetros	PyCharm (JetBrains, s.f.)	Google Colab (Google, s.f.)	VS Code (Microsoft, s.f.)
Compatibilidad	Windows, Mac, Linux	Desde un navegador independientemente del sistema operativo	Windows, Linux, Mac

Requisitos mínimos	4GB de RAM 2.5 GB de disco duro	Se ejecuta en la nube, no genera consumo local	1GB de RAM 0.5GB de disco duro
Consumo de GPU	GPU local	Ofrece GPU virtual	GPU local
Conexión a internet	No requiere	Obligatoria	No requiere
Acceso remoto	No	Si	No

4.2.1.3. Dataset.

Continuando con las herramientas necesarias para el desarrollo de este proyecto, se abordaron los diferentes conjuntos de datos disponibles, estos son de gran importancia pues repercuten directamente en los resultados que se obtienen del sistema, pues la calidad de estos dependió del dataset implementado, entre otros factores. Por esto se planteó una comparativa de tres dataset que aportan datos de conducción asistida, acordes a la investigación desarrollada tal como se muestra en la Tabla 7. Dicha comparativa se basó en criterios como la cantidad de datos para entrenamiento, prueba y validación del sistema, así como en el tipo de datos que ofrece.

Tabla 7. Comparativa de los dataset de libre acceso disponibles.

Parámetros	RoadText-1K: Text Detection & Recognition Dataset for Driving Videos (Reddy S. , Mathew, Gomez, & Rusinol, 2020)	PVS - Passive Vehicular Sensors Datasets (Menegazzo, 2021)	Berkely deep drive (McGunnigle, 2020)
Muestras de entrenamiento	500	9	7000

Muestras de prueba	300	0	2000
Muestras de validación	200	0	1000
Tipo de datos	Video	Video	Imágenes

Uno de los parámetros a tener en cuenta al momento de seleccionar un dataset, es la composición o características de los datos, pues en este caso, se necesita que estos contengan diferentes entornos, permitiendo entrenar un sistema capaz de responder a diferentes situaciones, por tal motivo en la Figura 9 se muestran imágenes extraídas de cada uno de los datasets disponibles. En el caso de los formatos en video se tomó una imagen de prueba de manera aleatoria.

a) RoadText-1K: Text Detection & Recognition Dataset for Driving Videos.

b) PVS - Passive Vehicular Sensors Datasets.

c) Berkely Deep Drive (BDD).

Figura 9. Muestra de los datos aportados por cada dataset.

De los datos mostrados en la Tabla 7, se puede ver que dos de los datasets disponibles contienen únicamente datos de video, los cuales, tienen una duración especifica, por tanto, en la Tabla 8 se muestra el tiempo de video aportado por cada dataset.

Tabla 8. Cantidad de tiempo de video aportado por los dataset.

RoadText-1K: Text Detection & Recognition Dataset for Driving Videos (Reddy S. , Mathew, Gomez, & Rusinol, 2020)		PVS - Passive Vehicular Sensors Datasets (Menegazzo, 2021)	
Videos aportados	Duración por video (Segundos)	Videos aportados	Duración por video (Segundos)
500	10	Video 1	1354
300	10	Video 2	1211
200	10	Video 3	982
		Video 4	1429
		Video 5	1305
		Video 6	936
		Video 7	1259
		Video 8	1213
		Video 9	898
Total	10000		10587

4.2.2. Asignación de Recursos

Una vez planteados los requerimientos de la problemática y las diferentes fuentes de libre acceso para los componentes o herramientas, se procedió a realizar la asignación de recursos, esto teniendo en cuenta las limitaciones y delimitaciones anteriormente expuesta, así como los recursos disponibles para el desarrollo del proyecto.

4.2.2.1. Selección del Lenguaje de Programación.

Para seleccionar el lenguaje de programación que más se ajusta a los requerimientos del proyecto, se propuso una matriz de selección, la cual evalúa la relevancia de cada una las opciones respecto a las demás, calificándolas como, mucho más importante (10), más importante (5), igual de importante (1), menos importante (1/5) y mucho menos importante (1/10).

Así mismo para la realización de cada matriz se debió evaluar el peso o importancia de cada criterio respecto a los demás, para posteriormente contrastar las opciones entre sí en función de cada criterio. En este proceso se deben tener en cuenta los siguientes términos:

FP → Factor de ponderación

PO → Peso de la opción

PF → Ponderación final

En cuanto a los criterios de evaluación, se tuvieron características como:

A → Nivel de abstracción

B → Manera de ejecución

C → Tipado

D → Paradigma de programación

En la Tabla 9 se puede apreciar la ponderación dada para cada criterio de evaluación bajo los cuales se realizó la selección del lenguaje de programación, estos criterios se evaluaron según la puntuación antes mencionada y teniendo en cuenta los requerimientos y necesidades anteriormente expuestos en este documento.

Tabla 9. Ponderación para los criterios de selección del lenguaje de programación.

	A	B	C	D	Suma	FP
A	X	1	1/5	1/5	1.4	0.0565
B	1	X	5	1	7	0.2823
C	5	1/5	X	5	10.2	0.4113
D	5	1	1/5	X	6.2	0.25
Total					24.8	

De igual manera, se plantean en la Tabla 10, Tabla 11, Tabla 12 y Tabla 13, las evaluaciones de cada opción con las demás en función de los criterios dados, esto permite obtener el PO.

Tabla 10. Ponderación de los lenguajes de programación según el nivel de abstracción.

	Nivel de abstracción				
	Java	Python	C++	Suma	PO
Java	X	1	1	2	0.3333
Python	1	X	1	2	0.3333
C++	1	1	X	2	0.3333
Total				6	

Tabla 11. Ponderación de los lenguajes de programación según la manera de ejecución.

	Manera de ejecución				
	Java	Python	C++	Suma	PO
Java	X	1/5	5	5.2	0.2537
Python	5	X	10	15	0.7317
C++	1/5	1/10	X	0.3	0.0146
Total				20.5	

Tabla 12. Ponderación de los lenguajes de programación según el tipado.

	Tipado				
	Java	Python	C++	Suma	PO
Java	X	1/5	1	1.2	0.0968
Python	5	X	5	10	0.8064
C++	1	1/5	X	1.2	0.0968
Total				12.4	

Tabla 13. Ponderación de los lenguajes de programación según el paradigma de programación.

	Paradigma de programación				
	Java	Python	C++	Suma	PO
Java	X	1/5	1	1.2	0.0968
Python	5	X	5	10	0.8064

C++	1	1/5	X	1.2	0.0968
Total				12.4	

Por último, una vez calculado el FP de los criterios de selección y el PO de los lenguajes dados, se procede a calcular la PF, para esto se realiza una tabla que relaciona los lenguajes de programación con los criterios de selección y se multiplica el FP de cada criterio por el PO de cada opción, finalmente se suman los productos de cada fila, tal como se muestra en la Tabla 14.

Tabla 14. Cálculo de la ponderación final para los lenguajes de programación.

	A	B	C	D	PF
Java	0.0565*0.3333 0.0188	0.2823*0.2537 0.0716	0.4113*0.0968 0.0398	0.25*0.0968 0.0242	0.1541
Python	0.0565*0.3333 0.0188	0.2823*0.7317 0.2066	0.4113*0.8064 0.3317	0.25*0.8064 0.2016	0.7587
C++	0.0565*0.3333 0.0188	0.2823*0.0146 0.0041	0.4113*0.0968 0.0398	0.25*0.0968 0.0242	0.0869

De la Tabla 14 se puede concluir que el lenguaje de programación que más se adapta a los requerimientos y necesidades del proyecto es Python, pues este obtuvo una ponderación final de 0.7585 la cual, es mayor en comparación con la ponderación final obtenida por los otros dos lenguajes de programación, por tal motivo se hizo uso del lenguaje Python para la codificación del sistema de corrección de ruido desarrollado en esta investigación.

4.2.2.2. Selección del Entorno de Desarrollo Integrado.

Con base en las características correspondientes a cada IDE expuestas en la Tabla 6, y teniendo en cuenta las limitaciones de hardware y software expuestas en el primer capítulo de este documento, se consideraron las herramientas ofrecidas por Google colab como primera opción para el desarrollo del sistema de corrección de ruido, esto debido a las ventajas que este ofrece entorno al consumo de GPU y el acceso remoto, brindando la posibilidad de avanzar en el proyecto desde cualquier equipo sin necesidad de instalar programas adicionales a excepción de un navegador web. Por otra parte, no se descarta el uso de las herramientas ofrecidas por PyCharm o VS Code para llevar a cabo pruebas con procesamiento de videos.

4.2.2.3. Selección del Dataset.

Como se mostró anteriormente, los tres dataset contemplados aportan datos de calidad y útiles para el proyecto, otorgando cierto nivel de libertad al momento de realizar el entrenamiento, es por ello que se tendrán en cuenta los tres. Para las pruebas realizadas y mostradas en este documento, se usó principalmente el dataset Berkeley Deep drive. En lo concerniente a los dataset que aportan videos, se tendrán en cuenta para pruebas de video, para lo cual, se plantea la opción de tomar un frame cada determinado tiempo, esto teniendo en cuenta los FPS (fotogramas por segundo) a los cuales trabaja la cámara de un sistema de conducción asistida, al tiempo que se reduce el costo computacional necesario, pues se estaría reduciendo la cantidad de datos a procesar.

4.2.3. Estructura del Software

Este proyecto se desarrolló bajo una estructura dividida en dos fases, inicialmente se encuentra la fase de entrenamiento que tiene por entrada el dataset con los datos predispuestos para esta tarea y otorga a la salida un modelo entrenado en el proceso de denoising orientado a la

conducción asistida, tal como se ve representado en el diagrama de la Figura 10. Dicho diagrama se encuentra conformado principalmente por dos algoritmos, uno para la lectura y procesamiento de los datos, que posteriormente se les adicionara ruido y se almacenaran en dos vectores (imágenes originales e imágenes ruidosas), cabe aclarar que dicho proceso se realiza tanto para las imágenes de entrenamiento como para las de validación. Una vez obtenidos los vectores correspondientes, estos son usados por el segundo algoritmo encargado del proceso de aprendizaje o entrenamiento de la red, a través del cual se obtiene el modelo DCAEAD (Denoising Convolutional Autoencoder for Assited Driving) capaz de llevar a cabo la tarea de corrección de ruido en imágenes.

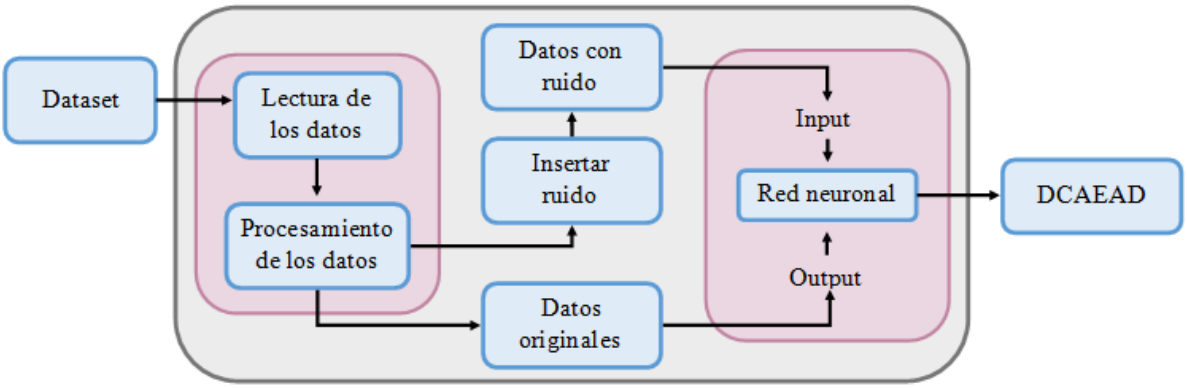


Figura 10. Diagrama de bloques correspondiente a la fase de entrenamiento.

Adicionalmente, es necesario mencionar que, aunque el sistema de corrección de ruido está diseñado para procesar imágenes a color (RGB), este fue entrenado con imágenes de un solo canal (escala de grises), puesto que se evidencio que de esta manera se mejoraban las estimaciones realizadas por el sistema. Teniendo en cuenta lo anterior, durante la implementación se separa la imagen de entrada en sus tres canales y estos se procesan de manera individual, para posteriormente unirse y conformar la imagen estimada.

4.2.3.1. Parámetros de la Red.

Matemáticamente una red neuronal es un modelo no lineal que se puede representar por la siguiente ecuación,

$$\hat{Y} = M_\theta \hat{X} \tag{7}$$

donde, $\hat{Y}$ corresponde a la predicción o resultado obtenido, siendo en este caso las imágenes con corrección de ruido (decodificadas), $\hat{X}$ representa los datos de entrada con los cuales trabajara la red (imágenes ruidosas), θ es la cantidad de parámetros y M_θ consiste en el modelo que se entrena para realizar el proceso de denoising en imágenes.

Para el desarrollo de este proyecto, se trabajó con una red de tipo CNN autoencoder, concatenando las capas de Encoder y Decoder, cada una de estas partes cuenta con 4 capas, las cuales tienen función de activación ReLu y un tamaño de filtro de 3x3 para las convoluciones. En cuanto a las convoluciones, en el Encoder se implementaron dos por capa, esto con la finalidad de dividir un bloque convolucional en dos más pequeños, de esta forma se mantiene la extracción de características de la imagen y se reduce la cantidad de parámetros entrenables, disminuyendo el costo computacional de la red. En cuanto al Decoder solo se aplicó una convolución por capa, además de incluir capas de Dropout con un rate de 0.3 y capas de BathNormalization para reducir los efectos del overfitting y mejorar el proceso de entrenamiento.

Por otra parte, se usaron las capas de keras MaxPooling2D y UpSampling2D, como métodos para reducir y aumentar las dimensiones de los datos de entrada, así mismo, se tuvo en cuenta que, al redimensionar las imágenes, se pudieran realizar cuatro veces el proceso de MaxPooling2D sin perder información, es decir que, al reducir su tamaño a la mitad, sus dimensiones fuesen valores enteros. Finalmente, se modificó el learning rate del optimizador

Adam a 3×10^{-4}, esto permitió disminuir considerablemente las perdidas durante el proceso de entrenamiento. Lo anterior se representa de manera gráfica con el esquema de la Figura 11, donde los recuadros naranjas correspondes a las capas de entrada y salida de la red; los recuadros azules representan el Encoder; los recuadros purpura representan el Decoder; y los recuadros rojos hacen referencia al espacio latente.

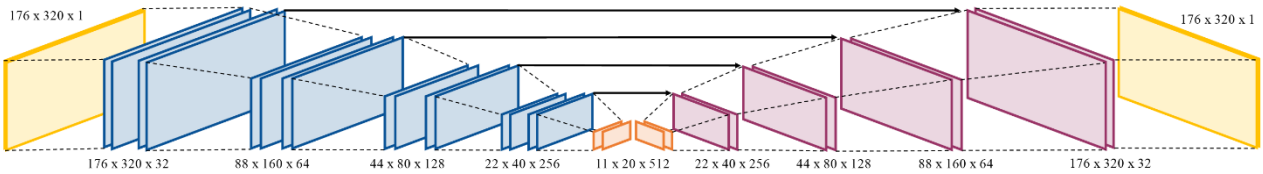


Figura 11. Esquema representativo de la estructura de la red.

4.3. Implementación y Pruebas de Funcionamiento

Teniendo en cuenta el diseño de la red antes realizado, y los recursos disponibles, se procedió a realizar la implementación o desarrollo del sistema de corrección de ruido, llevando a cabo la programación necesaria y aplicando pruebas unitarias que permitieron validar el funcionamiento del sistema frente a los dos tipos de ruido abordados en esta investigación.

4.3.1. Programación en Python

Con base en la estructura establecida para el software, los requerimientos que se desean satisfacer, la asignación de recursos realizada y las limitaciones establecidas anteriormente, se procedió a codificar el algoritmo en lenguaje Python, siguiendo un paradigma de programación estructural.

4.3.1.1. Lectura y Procesamiento de los Datos.

Como se puede ver en el diagrama de bloques de la Figura 10, la programación realizada se encuentra inicialmente conformada por un algoritmo que permite leer y procesar los datos. Para

esta tarea se realizó una función que recibe como parámetros de entrada la cantidad de datos que se desea leer y el path donde se encuentran almacenadas las imágenes. En primer lugar, la función lee el path con ayuda del módulo ‘os’, este devuelve un vector con el nombre de todos los archivos que se encuentran almacenados en la dirección ingresada, en este caso las imágenes de entrenamiento.

Posteriormente, se limita la cantidad de datos según el parámetro ingresado a la función, seguido de esto, se realiza la lectura de cada uno de los datos mediante un ciclo for y con ayuda de las herramientas ofrecidas por la librería OpenCV, en específico, la función *imread()*, dentro de esta función se especifica que las imágenes a leer serán almacenadas en escala de grises.

Luego se realiza un procesamiento a los datos leídos, el cual, consiste en redimensionar las imágenes con la función *cv2.resize()*, para finalmente almacenar todos los datos en un vector, dicho vector se convierte de lista a array de numpy mediante la función numpy.array(), además de normalizar los datos dividiendo entre 255, puesto que se requiere que estos se encuentren en el rango de 0 a 1. Finalmente, la función retorna un vector el cual contiene las imágenes leídas y procesadas. Cabe aclarar que el proceso anterior se realiza tanto con los datos de entrenamiento como con los de validación, y corresponde al pseudocódigo mostrado en el Algoritmo 1.

Algoritmo 1 Lectura y procesamiento de datos

```
procedure read_data(path, quantity)
    data ← [Empty list]
    names_images ← os.listdir(path)[: quantity]
    for img in names_images do
        dir_img ← path + "/" + img
        image ← imread(dir_image, Gray_Escale)
        image ← resize(image, (image size))
        data ← append(image)
    end for
    data ← numpy.array(data, dtype)
    data ← data/255
    return data
end procedure
```

4.3.1.2. Insertar Ruido.

Para el proceso de agregar ruido a los datos, se trabajó con diferentes tipos de ruido, donde cada uno de estos tiene una manera particular de programarse, por este motivo se realizaron diferentes funciones que permiten agregar cada uno de estos tipos de ruido por separado.

- ***Gaussiano.***

Para el proceso de agregar ruido de tipo gaussiano o AWGN se codificó una función que recibe como parámetros el vector que almacena las imágenes a las cuales se les quiere adicionar ruido y la desviación estándar correspondiente al nivel de ruido a adicionar, haciendo uso de la función *random.normal()* ofrecida por la librería numpy la cual, extrae muestras aleatorias de una distribución normal (Gaussiana), dicha función se basa en la fórmula de densidad de la probabilidad para la distribución gaussiana, la cual es (Papoulis, 2002),

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \tag{8}$$

donde, μ es la media o centro de la distribución, que en este caso toma un valor de 0, y σ es la desviación estandar, a través de este ultimo parámetro se controla el nivel de ruido que se inserta a la imagen (NumPy Developers, s.f.). Seguidamente se hace uso de la función *numpyp.clip()* para recortar o limitar los valores de las imágenes ruidosas en un rango de 0 a 1, tal como se habia normalizado en la lectura de los datos, para finalmente retornar un vector que contiene las imágenes ruidosas, proceso que se muestra en el Algoritmo 2.

Algoritmo 2 Proceso de adición de ruido gaussiano

```
procedure add_gaussian_noise(data, noise_factor)
    data_noisy ← data + random.normal(noise_factor)
    data_noisy ← np.clip(data_noisy, value_min, value_max)
    return data_noisy
end procedure
```

- ***Salt & Pepper.***

Para emular la presencia de ruido Salt & Pepper se parte de la probabilidad de aparición de pixeles negros o blancos, para ello, en esta investigación se plantea trabajar un rango de probabilidad de 0 a 1, esto debido a que se trabajó con la función *util.random_noise()* proporcionada por la librería Skimage. Adicionalmente, se estableció una proporción de 0.5, es decir, que existe $Prob/2$ de aparecer pixeles negros o blancos, por tal motivo, si se ingresa un valor de probabilidad de 1, la imagen estará compuesta únicamente de ruido, perdiendo toda la información allí contenida.

Como parámetro de control de la generación de pixeles, se parte de obtener un valor aleatorio con ayuda de la librería numpy, dicho valor se encuentra en el rango de 0 a 1, y define si se adiciona un pixel blanco, negro o se copia el pixel original, esto en función del rango donde se encuentre. Lo anterior se puede representar de manera grafica por el esquema de la Figura 12.

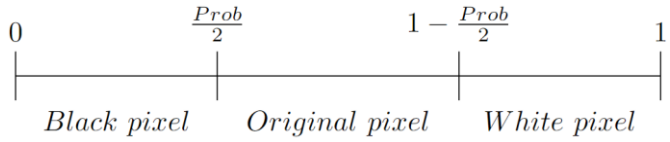


Figura 12. Representación gráfica de la probabilidad de insertar ruido Salt & Pepper.

4.3.1.3. Modelo del Sistema DCAEAD.

Con base en los parametros establecidos en la actividad anterior, se codificó el algoritmo correspondiente a la red neuronal, este consta de tres partes, las cuales son encoder, latent space representation y decoder.

El encoder esta conformado por cuatro capas y estas a su vez consisten en dos convoluciones con los parametros definidos en la fase de diseño y un MaxPooling con un pool_size de 2×2, de igual manera se conforman cada una de las capas, donde el unico parametro que varia de una a

otra es la cantidad de filtros que se aplica en las convoluciones. Finalmente estas se conectan de manera consecutiva, donde la salida de una es la entrada de la siguiente.

El latent space representation consiste en una representación comprimida de los datos que se estan trabajando, para la codificación, se tuvo en cuenta unicamente las convoluciones a realizar con la cantidad de filtros diseñada anteriormente, a su vez, a la entrada de esta capa se recibe la salida obtenida del encoder.

El decoder se encarga de reconstruir los datos a partir de las caracteristicas extraidas por el encoder, por tanto su codificación es bastante similar, pues consta de cuatro capas donde cada una de estas contiene un UpSampling con un tamaño de 2×2, seguido de una capa de dropout con un rate de 0.3, posteriormente se realiza una convolución con un kernel de 3×3, en esta los filtros seran los mismos que se implementaron en el encoder, pero, en orden inverso, luego de esto se agregó una capa de BathNormalization y finalmente se concatena la salida con la capa del encoder que posee las mismas dimensiones.

Por último, se crea el modelo con ayuda de la clase Model proporcionada por keras y se configura el mismo para el entrenamiento, por medio de la función model.compile(), modificando el learning rate y especificando como función de perdidad el error cuadratico medio. Tal como se muestra en el Algoritmo 3.

Algoritmo 3 Modelo de la red

$input \leftarrow keras.input(shape)$
Encoder start
    $x \leftarrow convolution(filters, kernel\ size, padding, activation)(input)$
    $x \leftarrow convolution(filters, kernel\ size, padding, activation)(x)$
    $x \leftarrow MaxPool(pool\ size, padding)(x)$
.
.
.
Encoder end
Latent space representation
    $latent \leftarrow convolution(filters, kernel\ size, padding, activation)(Encoder\ output)$

9: $latent \leftarrow convolution(filters, kernel\ size, padding, activation)(latent\ )$
10: **Decoder start**
11: $x \leftarrow UpSamp(size)(latent)$
12: $x \leftarrow dropout(rate)(x)$
13: $x \leftarrow convolution(filters, kernel\ size, padding, actiation)(x)$
14: $x \leftarrow BathNormalization(\ )(x)$
15: $x \leftarrow Concatenate(\ )(Encoder\_Layer, x)$
.
.
.
16: **Decoder end**
17: $autoencoder \leftarrow model(input, decoder\ output)$
18: $opt \leftarrow optimizer\ Adm(learning\ rate)$
19: $autoencoder \leftarrow compile(opt, loss)$

4.3.1.4. Entrenamiendo del Sistema.

Para entrenar el modelo antes descrito, se implementó el método fit, el cual es uno de los API para entrenamiento de modelos facilitado por Keras, este recibe como parámetros los datos de entrada (imágenes ruidosas), datos de salida (imágenes originales), cantidad de épocas para entrenamiento, número de muestras por iteración (bath_size) y los datos para validación. Como resultado se obtiene el modelo entrenado y los datos correspondientes a las pérdidas de entrenamiento y validación, tal como se aprecia en el Algoritmo 4.

Algoritmo 4 Entrenamiento de la red

1: $history \leftarrow fit\ method, model$
2: $x \leftarrow training\ data\ noisy$
3: $y \leftarrow training\ data$
4: $epochs \leftarrow Número\ de\ muestras\ por\ iteración$
5: $bath\_size \leftarrow Número\ de\ muestras\ por\ iteración$
7: $validation\_data \leftarrow valid\ data\ noisy\ , valid\ data$

4.3.2. Ajuste del Software a los Requerimientos

Una vez planteada la programación correspondiente a cada una de las partes del software, se realizó un ajuste con base en los requerimientos de la problemática. Inicialmente, las imágenes de entrada se deben redimensionar, pues el tamaño de estas es considerablemente grande, por tal

motivo se optó por trabajar con imágenes de 176×320 pixeles. Por otro lado, también se tuvo en cuenta la cantidad de datos a cargar para entrenamiento y validación, puesto que la programación se desarrolló a través de Google Colab y este ofrece un acceso limitado a recursos como RAM y GPU, por tal motivo se decidio trabajar con 1000 imágenes para entrenamiento y 100 para validación.

Finalmente, al momento de ejecutar el comando *auoencoder.fit()*, se estableció una cantidad de 100 épocas de entranmiento y un bath size de 16, pues al aumentar estos valores se saturaban los recursos disponibles por Google Colab y por tanto se cancelaba la ejecución del entrenamiento. A manera de resumen, en la Tabla 15 se pueden apreciar los ajustes y eleciones que se mencionaron anteriormente. Cabe mencionar que los valores de σ y probabilidad usados durante el entrenamiento, se establecieron teniendo en cuenta que presentan una equivalencia en terminos de PSNR y SSIM, es decir, al aplicar las métricas de calidad a las imágenes ruidosas, tengan valores aproximadamente iguales, esto con la finalidad de poder evaluar bajo las mismas condiciones la respuesta del sistema frente a ambos tipos de ruido, optando por entrenar el sistema con $\sigma = 50$ y $Prob = 0{,}0950$.

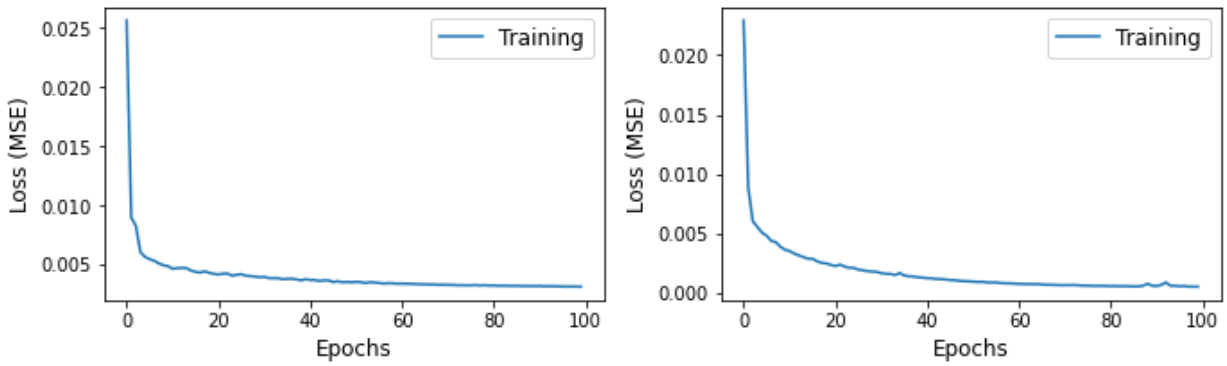
Tabla 15. Parámetros seleccionados para el entrenamiento de la red.

Parámetro	Selección
Tamaño de dato	176×320 pixeles
Cantidad de datos para entrenamiento	1000
Cantidad de datos para validación	100
Epocas de entrenamiento	100

Bath size	16
σ (AWGN)	50
$Prob$ (Salt & Pepper)	0,0950

Una vez entrenado el modelo, la función *model.fit()* permite obtener los datos correspondientes a las pérdidas, tanto para el entrenamiento, como para la validación, dichos valores indican el error que existe entre la imagen estimada y la imagen original, es decir, entre más cercanos se encuentren de cero, mejores serán las estimaciones realizadas por el modelo.

Teniendo en cuenta lo anterior, en la Figura 13 se pueden ver las curvas correspondientes las pérdidas de entrenamiento y validación, obtenidas por los modelos entrenamos con AWGN y Salt & Pepper, notando que el modelo entrenado con Salt & Pepper presenta unas pérdidas de entrenamiento cercanas a $4{,}3 \times 10^{-4}$, siendo más bajas respecto al modelo entrenado con AWGN, donde el valor mínimo fue aproximadamente $3{,}1 \times 10^{-3}$, lo cual repercute directamente en la calidad de los resultados obtenidos, como se podrá ver en los siguientes capítulos de este documento.



a) Con AWGN. b) Con ruido Salt & Pepper.

Figura 13. Perdidas obtenidas para los entrenamientos realizados.

4.3.3. Pruebas Unitarias

Con la finalidad de evaluar el funcionamiento del sistema DCAEAD, se evaluó la respuesta frente a pocas imágenes de entrada. Durante esta evaluación se optó por trabajar con diferentes niveles de ruido, de modo que se pudiera evidenciar de manera específica los resultados obtenidos al procesar ruido Salt & Pepper y AWGN. Para llevar a cabo una evaluación objetiva se aplicaron las métricas de PSNR y SSIM a las imágenes ruidosas y estimadas, teniendo en cuenta que el PSNR tiende a infinito y el SSIM tiene un valor máximo de 1, cuando se trata de la imagen original.

Adicionalmente, fue necesario encontrar una equivalencia entre la escala manejada en AWGN y en Salt & Pepper para cada nivel de ruido manejado, para esto se usó como referencia el PSNR, codificando una función que permitió a partir de un valor de PSNR encontrar el valor de probabilidad que lo generaba, obteniendo los valores mostrados en la Tabla 16.

Tabla 16. Equivalencias entre σ y Prob en función del PSNR producido por la imagen ruidosa respecto a la original.

Nivel de ruido	1	2	3	4	5	6	7
PSNR (dB)	22,39	19,07	16,80	15,10	13,76	12,68	11,79
σ	20	30	40	50	60	70	80
$Prob$	0,0178	0,0380	0,0642	0,0950	0,1291	0,1659	0,2035

En la Taba 16 se puede apreciar que los valores de σ varían en un rango de 20 a 80, valores que pueden parecer un poco elevados, sin embargo, esto se debe a que se está trabajando con imágenes normalizadas, por tanto, el factor de ruido, en este caso la desviación estándar, también

se debe normalizar, dividiendo este valor entre 255, siendo equivalente a agregar el mismo factor de ruido en una imagen sin normalizar, tal como se apreciar en el Anexo 3.

4.3.3.1. Pruebas con Ruido Gaussiano.

Para las pruebas realizadas con ruido gaussiano se seleccionó en primer lugar una imagen del conjunto de datos de test, teniendo en cuenta que contenga ciertos parámetros en los cuales se puede evaluar la funcionalidad del sistema, como lo es la iluminación, la textura, las características que esta posee, entre otros parámetros.

Seguidamente, se adicionó ruido a la imagen, para lo cual se usaron los valores de σ vistos en la Tabla 16, esto con la finalidad de evidenciar la repuesta del sistema a diferentes niveles de ruido, así mismo, para ejemplificar este proceso, en la Figura 14 se muestran tres imágenes correspondientes a un nivel de ruido bajo ($\sigma = 20$), medio ($\sigma = 50$) y alto ($\sigma = 80$), además de esto, se calculó el PSNR y SSIM de cada imagen respecto a la original, notando que ambas métricas disminuyen a medida que el nivel de ruido aumenta en la imagen.

Finalmente, se procesaron las imágenes con el sistema DCAEAD, obteniendo así las imágenes vistas en la Figura 14, en estas se puede apreciar la mejoría realizada en cuanto a la calidad, como se comprueba con las métricas, pues en los tres casos evaluados, se presenta un incremento tanto del PSNR como del SSIM en comparación con las imágenes ruidosas. Además de lo anterior, se pudo notar que el sistema dio una mejor respuesta cuando se evaluó con el mismo nivel de ruido usado durante el entrenamiento ($\sigma = 50$).

a) Imagen original

22,85 dB / 0,5735

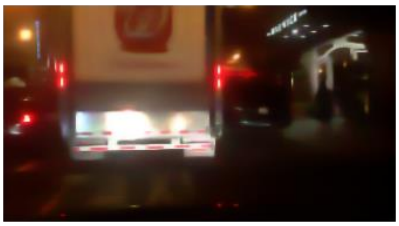
b) Ruido a $\sigma = 20$

15,76 dB / 0,2484

c) Ruido a $\sigma = 50$

11,28 dB / 0,1378

d) Ruido a $\sigma = 80$



26,21 dB / 0,7707

e) Estimación para $\sigma = 20$

27,07 dB / 0,8651

f) Estimación para $\sigma = 50$

20,06 dB / 0,5114

g) Estimación para $\sigma = 80$

Figura 14. Pruebas realizadas con el modelo entrenado para corregir AWGN.

De las estimaciones mostradas en la Figura 14, se puede ver que, para el valor de desviación estándar más alto $\sigma = 80$, la imagen se aprecia con ruido, pues, aunque si se realiza una mejora en la calidad, el sistema no es capaz de procesar de manera eficiente niveles de AWGN tan altos.

4.3.3.2. Pruebas con Ruido Salt & Pepper.

De manera similar al análisis realizado con el ruido AWGN, se realizaron pruebas del sistema DCAEAD entrenado con imágenes que contenían ruido Salt & Pepper. Para esto, se

seleccionó una imagen de prueba y se le adiciono ruido Salt & Pepper a tres valores de probabilidad, los cuales fueron, $Prob = 0{,}0178$; $Prob = 0{,}0950$ y $Prob = 0{,}2035$, esto teniendo en cuenta que, según lo visto en la Tabla 16, son las equivalencias de los niveles de ruido usados durante las pruebas con AWGN, lo cual se puede comprobar al calcular el PSNR, notando que este es aproximado al obtenido por las imágenes con AWGN.

Posteriormente, se procesaron las imágenes, llevando a cabo la corrección de ruido en cada una de ellas, obteniendo de esta forma las estimaciones mostradas en la Figura 15. Adicionalmente, se calcularon los valores de PSNR y SSIM para las imágenes ruidosas y las estimaciones, notando que en las tres pruebas realizadas se obtuvo una ganancia media de 18,07 dB en el PSNR, representando un incremento en la calidad de la imagen, tal como se puede apreciar de manera visual. En cuanto a los valores de SSIM, estos presentaron un incremento considerable respecto a lo obtenido a partir de las imágenes ruidosas, manteniendo valores por encima de 0,93.

Finalmente, de las pruebas realizadas con AWGN y Salt & Pepper, correspondientes a las imágenes de la Figura 14 y Figura 15 respectivamente, se puede notar que el sistema DCAEAD muestra superioridad al momento de procesar imágenes con ruido Salt & Pepper, incrementando considerablemente los valores de PSNR y SSIM, comportamiento que se pudo redecir desde la fase de entrenamiento, donde se notó que las perdidas disminuyeron más al momento de entrenar con imágenes que contenían ruido Salt & Pepper. Lo anterior, se atribuye principalmente a la naturaleza de cada tipo de ruido, puesto que en Salt & Pepper, se reemplazan algunos pixeles, mientras que en AWGN, estos se ven distorsionados, dificultando el proceso de corrección del mismo.

a) Imagen original

22,85 dB / 0,5735

b) Ruido a $Prob = 0{,}0178$

15,21 dB / 0,3730

c) Ruido a $Prob = 0{,}0950$

11,86 dB / 0,2122

d) Ruido a $Prob = 0{,}2035$

39,81 dB / 0,9953

e) Estimación realizada para $Prob = 0{,}0178$

34,58 dB / 0,9794

f) Estimación realizada para $Prob = 0{,}0950$

29,74 dB / 0,9316

g) Estimación realizada para $Prob = 0{,}2035$

Figura 15. Pruebas realizadas con el modelo entrenado para corregir Salt & Pepper.

4.4. Evaluación del Sistema

Una vez realizadas las pruebas unitarias y validado el funcionamiento del sistema de corrección de ruido, se hizo necesario llevar a cabo una serie de pruebas que permitieron comprobar que el modelo no memorizo las características de una sola imagen, sino que este aprendió a corregir el ruido, independientemente de la imagen de entrada. Por otra parte,

teniendo en cuenta los requerimientos del proyecto, es necesario verificar el costo computacional del mismo, para se tendrá como base el tiempo de procesamiento del sistema.

4.4.1. Análisis Estadístico

Para llevar a cabo una evaluación más completa del funcionamiento del sistema DCAEAD, se realizaron pruebas con un mayor número de imágenes de entrada, en específico con 30 imágenes seleccionadas del set de datos. Esto permitió recolectar datos estadísticos de PSNR y SSIM, ayudando a demostrar que el modelo es capaz de responder satisfactoriamente frente a diferentes imágenes. Realizando este proceso de evaluación con AWGN y ruido Salt & Pepper.

4.4.1.1. Resultados con AWGN.

Inicialmente se realizan las pruebas a partir de imágenes con adición de AWGN, implementando la escala de desviación estándar mostrada en la Tabla 16, a partir de estas se calcularon las imágenes estimadas y se aplicaron las métricas de PSNR y SSIM, de modo que permitió evaluar de forma objetiva el funcionamiento del sistema.

Dado que se cuenta con un total de 30 imágenes para realizar las pruebas, estas se procesan y se toman los promedios de manera global, es decir, se calcula el valor de PSNR y SSIM de cada una de las imágenes y posteriormente se promedian para obtener un único valor por cada nivel de desviación estándar, realizando el mismo proceso para cada uno de los niveles de ruido con los cuales se evaluó el sistema.

Una vez realizado el proceso anteriormente mencionado, se grafican los valores correspondientes al PSNR de las imágenes estimadas junto con la curva correspondiente al PSNR de las imágenes ruidosas, como se puede ver en la Figura 16, esto con la finalidad de poder visualizar la ganancia que se realiza en esta métrica gracias a la aplicación del sistema DCAEAD.

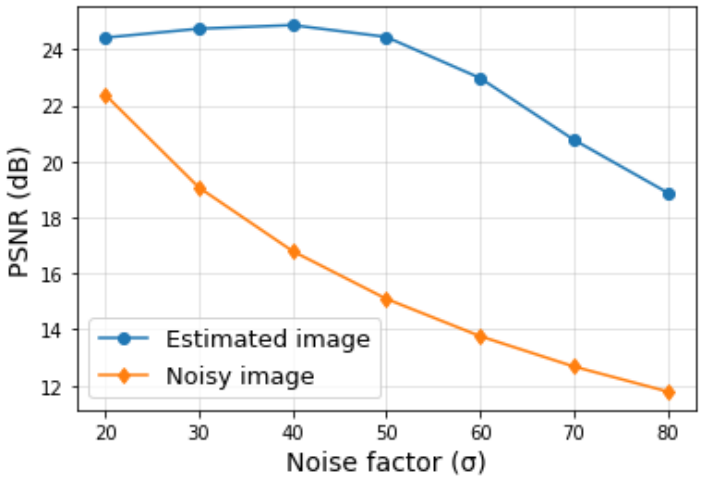


Figura 16. Curvas correspondientes a los valores promedio de PSNR obtenidos a partir de las imágenes estimadas y las imágenes con AWGN.

De las curvas mostradas en la Figura 16 se puede evidenciar que el PSNR tiende a dar mejores resultados para niveles bajos de σ $(20 - 50)$, donde se obtienen valores superiores a 24 dB, notando un leve incremento al inicio de la curva, sin embargo, posteriormente decrementa considerablemente hasta llegar a valores cercanos a 19 dB, alcanzando el máximo de PSNR para $\sigma = 40$, donde se obtiene una ganancia aproximada de 8 dB.

De igual manera se grafican los valores promedios correspondientes a la métrica de SSIM, tal como se muestra en la curva de la Figura 17. Dichas curvas poseen un comportamiento similar al mostrado con el PSNR, pues para valores bajos de $\sigma = 20$ hasta $\sigma = 50$, se obtiene un SSIM superior a 0.8, valor que posteriormente decrece a medida que se aumenta el nivel de ruido, comportamiento que es consistente con lo antes visto en las pruebas unitarias, pues la mejor respuesta del sistema se obtuvo para valores de $\sigma = 50$, donde se pudo apreciar un incremento de aproximadamente 0,5.

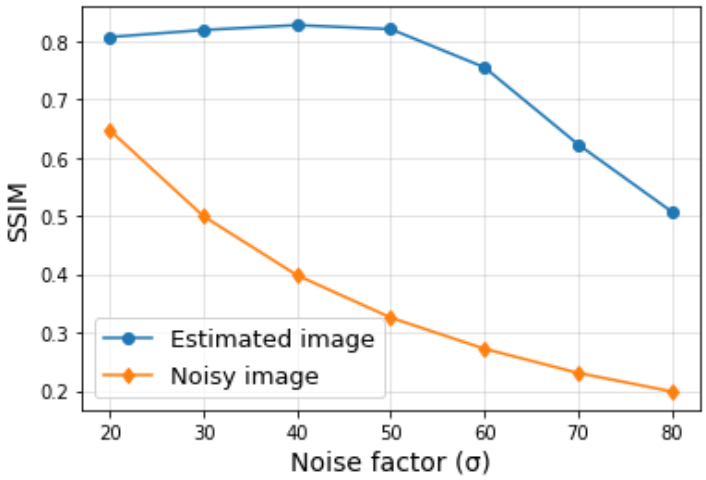


Figura 17. Curvas correspondientes a los valores promedio de SSIM obtenidos a partir de las imágenes estimadas y las imágenes con AWGN.

Con la finalidad de brindar mayor información en cuanto a los resultados analizados, en el Anexo 4, se pueden ver los valores promedio de PSNR y SSIM correspondientes a las gráficas de la Figura 16 y Figura 17, en estas se puede notar de manera precisa, el incremento en las métricas de calidad luego de procesar las imágenes con el sistema DCAEAD.

4.4.1.2. Resultados con Ruido Salt & Pepper.

De manera similar al proceso mostrado anteriormente, se llevaron a cabo las pruebas con el modelo entrenado para corregir ruido Salt & Pepper. Para esto se evaluó la respuesta frente a 7 niveles de ruido (Probabilidad), los cuales corresponden a los mostrados en la Tabla 16. Dichas pruebas se realizaron con las 30 imágenes seleccionadas anteriormente, esto con la finalidad de evaluar el sistema bajo las mismas condiciones y poder contrastar los resultados obtenidos frente a cada tipo de ruido.

Luego de procesar las imágenes con el sistema DCAEAD, se calcularon los valores promedio de PSNR para las imágenes estimadas y las imágenes ruidosas respecto a la original, valores que corresponden a las curvas mostradas en la Figura 18. De dichas curvas se puede

notar que el PSNR correspondiente a las estimaciones se incrementó considerablemente respecto al obtenido por las imágenes ruidosas, alcanzando un valor superior a 35 dB para imágenes que contenían niveles de ruido bajos ($Prob = 0{,}0178$). Posteriormente, los valores de PSNR decrementan a medida que se aumenta el nivel de ruido, llegando a un valor mínimo de aproximadamente 28 dB para un valor de $Prob = 0{,}2035$, el cual continúa siendo aceptable teniendo en cuenta que realizo un incremento de 16 dB sobre el valor de la imagen con ruido.

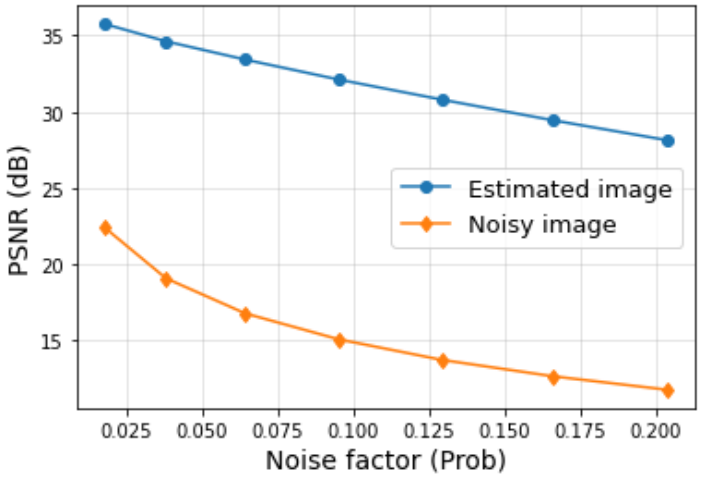


Figura 18. Curvas correspondientes a los valores promedio de PSNR obtenidos a partir de las imágenes estimadas y las imágenes con ruido Salt & Pepper.

En cuanto a los resultados obtenidos en función del SSIM, estos reflejan el comportamiento mostrado en las curvas de PSNR, como se puede ver en la Figura 19, notando que los valores de SSIM se mantienen por encima de 0,9 para todos los niveles de ruido evaluados, obteniendo el nivel máximo para el valor de probabilidad más bajo ($Prob = 0{,}0178$).

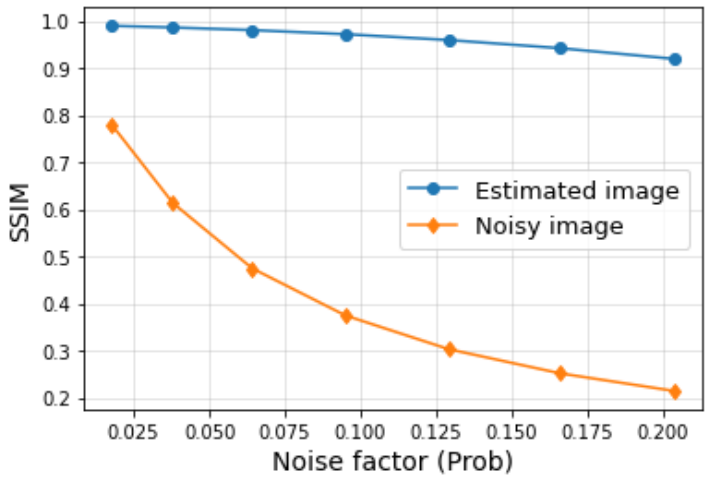


Figura 19. Curvas correspondientes a los valores promedio de SSIM obtenidos a partir de las imágenes estimadas y las imágenes con ruido Salt & Pepper.

Por último, con la finalidad de complementar la información analizada anteriormente, se adjuntan en el Anexo 5 los valores correspondientes a las curvas vistas en la Figura 18 y Figura 19. Dichos valores corresponden al PSNR y SSIM obtenidos a partir de las imágenes estimadas y ruidosas respecto a la original, esto luego de evaluar la respuesta del sistema DCAEAD entrenado para corregir ruido Salt & Pepper.

4.4.1.3. Ganancia del Sistema en Función de PSNR y SSIM.

Además de las curvas mostradas anteriormente, se calculó la ganancia obtenida por el sistema frente a cada nivel de ruido, esta corresponde al incremento en los valores de PSNR y SSIM, es decir, las métricas obtenidas por las imágenes estimadas, menos los valores obtenidos a partir de las imágenes con ruido AWGN y Salt & Pepper respectivamente.

Inicialmente se calculó la ganancia en términos del PSNR, obteniendo las curvas mostradas en la Figura 20. De allí se puede ver que, aunque ambas pruebas muestran un comportamiento similar, el modelo entrenado para corregir imágenes con ruido Salt & Pepper alcanzo una

ganancia máxima de aproximadamente 17 dB, demostrando superioridad sobre el entrenado con AWGN, pues este último una ganancia máxima de aproximadamente 9 dB.

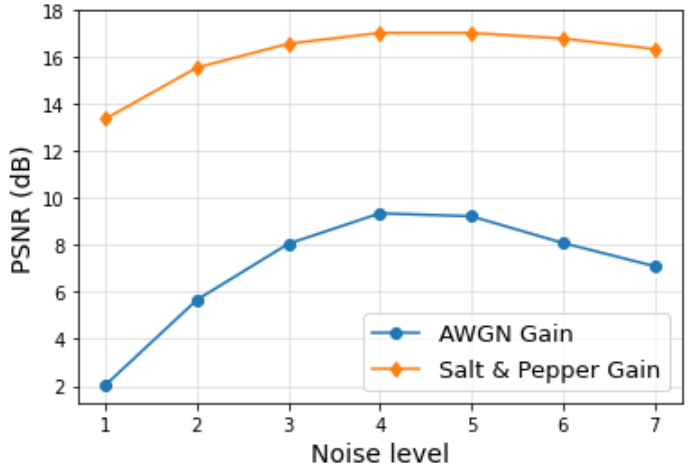


Figura 20. Ganancia del sistema DCAEAD en términos de PSNR.

Por otro lado, la ganancia en términos de SSIM demostró un comportamiento similar durante los 4 primeros niveles de ruido, donde incremento hasta 0,5 para AWGN y 0,6 para Salt & Pepper. Posteriormente, a partir del 5 nivel de ruido, se puede notar que el SSIM de las imágenes estimadas a partir de AWGN empieza a decrecer hasta aproximadamente 0,3. Por otro lado, los valores correspondientes al ruido Salt & Pepper continúan incrementando, llegando a obtener una ganancia de 0,7 sobre la imagen ruidosa, tal como se puede ver en las curvas de la Figura 21.

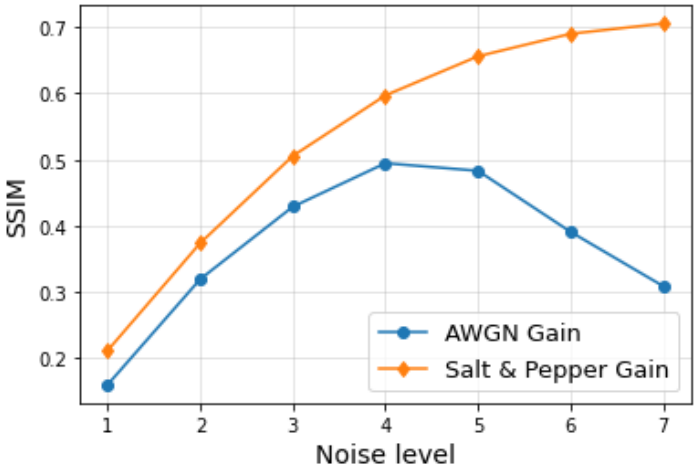


Figura 21. Ganancia del sistema DCAEAD en términos de SSIM.

4.4.1.4. Contrastar Resultados.

Finalmente, se compararon los resultados obtenidos con respecto a los valores de PSNR mostrados durante la identificación de requerimientos (Tabla 4). Para poder evaluar el sistema bajo las mismas condiciones, se tomaron los resultados a 3 niveles de σ en el caso de AWGN y sus correspondientes equivalencias de $Prob$ en el caso de ruido Salt & Pepper, tal como se muestra en la Tabla 17. De allí se puede notar que el sistema entrenado para corregir AWGN, obtuvo resultados inferiores a lo planteado en los requerimientos en las pruebas con $\sigma = 30$ y $\sigma = 70$, mientras que para las pruebas realizadas con $\sigma = 50$ el SSIM fue superior, aunque, el PSNR está 2 dB por debajo. En cuanto a los resultados obtenidos a partir del sistema entrenado para corregir ruido Salt & Pepper, se puede ver que tanto el PSNR como el SSIM es superior a lo planteado en los requerimientos.

Tabla 17. Evaluación de los resultados promedio obtenidos con base en los requerimientos.

	Factor de ruido (σ / $Prob$)					
	30 / 0,0380		50 / 0,0950		70 / 0,1659	
	PSNR (dB)	SSIM	PSNR (dB)	SSIM	PSNR (dB)	SSIM
Requerimientos	29,11	0.8632	26,48	0,7956	24,80	0,7487
AWGN	24,74	0,8189	24,45	0,8203	30,78	0,6226
Salt & Pepper	34,62	0,9862	32,12	0,9719	28,15	0,9195

4.4.2. Análisis de Costo Computacional

Uno de los requerimientos del proyecto se centró en el tiempo de ejecución, esto debido a que el sistema se desarrolló para procesar imágenes de conducción asistida, siendo necesario

obtener una respuesta en tiempo real, pues debido al campo de acción, el tiempo de procesamiento hace la diferencia entre poder prevenir o no un posible accidente.

Al momento de realizar la evaluación del tiempo de ejecución fue necesario tener en cuenta el equipo de cómputo con el cual se realizaron las pruebas, pues ofreció un limitado acceso a recursos como GPU y RAM, recurriendo a Google Colab donde, aunque se tiene acceso a GPU y RAM, estos recursos también se ven limitados, afectando directamente el tiempo de ejecución del algoritmo. Por otra parte, se debe tener en cuenta que el modelo se entrenó con imágenes de un solo canal, sin embargo, se está implementando para procesar imágenes a color. Para esto se toma la imagen de entrada y se separa en sus tres canales (R, G, B), luego se procesa cada canal por separado y finalmente se unen los tres para conformar la imagen estimada. Por tal motivo, durante el proceso de corrección de ruido, el sistema DCAEAD tiene que llevar a cabo 3 estimaciones.

Teniendo en cuenta lo anterior, se evaluó el tiempo de respuesta del sistema DCAEAD en función de la cantidad de datos de entrada. Para llevar a cabo estas pruebas, se fue incrementando la cantidad de imágenes del vector Test, de 1 a 30, y se midió el tiempo que le tardaba el modelo en estimar los tres canales de la imagen, obteniendo de esta forma la curva vista en la Figura 22.

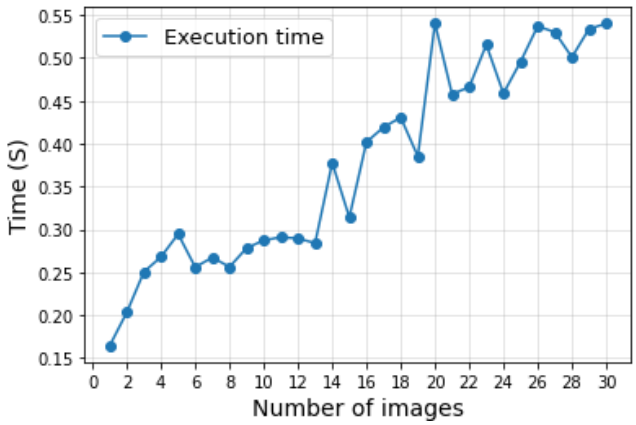


Figura 22. Tiempo de ejecución del sistema en función de la cantidad de datos a procesar.

4.4.3. Retroalimentación

Los resultados mostrados anteriormente fueron producto del desarrollo de un sistema de corrección de ruido denominado DCAEAD. Sin embargo, los valores de PSNR y SSIM obtenidos a partir de las estimaciones (Anexo 4 y Anexo 5), inicialmente eran inferiores (Torres, y otros, 2022), para llegar a obtener los resultados aquí mostrados se llevaron a cabo una serie de cambios o modificaciones en el sistema, producto de la retroalimentación realizada.

Uno de las modificaciones realizadas en el sistema fue reemplazar una capa convolucional de n cantidad de filtros, por dos capas con $n/2$ filtros, esto permitió disminuir la cantidad de parámetros y, disminuir los recursos necesarios para ejecutar el sistema, además de mantener el proceso de extracción de características con la misma cantidad de filtros en total.

Por otra parte, la red inicialmente tenía un tamaño de entrada de $360 \times 640 \times 3$, sin embargo, esto limitaba considerablemente la cantidad de datos de entrenamiento y validación que se podían utilizar sin llegar a saturar los recursos ofrecidos por Colab, permitiendo usar 370 imágenes de entrenamiento, 70 de validación y un Bath size de 8. Por tal motivo, se optó por cambia el tamaño de las imágenes de entrada, redimensionándolas a $176 \times 320 \times 1$, esto permitió importar 1000 imágenes del entrenamiento, 100 de validación y duplicar el Bath size.

Luego de modificar el tamaño de las imágenes de entrada, se pudo notar una un decremento en los recursos consumidos por el sistema, viendo la oportunidad de adicionar una capa de más en el Encoder y en el Decoder, permitiendo realizar un procesamiento más completo de las imágenes, extrayendo una mayor cantidad de información durante la fase del Encoder y realizando una mejor reconstrucción en el Decoder. Adicionalmente, las capas de Dropout y BathNormalization se removieron del Encoder y se incluyeron únicamente en el Decoder.

Teniendo en cuenta que el sistema DCAEAD inicialmente estaba basado en una arquitectura autoencoder, se decidió concatenar cada capa del Encoder con la correspondiente capa del Decoder que posee las mismas dimensiones, esto permitió extraer una mayor cantidad de características de la imagen de entrada. Adicionalmente, se modificó el learning rate, puesto que este era de $2{,}5 \times 10^{-4}$ y se incrementó a 3×10^{-4}. Estos cambios contribuyeron a mejorar el proceso de aprendizaje de la red, lo cual se vio reflejado en las pérdidas de entrenamiento y validación.

Por último, se optó por trabajar con escalas de ruido normalizadas, pues inicialmente se estaban manejando valores de $0 < \sigma \leq 1$, sin embargo, en la revisión de la literatura se notó que esta escala es normalizada y se trabaja en un rango de $0 < \sigma \leq 100$. Por lo anterior se decidió realizar el cambio de escala y calcular las correspondientes equivalencias de $Prob$, permitiendo contrastar los resultados con otras investigaciones.

4.5. Divulgación de Resultados

Como objetivo final de este trabajo de investigación se llevó a cabo la divulgación de resultados a través de participaciones en eventos de divulgación científica, esto con la finalidad de presentar los avances realizados en el campo de la corrección de ruido en imágenes de conducción asistida, de modo que represente un aporte para futuras investigaciones en este tema de investigación.

En primer lugar, se presentó la ponencia titulada “DCAEAD: Denoising System for Assisted Driving Images Applying an Autoencoder Architecture” en el evento IEEE International Conference on Automation / XXV Congress of the Chilean Association of Automatic Control (ICA–ACCA) (Ver Anexo 6). Dicho evento se llevó a cabo se realizó de manera virtual del 24 al 28 de octubre de 2022, en la ciudad de Curicó, Chile. Adicionalmente, la participación en el

evento brindaba la oportunidad de realizar la publicación de un artículo de investigación en IEEE Xplore. Por tal motivo, se sometió el artículo a proceso de evaluación por pares, siendo aprobado por los evaluadores y publicado en enero de 2023 (Ver Anexo 7).

Finalmente, como producto de esta investigación, se presentó en la IX Semana Internacional de Ciencia, Tecnología e Innovación (SICTeI 2022) la ponencia titulada “Methodology Used for the Development of Image Noise Correction Systems Based on Deep Learning” (Ver Anexo 8), dicho evento se llevó a cabo de manera presencial del 29 de noviembre al 02 de diciembre de 2022, en la Universidad Francisco de Paula Santander, Cúcuta, Colombia.

Finalmente, en la Tabla 18 se puede ver un resumen de las participaciones en eventos de divulgación científica asociadas al desarrollo del sistema DCAEAD. Cabe mencionar que en este resumen no se incluyeron tres artículos de investigación en proceso de revisión.

Tabla 18. Resumen de participaciones en eventos de divulgación científica.

Titulo	Tipo de producto	Base de datos	Anexo
DCAEAD: Denoising system for assisted driving images applying an autoencoder architecture	Ponencia	IEEE Xplore	Anexo 6
DCAEAD: Denoising system for assisted driving images applying an autoencoder architecture	Articulo científico	IEEE Xplore	Anexo 7
Methodology Used for the Development of Image Noise Correction Systems Based on Deep Learning	Ponencia	Memorias SICTeI 2022	Anexo 8

5. Conclusiones

Existen diferentes herramientas que facilitan el procesamiento digital de imágenes y en específico la corrección de ruido, siendo claro ejemplo de estas los filtros, pero, este tipo de herramientas pierden funcionalidad al incrementar el nivel de ruido a procesar, por tal motivo, se han venido desarrollando alternativas basadas en Deep Learning que ayudan a realizar el proceso de corrección de ruido en imágenes donde los filtros son ineficientes, abordando principalmente ruido Salt & Pepper, AWGN, Speckle, entre otros, y respondiendo a problemáticas en áreas como la medicina, la industria o los sistemas automotrices. Sin embargo, luego de revisar la literatura se pudo evidenciar que no se ha profundizado este tema de estudio para imágenes de conducción asistida, donde se ve la necesidad de procesar imágenes con el menor ruido posible. Adicionalmente, se pudo evidenciar la importancia de identificar los requerimientos a tener en cuenta al trabajar con procesamiento digital de imágenes en el campo de la conducción asistida, permitiendo establecer delimitaciones durante el proceso de investigación, de este modo se pudo plantear una metodología estructurada que permitió dar respuesta a los objetivos planteados.

Teniendo en cuenta lo anterior, esta investigación se centró en desarrollar un sistema de corrección de ruido en imágenes usadas en procesos de conducción asistida (DCAEAD). Para conseguir esto, fue necesario diseñar el algoritmo de Deep Learning basado en una modificación de redes neuronales convolucionales de tipo autoencoder. Evidenciando que esta estructura en específico, aunque posee aproximadamente el doble de parámetros (6’667.905) respecto al modelo de red inicial (3’099.907), favorece el proceso de extracción de características de la imagen ruidosa, como bordes, texturas, entre otros, permitiendo reconstruir los datos a partir de dicha información, reduciendo de esta forma el ruido presente en la imagen.

En cuanto al desarrollo del algoritmo se vio la importancia de acceder a recursos de libre acceso, en este caso específico se recurrió al lenguaje de programación Python por las grandes ventajas que este ofrece en cuanto a variedad de librerías, funciones para procesamiento de imágenes y herramientas para trabajar con redes neuronales artificiales, además de ser compatible con Google Colab, brindando acceso a GPU, lo cual aceleró los procesos de entrenamiento y ejecución del sistema de corrección de ruido.

Durante el proceso de entrenamiento del sistema DCAEAD se aplicó la medida del error medio cuadrático (MSE) como función de perdida, debido a las propiedades de entrenamiento, donde el principal parámetro de control de los pesos y sesgos de la red se basó en disminuir el valor de la función de perdida, llegando a obtener valores cercanos a $4{,}3 \times 10^{-4}$ para el modelo entrenado con ruido Salt & Pepper, y aproximadamente $3{,}1 \times 10^{-3}$ para el modelo entrenado con AWGN. Adicionalmente, la realización de pruebas unitarias permitió comprobar la funcionalidad del sistema DCAEAD frente a imágenes con ruido Salt & Pepper y AWGN, permitiendo llevar a cabo un proceso de retroalimentación que contribuyó a mejorar los resultados obtenidos.

El sistema de corrección de ruido desarrollado en esta investigación (DCAEAD) mostró un mayor nivel de funcionalidad para procesar ruido Salt & Pepper, permitiendo obtener imágenes de hasta un 99% de similitud estructural respecto a la imagen original, respuesta que se ve reflejada en el PSNR donde se incrementó hasta 17 dB por encima del valor con ruido. Por otra parte, los resultados frente a AWGN, aunque muestran una mejora en la calidad de las imágenes, obteniendo hasta un 83% de similitud estructural, la ganancia en términos de PSNR (9 dB) es considerablemente inferior a lo obtenido a partir de ruido Salt & Pepper, factor que se atribuye principalmente a la forma en que el ruido se manifiesta en la imagen.

Referencias Bibliográficas

Agencia Nacional de Seguridad Vial. (s.f.). *Cifras año en curso*. Recuperado el Enero de 2023, de https://ansv.gov.co/es/observatorio/estad%C3%ADsticas/cifras-ano-en-curso

Ahn, B. C. (2017). Block-Matching Convolutional Neural Network for Image Denoising. *Computer Vision and Pattern Recognition*.

Anwar, S., & Barnes, N. (2019). Real Image Denoising With Feature Attention. *Proceedings of the IEEE/CVF International Conference on Computer Vision.* Seoul, Korea.

Ashfahani, A., Pratama, M., Lughofer, E., & Ong, Y.-S. (2020). DEVDAN: Deep evolving denoising autoencoder. *Neurocomputing, 390,* 297-314.

Atienza, R. (2018). *Advanced Deep Learning with Keras: Apply deep learning techniques, autoencoders, GANs, variational autoencoders, deep reinforcement learning, policy gradients, and more.* Packt Publishing Ltd.

Bajaj, K., Kumar, D., & Aquib, M. (2020). Autoencoders Based Deep Learner for Image Denoising. *Procedia Computer Science, 171,* 1535-1541.

Bank, D., Koenigstein, N., & Giryes, R. (2021). Autoencoders. arXiv.

Barradas, C. (2020). Estudio y diseño de un proceso de monitorización aplicado aun motor eléctrico por medio de la inteligencia artificial. Barcelona: Universidad politècnica de Catalunya.

Benavides, Ó. (2004). La innovación tecnológica desde una perspectiva evolutiva. *Cuadernos de economía, 23*(41), 49-70.

Burguillos, C. (2016). Algoritmos de lenguaje máquina para la detección de vehículos mediante cámaras infrarrojas. Universidad Carlos III de Madrid.

C++ Foundation. (s.f.). *News, Status & Discussion about Standard C++*. Recuperado el Septiembre de 2021, de https://isocpp.org/

Cano, L., & Bautista, R. (2019). Alcance del derecho al olvido en el tratamiento de datos personales en Colombia. *Verba luris*(41), 45-63.

Caparrós, A. (1999). El comportamiento humano en conducción: factores perceptivos, cognitivos y de respuesta. *Cognición y Psicología Aplicada a la conducción de vehículos* .

Charte, F. (2021). Autoencoders ¿Qué son, para qué sirven y cómo funcionan? Jaén: DaSCI.

Christian, S., Wei, L., Yangqing, J., Pierre, S., Scott, R., Dragomir, A., . . . Andrew, R. (2015). Going deeper with convolutions. *Conference on Computer Vision and Pattern Recognition (CVPR).* Boston, USA.

Congreso de Colombia. (1993). *Ley 44 de 1993.* Recuperado el Enero de 2023, de https://www.funcionpublica.gov.co/eva/gestornormativo/norma.php?i=3429#:~:text=%22Los%20derechos%20consagrados%20a%20favor,a%20partir%20de%20su%20muerte.

Continental. (2020). *Blind Spot Detection.* Recuperado el 6 de Mayo de 2021, de http://www.continental-corporation.com

Contreras, G., Pabón, J., García, H., Rojas, F., & Arguello, H. (2021). Correction of Designed Compressive Spectral Imaging Measurements Using a Deep Learning-Based Method. *XXIII Symposium on Image, Signal Processing and Artificial Vision (STSIVA).* Popayán, Colombia.

Dabov, K., Foi, A., Katkovnik, V., & Egiazarian, K. (2007). Image Denoising by Sparse 3-D Transform-Domain Collaborative Filtering. *IEEE Transactions on Image Processing , 16*(8), 2080 - 2095.

de los Reyes, Á. (2019). Análisis de rendimiento de eutoencoders en entornos big data. Madrid: Universidad Carlos III.

Doulamis, N., Doulamis, A., Voulodimos, A., & Protopapadakis, E. (2018). Deep Learning for Computer Vision: A Brief Review. *Computational intelligence and neuroscience, 2018*.

Elad, M., & Aharon, M. (2006). Image Denoising Via Sparse and Redundant Representations Over Learned Dictionaries. *IEEE Transactions on Image Processing, 15*(12), 3736-3745.

Esqueda, J. (2002). Fundamentos de Procesamiento de Imágenes. *CONATEC.*

Fernández, M. (1996). *Señales aleatorias y ruido.* Valladolid: E.T.S. de Ingenieros de Telecomunicación.

Florez, M. (2001). Los sistemas inteligentes de transporte ITS. *Ciencia e Ingeniería Neogranadina*(10), 39-45.

Google. (s.f.). *What is Colaboratory?* Recuperado el Septiembre de 2021, de https://colab.research.google.com/

Gu, S., Zhang, L., Zuo, W., & Feng, X. (2014). Weighted Nuclear Norm Minimization with Application to Image Denoising. *IEEE Conference on Computer Vision and Pattern Recognition.* Columbus, OH, USA.

Gupta, A., Anpalagan, A., Guan, L., & Shaharyar, A. (2021). Deep learning for object detection and scene perception in self-driving cars: Survey, challenges, and open issues. *Array, 100057, 10*.

Hashisho, Y., Albadawi, M., Krause, T., & Freiherr, U. (2019). Underwater Color Restoration Using U-Net Denoising Autoencoder. *11th International Symposium on Image and Signal Processing and Analysis (ISPA).* Dubrovnik, Croatia.

Horé, A., & Ziou, D. (2010). Image Quality Metrics: PSNR vs. SSIM. *20th International Conference on Pattern Recognition.* Istanbul, Turkey.

Huynh-Thu, Q., & Mohammed, G. (2008). Scope of validity of PSNR in image/video quality assessment. *Electronics letters, 44*(13), 800-801.

Jain, P., & Tyagi, V. (2016). A survey of edge-preserving image denoising methods. *Information Systems Frontiers, 18*, 159-170.

Jarrett, K., Kavukcuoglu, K., Ranzato, A., & LeCun, Y. (2009). What is the best multi-stage architecture for object recognition? *12th International Conference on Computer Vision.* Kyoto, Japon.

JetBrains. (s.f.). *PyCharm*. Recuperado el Sepptiembre de 2021, de www.jetbrains.com/es-es/pycharm/

Kotevski, Z., & Mitrevski, P. (2009). Experimental Comparison of PSNR and SSIM Metrics for Video Quality Estimation. *ICT Innovations*, 357-366.

Larrue, T., Li, Y., Meng, X., & Han, C.-M. (2018). Denoising Videos with Convolutional Autoencoders. *Proceedings of ACM Conference.* New York: University of Maryland.

Lau, M., & Hann, K. (2018). Review of Adaptive Activation Function in Deep Neural Network. *Conference on Biomedical Engineering and Sciences (IECBES).* Sarawak, Malasia.

Leal, J. (2021). Características de la infraestructura que pueden favorecer la conducción asistida y automatizada. *Revista digital del CEDEX*(197), 106-117.

León, A., Bermeo, J., Paredes, J., & Torres, H. (2020). Una revisión de las métricas aplicadas en el procesamiento de imágenes. *RECIMUNDO: Revista Científica de la Investigación y el Conocimiento, 4*(3), 267-273.

Li, L., Yu, X., Jin, Z., Zhao, Z., Zhuang, X., & Liu, Z. (2020). FDnCNN-based image denoising for multi-labfel localization measurement. *Measurement, 152*.

Lore, K., Akintayo, A., & Sarkar, S. (2017). LLNet: A deep autoencoder approach to natural low-light image enhancement. *Pattern Recognition, 61*, 650-662.

Lv, H., & Li, H. (2021). Denoising method of low illumination underwater motion image based on improved canny. *Microprocessors and microsystems, 82*.

Majumdar, A. (2019). Blind Denoising Autoencoder. *IEEE Transactions on Neural Networks and Learning Systems, 30*(1), 312-317.

Marreiros, A., Daunizeau, J., Kiebel, S., & Friston, K. (2008). Population dynamics: Variance and the sigmoid activation function. *NeuroImage, 42*(1), 147-157.

MathWorks. (s.f.). *Métricas de calidad de imagen*. Recuperado el 22 de Junio de 2021, de es.mathworks.com

MathWorks. (s.f.). *Soporte para ISO 26262 en MATLAB y Simulink*. Recuperado el 3 de Julio de 2021, de es.mathworks.com

McCulloch, W., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics, 5*(4), 115–133.

McGunnigle, L. (2020). Berkely deep drive.

Menegazzo, J. (2021). PVS - Passive Vehicular Sensors Datasets.

Mérida, M. (2012). Reconocimiento biométrico basado en imágenes de huellas palmares. Universidad autónoma de Madrid.

Microsoft. (s.f.). *Visual Studio Code*. Recuperado el Septiembre de 2021, de https://code.visualstudio.com/

Ministerio de educación. (30 de Noviembre de 2020). *Protección de Datos Personales*. Recuperado el 27 de Julio de 2021, de www.mineducacion.gov.co

Ministerio de transporte . (5 de Enero de 2021). *Estadisticas.* Recuperado el 5 de Mayo de 2021, de https://www.mintransporte.gov.co/documentos/15/estadisticas

Mohd, F., & Jayant, R. (2013). Survey on various noises and techniques for denoising the color image. *International Journal of Application or Innovation in Engineering & Management (IJAIEM).*

NumPy Developers. (s.f.). *numpy.random.normal*. Recuperado el 16 de Febrero de 2022, de https://numpy.org/doc/stable/reference/random/generated/numpy.random.normal.html

Open source initiative . (s.f.). *The MIT License*. Recuperado el 5 de Julio de 2021, de https://opensource.org/licenses/MIT

Oracle. (s.f.). *Java*. Recuperado el Septiembre de 2021, de dev.java

Organización mundial de la salud (OMS). (2017). *10 datos sobre la seguridad vial en el mundo*. Recuperado el 7 de Mayo de 2021, de https://www.who.int/

Papoulis, A. (2002). *Probability, Random Variables, and Stochastic Processes.* McGraw-Hill.

Parmar, J., & Patil, S. (2013). Performance evaluation and comparison of modified denoising method and the local adaptive wavelet image denoising method. *International Conference on Intelligent Systems and Signal Processing (ISSP).* Vallabh Vidyanagar, India.

Pereira, A. (2017). La ley de software libre en Colombia: tecnologías, imaginarios y reivindicaciones políticas. Bogotá D.C: Universidad del Rosario.

Poynton, C. (2012). *Digital Video and HD: Algorithms and Interfaces.* Elsevier.

Python Software Foundation. (s.f.). *Python*. Recuperado el Septiembre de 2021, de www.python.org/

Reddy, S., Mathew, M., Gomez, L., & Rusinol, K. D. (2020). RoadText-1K: Text Detection & Recognition Dataset for Driving Videos.

Reddy, S., Mathew, M., Gomez, L., Rusinol, M., & Karatzas, D. (2020). RoadText-1K: Text Detection & Recognition Dataset for Driving Videos. *2020 IEEE International Conference on Robotics and Automation (ICRA).* Paris.

Riveros, J. (2013). Reconocimiento y tratamiento de imágenes en sistemas embebidos para la detección de enfermedades en plantas. Bogotá D.C.: Universidad de los andes.

Rohrer, C. (2020). Adversarial detection using denoising autoencoder. Politecnico Milano.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, *65*(6), 386.

SAE International. (30 de Abril de 2021). *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*. Recuperado el 3 de Julio de 2021, de www.sae.org

Szeliski, R. (2010). *Computer Vision: Algorithms and Applications.* Springer Science & Business Media.

Tai, Y., Yang, J., Liu, X., & Xu, C. (2017). MemNet: A Persistent Memory Network for Image Restoration. *Proceedings of the IEEE international conference on computer vision*, (págs. 4539-4547).

Talero, W. (2015). Seguridad en los autos con sistemas de apoyo a la conducción. Universidad Piloto de Colombia.

Tian, C., Fei, L., Zheng, W., Xu, Y., & Lin, C.-W. (2020). Deep learning on image denoising: An overview. *Neural Networks, 131*, 251-275.

Torres, L., Ardila, A., Contreras, G., Castro, S., Medina, B., & Guevara, D. (2022). DCAEAD: Denoising system for assisted driving images applying an autoencoder architecture. *IEEE International Conference on Automation/XXV Congress of the Chilean Association of Automatic Control (ICA-ACCA).* Curicó, Chile.

Visual Data. (s.f.). Recuperado el 8 de Mayo de 2021, de https://visualdata.io/

Wang, S., Wang, H., Xiang, S., & Yu, L. (2020). Densely connected convolutional network block based autoencoder for panorama map compression. *Signal Processing: Image Communication, 80*, 115678.

Wu, P., Liu, J., Li, M., Sun, Y., & Shen, F. (2020). Fast sparse coding networks for anomaly detection in videos. *Pattern Recognition, 107*, 107515.

Xu, D., Yan, Y., Ricci, R., & Sebe, N. (2017). Detecting anomalous events in videos by learning deep representations of appearance and motion. *Computer Vision and Image Understanding, 156*, 117-127.

Yapici, A., & Akcayol, A. (2021). A Review of Image Denoising With Deep Learning. *2nd International Informatics and Software Engineering Conference.* Ankara, Turkey.

Zhang, K., Zuo, W., Chen, Y., Meng, D., & Zhang, L. (2017). Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising. *IEEE Transactions on Image Processing, 26*(7), 3142-3155.

Anexos

Anexo 1. Síntesis de la Revisión del Estado del Arte.

Artículo de investigación	Aportes	Recomendaciones
Deep learning for object detection and scene perception in self-driving cars: Survey, challenges, and open issues (Gupta, Anpalagan, Guan, & Shaharyar, 2021).	Presenta un análisis bastante completo de las diferentes técnicas usadas en el campo de la conducción asistida, abarcando los retos presentes en esta tarea y brindando datos útiles para investigaciones que pretendan abordar este campo.	Plantea que uno de los inconvenientes en la conducción asistida es que los datos obtenidos a través de video muchas veces no son suficientes y se deben complementar con otro tipo de sensores para poder hacer una toma de decisiones acertada.
Densely connected convolutional network block based autoencoder for panorama map compression (Wang, Wang, Xiang, & Yu, 2020).	En este artículo de investigación se presenta un método que pretende comprimir imágenes de entorno por medio de un autoencoders basado en un bloque de red convolucional densamente conectado, lo cual después de varias	Dado que la investigación se centró únicamente en el procesamiento de imágenes, no se cuenta con antecedentes en el campo del procesamiento de videos, lo cual representa un grado mayor de dificultad pues haría falta adaptar el sistema

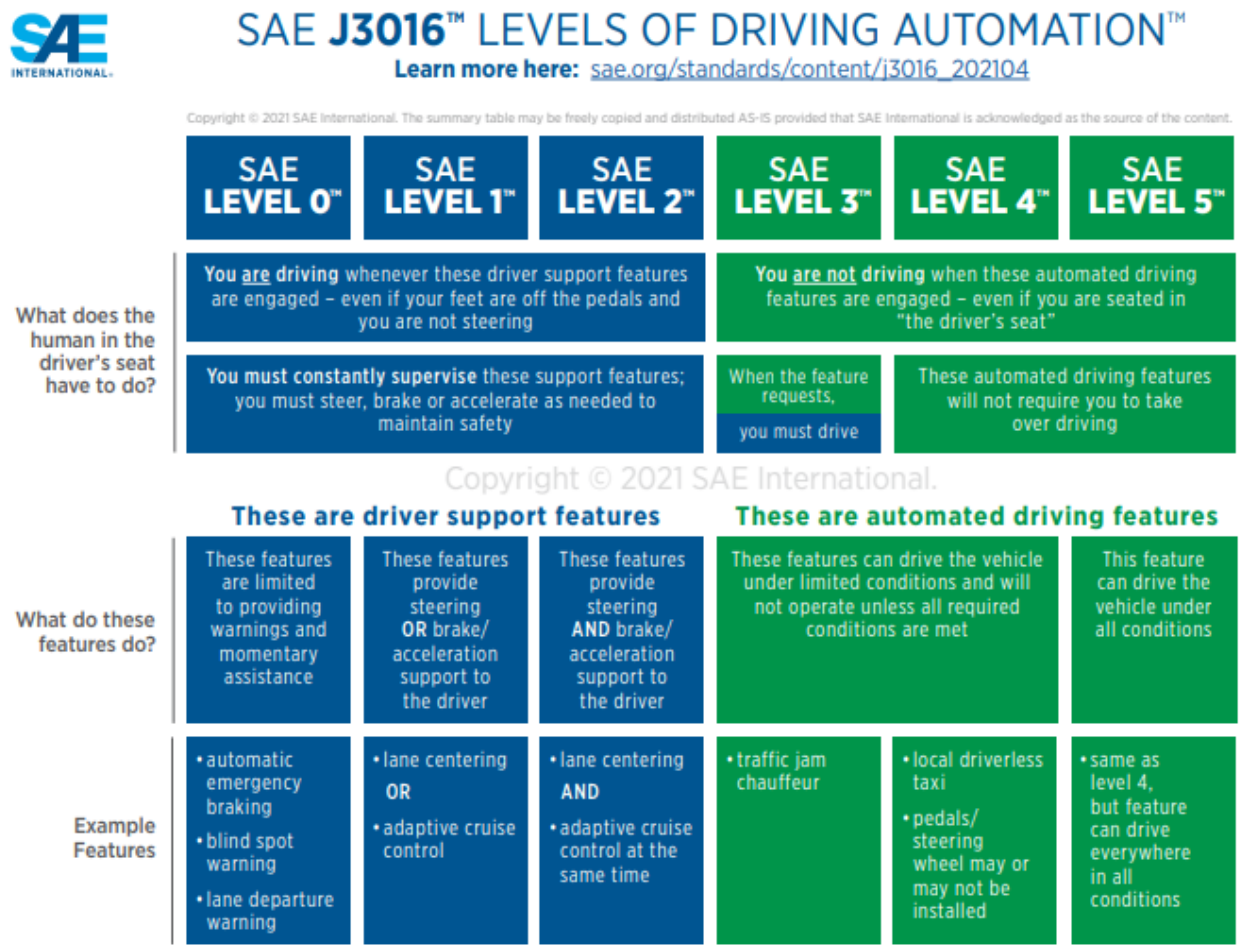
	pruebas demostró mayor eficiencia con respecto a otros medios en términos de PSNR y SSIM. Demostrando que el autoencoder propuesto puede reconstruir mapas con alta calidad visual.	para procesar videos y comprobar que la tasa de eficiencia se mantiene.
Detecting anomalous events in videos by learning deep representations of appearance and motion (Xu, Yan, Ricci, & Sebe, 2017).	En este artículo se investiga principalmente sobre la identificación de anomalías en videos por medio de implementación de autoencoders de eliminación de ruido apilados, esto con la finalidad de aprovechar la información complementaria de los patrones de apariencia y movimiento.	A pesar de que el sistema luego de ser evaluado obtuvo buenos resultados, no se considera del todo efectivo, pues requiere de un alto costo computacional para el procesamiento de video en tiempo real, lo cual se pretende solucionar en futuras investigaciones.
Fast sparse coding networks for anomaly detection in videos (Wu, Liu, Li, Sun, & Shen, 2020).	Dado que el procesamiento de videos en muchas ocasiones es más complejo que el procesamiento de imágenes, es que surgen	Los autores sugieren que, para mejorar el sistema presentado en este artículo de investigación, en futuras investigaciones se podría

	métodos como el expuesto en este artículo, que propone una red de codificación dispersa rápida (FSCN), a través de la cual pretenden detectar anomalías de video. Finalmente, luego de evaluar el sistema, concluyen que este es cientos o incluso miles de veces más rápido en la etapa de prueba.	entrenar el modelo en una etapa, es decir, extracción de características de entrenamiento modelo y FSCN simultáneamente.
Denoising Videos with Convolutional Autoencoders (Larrue, Li, Meng, & Han, 2018).	En este artículo se presenta un sistema de reducción de ruido en videos por medio de autoencoders convolucionales, aplicado en específico al ruido de tipo gaussiano.	Se evidencia que uno de los inconvenientes presentes fue la falta de tiempo para el entrenamiento dada la cantidad de parámetros que la red debe aprender.
Blind Denoising Autoencoder (Majumdar, 2019).	En esta investigación se analiza el desempeño de un autoencoder para la eliminación de ruido a ciegas, es decir, el autoencoder	Dado que el sistema planteado demostró ser efectivo en la reducción de ruido en imágenes, queda por estudiarse si dicha efectividad

	aprende de la propia muestra a medida que realiza el proceso de reducción de ruido. De esta forma se pretende contrarrestar el inconveniente presente en otras investigaciones donde el tiempo de entrenamiento es muy grande (Larrue, Li, Meng, & Han, 2018).	se mantiene al procesar datos en formato de video.
Denoising method of low illumination underwater motion image based on improved canny (Lv & Li, 2021)	Esta investigación propone un sistema de denoising basado en “improved canny”, a través del cual pretender mejorar la eficiencia al momento de procesar imágenes subacuáticas con baja iluminación. Luego de realizar varias pruebas se obtuvieron resultados satisfactorios, mostrando un valor máximo de SSIM de 0.92.	Luego de comparar los resultados obtenidos en términos de SSIM y PSNR con otras investigaciones, comprobaron que el método aplicado en esta investigación es más eficiente, sin embargo, estos resultados un se pueden mejorar explorando a mayor profundidad el algoritmo canny.

FDnCNN-based image denoising for multi-labfel localization measurement (Li, y otros, 2020)	A través de esta investigación los autores desarrollan un sistema de localización 3D basado en la eliminación de ruido en imágenes para lo cual proponen la aplicación de una red neuronal convolucional flexible Feed-forward (FDnCNN), buscando un equilibrio entre el proceso de eliminación de ruido y el consumo realizado por la GPU.	Como recomendación para futuras investigaciones los autores proponen aplicar al sistema una red neurona convolucional profunda (DCNN), con la finalidad de mejorar los resultados obtenidos.

Anexo 2. Niveles de Automatización de la Conducción (SAE J3016) *(SAE International, 2021)*.



Anexo 3. Comparación entre adición de ruido a imágenes normalizadas y no normalizadas.

Con la finalidad de demostrar la equivalencia entre adicionar ruido a imágenes normalizadas y no normalizadas, se tomó una imagen de prueba, a la cual se le añadió ruido Gaussiano en cada una de las escalas planteadas, notando que estas producen aproximadamente el mismo valor de PSNR, teniendo una diferencia de $0{,}0058\ dB$, como se puede apreciar a continuación.

Imagen normalizada

Nivel de ruido insertado, $\sigma = 50/255$

$PSNR = 15{,}27937\ dB$

Imagen no normalizada

Nivel de ruido insertado, $\sigma = 50$

$PSNR = 15{,}28517\ dB$

Anexo 4. Valores promedio de PSNR y SSIM obtenidos a partir del modelo entrenado con AWGN.

	Imágenes ruidosas		Imágenes estimadas	
Factor de ruido (σ)	PSNR (dB)	SSIM	PSNR (dB)	SSIM
20	22,39	0,6468	24,42	0,8067
30	19,07	0,4994	24,74	0,8189
40	16,80	0,3981	24,87	0,8272
50	15,09	0,3254	24,45	0,8203
60	13,76	0,2720	22,98	0,7548
70	12,68	0,2306	20,78	0,6226
80	11,79	0,1988	18,89	0,5072

Anexo 5. Valores promedio de PSNR y SSIM obtenidos a partir del modelo entrenado con Salt & Pepper.

	Imágenes ruidosas		Imágenes estimadas	
Factor de ruido ($Prob$)	PSNR (dB)	SSIM	PSNR (dB)	SSIM
0.0178	22,40	0,7794	35,76	0,9898
0.0380	19,09	0,6132	34,62	0,9862
0.0642	16,79	0,4745	33,43	0,9805
0.0950	15,10	0,3745	32,12	0,9719
0.1291	13,75	0,3029	30,81	0,9595
0.1659	12,67	0,2516	29,45	0,9422
0.2035	11,79	0,2143	28,15	0,9195

Anexo 6. Certificado de participación en el evento IEEE ICA-ACCA 2022.

Anexo 7. Artículo publicado en IEEE Xplore producto de la participación en el evento IEEE ICA-ACCA 2022.

Conferences > 2022 IEEE International Confe...

DCAEAD: Denoising system for assisted driving images applying an autoencoder architecture

Publisher: IEEE | Cite This | PDF

Luis Torres ; Andrea Ardila ; Ghiordy Contreras ; Sergio Castro ; Byron Medina ; Dinael Guevara All Authors

Abstract

Document Sections

I. Introduction

II. Berkeley Deep Drive Dataset

III. Denoising Convolutional Autoencoder For

Abstract:

The purpose of this research work is to develop a system to correct the noise present in images, using computer vision and automatic learning tools, specifically, focused on processing images used by assisted driving systems, in which the presence of noise represents a loss of information, directly affecting the efficiency of such systems.To achieve the above, a methodological process of analysis, design, and evaluation was followed, thus obtaining a neural network model capable of responding to the proposed task, which was verified by performing tests and applying metrics such as PSNR and SSIM, observing that the system can increase up to 14 dB the PSNR value and approximately 0.76 the SSIM value concerning the initial one, this specifically for the case of the tests with Gaussian noise, while for the Salt & Pepper noise the PSNR value increased by 8.53 dB and approximately 0.73 the SSIM value concerning the initial one.

Anexo 8. Certificado de participación en la IX Semana Internacional de Ciencia, Tecnología e Innovación.

9na Semana Internacional de Ciencia, Tecnología e innovación

LA UNIVERSIDAD FRANCISCO DE PAULA SANTANDER CÚCUTA Y SECCIONAL OCAÑA

CERTIFICA QUE:

LUIS CARLOS TORRES VEGA
C.C. 1005073089

Participó como PONENTE en el evento **IX SEMANA INTERNACIONAL DE CIENCIA, TECNOLOGÍA E INNOVACIÓN**, con el trabajo de investigación titulado "METODOLOGÍA EMPLEADA PARA EL DESARROLLO DE SISTEMAS DE CORRECCIÓN DE RUIDO EN IMÁGENES BASADO EN DEEP LEARNING".

El evento se desarrolló del 29 de noviembre al 02 de diciembre del año 2022 con una intensidad de 32 horas.

Se expide esta certificación a los 02 días del mes de diciembre del 2022, San José de Cúcuta, Colombia.

SERGIO IVÁN QUINTERO AYALA
VICERRECTOR ASISTENTE DE INVESTIGACIÓN Y EXTENSIÓN
UNIVERSIDAD FRANCISCO DE PAULA SANTANDER

NANCY RODRÍGUEZ COLORADO
DIRECTORA DIVISIÓN DE INVESTIGACIÓN Y EXTENSIÓN
UNIVERSIDAD FRANCISCO DE PAULA SANTANDER OCAÑA

ACREDITADA EN ALTA CALIDAD 6 años

Universidad Francisco de Paula Santander

Universidad Francisco de Paula Santander Ocaña - Colombia