

UNIVERSIDAD NACIONAL DE TRUJILLO

Facultad de Ciencias Físicas y Matemáticas

Escuela Profesional de Informática



**MODELO DE RECONOCIMIENTO AUTOMÁTICO DE
SEÑALES DE TRÁNSITO VEHICULAR MEDIANTE
APRENDIZAJE PROFUNDO DE REDES NEURONALES
CONVOLUCIONALES**

AUTOR: Josué Gastón Távara Idrogo

ASESOR: Jorge Luis Gutierrez Gutierrez

Trujillo - Perú

2019

**MODELO DE RECONOCIMIENTO AUTOMÁTICO DE
SEÑALES DE TRÁNSITO VEHICULAR MEDIANTE
APRENDIZAJE PROFUNDO DE REDES NEURONALES
CONVOLUCIONALES**

JOSUÉ GASTÓN TÁVARA IDROGO

**MODELO DE RECONOCIMIENTO AUTOMÁTICO DE
SEÑALES DE TRÁNSITO VEHICULAR MEDIANTE
APRENDIZAJE PROFUNDO DE REDES NEURONALES
CONVOLUCIONALES**

Tesis presentada a la Escuela Profesional de Informática en la Facultad de Ciencias Físicas y Matemáticas de la Universidad Nacional de Trujillo, como requisito para la obtención del Título profesional de Ing. Informático

ASESOR: DR. JORGE LUIS GUTIERREZ GUTIERREZ

Trujillo - Perú

2019

HOJA DE APROBACIÓN

Modelo De Reconocimiento Automático De Señales De Tránsito Vehicular Mediante Aprendizaje Profundo De Redes Neuronales Convolucionales

Josué Gastón Távara Idrogo

Tesis defendida y aprobada por el jurado examinador:

Prof. Dr. Jorge Luis Gutierrez Gutierrez - Asesor
Departamento de Informática - UNT

Prof. Mg. Anthony Gomez Gonzales
Departamento de Informática - UNT

Prof. Mg. Jose Luis Peralta Luján
Departamento de Informática - UNT

Trujillo, DIA de MES del 2019

Dedico esta tesis a:

Mi madre María y mi padre Gastón; amare et sapere vix deo concesitur.

Mis hermanas Carol y Marita por el apoyo continuo.

Agradecimientos

Agradezco a Dios por haberme bendecido en toda mi vida.

A mis profesores del Departamento de Informática, de los cuales recibí una gran cantidad de conocimientos y apoyo que formaron parte fundamental en el desarrollo de mi carrera universitaria.

A mi asesor Dr. Jorge Luis Gutierrez Gutierrez de la Universidad Nacional de Trujillo, que siempre se mostró disponible, interesado y capacitado para ayudarme, otorgándome las sugerencias necesarias para redactar esta investigación.

A la profesora Dra. Roseli Aparecida Francelin Romero de la Universidad de Sao Paulo por haber introducido el tema de Deep Learning en mi instancia como estudiante de esa universidad y haberme sugerido temas de investigación en esta área.

Resumen

La presente investigación tiene por objetivo principal implementar un modelo basado en el aprendizaje profundo de redes neuronales convolucionales para reconocer automáticamente señales de tránsito vehicular usando fundamentos de cálculo matemático, técnicas de procesamiento de imágenes y algoritmos de inteligencia artificial.

El proyecto se centra en un grupo de señales de Tránsito vehicular de Alemania y Perú, identificando 43 y 7 categorías respectivamente. Iniciando con la adquisición de imágenes, se procedió a realizar el procesamiento de estas con la finalidad de aumentar el conjunto de datos y poder ejecutar el aprendizaje profundo a través de diversos diseños de modelos de redes neuronales convolucionales.

Como resultado final, se obtuvo un modelo con buenos indicadores y resultados en el reconocimiento de señales de tránsito vehicular. De esta manera, se pretende contribuir en los esfuerzos de la industria automotriz en el campo de sistemas avanzados de asistencia al conductor, así como también puede formar parte de diversos mecanismos que buscan dar soluciones a la inseguridad vial.

Palabras claves: aprendizaje profundo, redes neuronales convolucionales, procesamiento de imágenes.

Abstract

The main objective of this research is to implement a model based on the deep learning through convolutional neural networks to automatically recognize vehicular traffic signals using mathematical calculation funds, image processing techniques and artificial intelligence algorithms.

The project focuses on a group of traffic signals from Germany and Peru, identifying 43 and 7 categories respectively. Starting with the acquisition of images, the processing of these activities is carried out with the purpose of increasing the data set so then be able to carry out in-depth learning through various designs of convolutional neural network architectures.

As a final result, a model with good indicators and results in the recognition of vehicular traffic signals was obtained. In this way, it is intended to contribute to the efforts of the automotive industry in the field of Advanced driver-assistance systems(ADAS), as well as being part of various mechanisms that seek to provide solutions to road safety.

Keywords: deep learning, convolutional neural networks, image processing.

Índice de figuras

1.1.	Señalización de velocidad	3
1.2.	Análisis de responsabilidad de accidentes	4
1.3.	Influencia de velocidad en accidentes	4
1.4.	Diagrama de bloques para el reconocimiento	6
2.1.	Algunas muestras de la base de datos GTSRB	15
2.2.	Ilustración de un modelo de aprendizaje profundo	23
2.3.	Diagrama de Venn donde cada sección incluye un ejemplo de una tecnología de IA (ilustra la relación entre estas diferentes disciplinas de la IA)	26
2.4.	Entrada y salida de una CNN	27
2.5.	Análisis de una CNN	28
2.6.	Terminología de capas complejas(izquierda) y de capas simples(derecha)	29
2.7.	Convolución entre el filtro y parte de la imagen	30
2.8.	Posicionamiento del kernel/filtro por pixel	31

2.9. Proceso matemático convolucional	32
2.10. Cálculo Convolucional	32
2.11. Conectividad dispersa vs No Dispersa	34
2.12. Capacidad receptiva en capas más profundas en una red convolucional	35
2.13. Resultado de Convolución (conjunto de mapas de activación. generado a partir de 3 filtros para 3 características: diagonal derecha, cruzamiento central y diagonal izquierda). Símbolo de convolución: \otimes	37
2.14. Imagen con stride igual a 2, para el filtro tanto en largura como anchura	38
2.15. Ejemplo de zero-padding con tamaño 2	38
2.16. Ejemplo de filtro creado a partir de los 3 aspectos mencionados	40
2.17. Funciones de activación	41
2.18. Procedimiento de la función ReLU	42
2.19. En la izquierda la red neuronal común y a la derecha la red neuronal diluida producida por la aplicación de dropout	43
2.20. Procesos que pueden ocurrir durante entrenamiento. Dropout evita el sobreajuste	44
2.21. Formas de agrupamiento(pooling)	45
2.22. Operación MAX-pooling en capa de Agrupación	46
2.23. Resultado de Agrupación	46
2.24. Max-pooling con filtro 2x2 y paso 2	47

2.25. Modelo de red neuronal convolucional profunda	48
2.26. Descenso de Gradiente a través de ajustes en los pesos y biases	52
2.27. Establecimiento de la Tasa de Aprendizaje	54
2.28. Comportamientos en la reducción de la Tasa de Aprendizaje	55
2.29. Ejemplo de Tasa de decaimiento de aprendizaje por época	55
2.30. Diseño del Modelo del Ciclo de Vida del Desarrollo	70
3.1. Diagrama de flujo para el desarrollo de la Investigación	72
3.2. Speed limit (20km/h)	73
3.3. Speed limit (50km/h)	73
3.4. Bumpy road	74
3.5. Children crossing	74
3.6. Turn left ahead	74
3.7. Distribución de ejemplos por señal para el entrenamiento (Total 39209)	74
3.8. Pare	75
3.9. Resalto	75
3.10. Zona Escolar	75
3.11. Peatones	75
3.12. Paradero	75
3.13. No Estacionar	76

3.14. Disminuir Velocidad	76
3.15. Distribución de ejemplos por señal para el entrenamiento (Total 614)	76
3.16. Distribución de ejemplos por señal para la evaluación - Alemania	77
3.17. Distribución de ejemplos por señal para la evaluación - Perú	78
3.18. El aumento de datos infla artificialmente los conjuntos de datos usando transformaciones que preservan las categorías de los objetos	79
3.19. Imágenes volteadas horizontalmente	80
3.20. Imágenes volteadas verticalmente	80
3.21. Imágenes volteadas primero horizontal y luego verticalmente	81
3.22. Imágenes que volteadas horizontal o verticalmente, cambian su categoría	81
3.23. Distribución de categoría, luego de aplicar el Flip (en sus distintos tipos)- Señales de Alemania	82
3.24. Ejemplo de cinco proyecciones por imagen - Dataset Perú y Alemania	83
3.25. Ejemplo de cinco rotaciones por imagen - Dataset Alemania y Perú	84
3.26. Ejemplo de cinco aplicaciones de zoom(in/out) por cada imagen - Dataset Alemania y Perú	85
3.27. Distribución eficaz de los valores de intensidad más frecuentes	86
3.28. Dataset balanceado (cada categoría posee igual cantidad de imágenes)	87
3.29. Dataset no balanceado - Señales de Tránsito de Perú	88
3.30. Imágenes a las cuales se le aplicaron la técnica CLAHE	90

3.31. Imágenes procesadas en escala de grises	91
3.32. Arquitectura de la CNN	93
3.33. Modelo del diseño de la Red propuesta	95
3.34. Modelo A Tasa de Acierto por iteración - Dataset de imágenes de Alemania	102
3.35. Modelo A Tasa de pérdida por iteración - Dataset de imágenes de Alemania	102
3.36. Modelo B Tasa de Acierto por iteración - Dataset de imágenes de Alemania	103
3.37. Modelo B Tasa de pérdida por iteración - Dataset de imágenes de Alemania	103
3.38. Modelo C Tasa de Acierto por iteración - Dataset de imágenes de Alemania	104
3.39. Modelo C Tasa de pérdida por iteración - Dataset de imágenes de Alemania	104
3.40. Modelo D Tasa de Acierto por iteración - Dataset de imágenes de Alemania	105
3.41. Modelo D Tasa de pérdida por iteración - Dataset de imágenes de Alemania	105
3.42. Modelo E Tasa de Acierto por iteración - Dataset de imágenes de Alemania	106
3.43. Modelo E Tasa de pérdida por iteración - Dataset de imágenes de Alemania	106
3.44. Análisis del Acierto durante el Entrenamiento de los modelos - Dataset Señales de Tránsito de Alemania	107
3.45. Análisis del Error durante el Entrenamiento de los modelos - Dataset Señales de Tránsito de Alemania	107
3.46. Análisis del Acierto en la Validación de los modelos - Dataset Señales de Tránsito de Alemania	108

3.47. Análisis del Error en la Validación de los modelos - Dataset Señales de Tránsito de Alemania	108
3.48. Modelo A Tasa de Acierto por iteración - Dataset de imágenes de Perú	109
3.49. Modelo A Tasa de pérdida por iteración	109
3.50. Modelo B Tasa de Acierto por iteración - Dataset de imágenes de Perú	110
3.51. Modelo B Tasa de pérdida por iteración	110
3.52. Modelo C Tasa de Acierto por iteración - Dataset de imágenes de Perú	111
3.53. Modelo C Tasa de pérdida por iteración	111
3.54. Modelo D Tasa de Acierto por iteración - Dataset de imágenes de Perú	112
3.55. Modelo D Tasa de pérdida por iteración	112
3.56. Modelo E Tasa de Acierto por iteración - Dataset de imágenes de Perú	113
3.57. Modelo E Tasa de pérdida por iteración	113
3.58. Análisis del Acierto durante el Entrenamiento de los modelos - Dataset Señales de Tránsito de Perú	114
3.59. Análisis del Error durante el Entrenamiento de los modelos - Dataset Señales de Tránsito de Perú	114
3.60. Análisis del Acierto en la Validación de los modelos - Dataset Señales de Tránsito de Perú .	115
3.61. Análisis del Error en la Validación de los modelos - Dataset Señales de Tránsito de Perú .	115
4.1. Matriz de Confusión del Modelo E - Dataset de imágenes de Alemania	120
4.2. Área debajo de la Curva ROC del Modelo E - Dataset de imágenes de Alemania	121

4.3.	Área debajo de la Curva PR del Modelo E - Dataset de imágenes de Alemania	121
4.4.	Matriz de Confusión del Modelo E - Dataset de imágenes de Perú	123
4.5.	Área debajo de la Curva ROC del Modelo E - Dataset de imágenes de Perú	124
4.6.	Área debajo de la Curva PR del Modelo E - Dataset de imágenes de Perú	124
4.7.	Flujograma para el reconocimiento de una señal de tránsito	125
4.8.	Interfaz reconociendo Señal de Tránsito “Wild Animals Crossing” - Alemania	126
4.9.	Interfaz reconociendo Señal de Tránsito “No Estacionar” - Perú	127

Índice de tablas

2.1. Indicadores para la investigación	63
3.1. Distribución Entrenamiento y Validación Dataset Balanceado - Señales de Tránsito de Alemania	88
3.2. Distribución Entrenamiento y Validación Dataset no Balanceado - Señales de Tránsito de Perú	89
3.3. Hiperparámetros del Modelo	92
3.4. Mini-batch e iteraciones en el Dataset de Señales de Tránsito de Alemania Balanceado . . .	93
3.5. Mini-batch e iteraciones en el Dataset de Señales de Tránsito de Perú No Balanceado . . .	93
3.6. Diseño A (73995 neuronas) - Dataset de Imágenes de Alemania	96
3.7. Diseño A (250679 neuronas) - Dataset de Imágenes de Perú	96
3.8. Diseño B (83339 neuronas) - Dataset de Imágenes de Alemania	97
3.9. Diseño B (274339 neuronas) - Dataset de Imágenes de Perú	97
3.10. Diseño C (82315 neuronas) - Dataset de Imágenes de Alemania	98

3.11. Diseño C (273315 neuronas) - Dataset de Imágenes de Perú	98
3.12. Diseño D (86411 neuronas) - Dataset de Imágenes de Alemania	99
3.13. Diseño D (277411 neuronas) - Dataset de Imágenes de Perú	99
3.14. Diseño E (90185 neuronas) - Dataset de Imágenes de Alemania	100
3.15. Diseño E (279623 neuronas) - Dataset de Imágenes de Perú	100
4.1. Indicadores de los 5 modelos entrenados en el Dataset - Alemania	119
4.2. Indicadores de los 5 modelos entrenados en el Dataset - Perú	122

Índice general

Dedicatoria	IV
Agradecimientos	V
Resumen	VI
Abstract	VII
Índice de Figuras	XIV
Índice de Tablas	XVI
1. Introducción	1
1.1. Justificación de la investigación	5
1.1.1. Justificación Académica	5
1.1.2. Justificación Social	6
1.2. Formulación del problema	8

1.3.	Hipótesis	8
1.4.	Objetivos	8
1.4.1.	Objetivo general	8
1.4.2.	Objetivos específicos	8
1.5.	Estructura de la tesis	9
2.	Materiales y Métodos	11
2.1.	Marco Teórico	11
2.1.1.	Antecedentes de la Investigación	11
2.1.1.1.	INTERNACIONALES:	12
2.1.1.2.	NACIONALES:	16
2.2.	Aprendizaje Profundo	17
2.3.	Red Convolutacional	26
2.3.1.	Capa Convolutacional	29
2.3.1.1.	Etapa de Convolución	29
2.3.1.2.	Capa ReLU (Rectified Linear Units)	40
2.3.1.3.	Técnica Dropout	43
2.3.1.4.	Capa de Agrupación(Pooling)	44
2.3.2.	Capa totalmente conectada (Fully-connected layer)	47
2.4.	Entrenamiento y Validación	50

2.4.1.	Descenso de Gradiente	50
2.4.1.1.	Gradiente Descendiente por Lotes (Batch Gradient Descent) . . .	52
2.4.1.2.	Gradiente Descendiente Estocástico (Stochastic Gradient Descent-SGD)	52
2.4.1.3.	Mini-batch Gradient Descent	53
2.4.2.	Tasa de Aprendizaje (Learning Rate)	53
2.4.3.	Optimizador ADAM	56
2.4.4.	Validación Cruzada	57
2.4.5.	Función Softmax	60
2.4.6.	Método de Regularización L2	61
2.5.	Método de la investigación	62
2.5.1.	Tipo de investigación	62
2.5.2.	Variables de la Investigación	62
2.5.2.1.	Variable Dependiente	62
2.5.2.2.	Variable Independiente	62
2.5.3.	Indicadores	63
2.5.4.	Recolección de Datos para la Construcción del Modelo	64
2.5.4.1.	Técnica de Recolección	64
2.5.4.2.	Población	64

2.5.4.3. Muestra	64
2.5.5. Recolección de Datos para el Entrenamiento y Evaluación del Modelo . . .	66
2.5.5.1. Técnica de Recolección	66
2.5.5.2. Población	66
2.5.5.3. Muestra	67
2.5.6. Etapas de la investigación	68
2.5.6.1. Metodología para el diseño del Modelo	69
3. Desarrollo de la Investigación	71
3.1. Análisis del conjunto de Imágenes	73
3.1.1. Datos Iniciales para el entrenamiento	73
3.1.1.1. Señales de Tránsito de Alemania	73
3.1.1.2. Señales de Tránsito de Perú	75
3.1.2. Datos para la evaluación	77
3.1.2.1. Señales de Tránsito de Alemania	77
3.1.2.2. Señales de Tránsito de Perú	78
3.1.3. Proceso de Aumento de Datos (Data Augmentation)	78
3.1.3.1. Flipping	80
3.1.3.2. Projection (Proyección)	82
3.1.3.3. Rotation (Rotación)	84

3.1.3.4. Zoom	85
3.1.3.5. Ecualización del histograma	86
3.1.4. Dataset final para el Entrenamiento	87
3.1.4.1. Señales de Tránsito de Alemania - Dataset Balanceado	87
3.1.4.2. Señales de Tránsito de Perú - Dataset No Balanceado	88
3.1.5. Pre-procesamiento de Imágenes (Normalization)	89
3.2. Arquitectura del Modelo	92
3.2.1. Diseño de la Red	95
3.2.1.1. Diseño A	96
3.2.1.2. Diseño B	97
3.2.1.3. Diseño C	98
3.2.1.4. Diseño D	99
3.2.1.5. Diseño E	100
3.3. Entrenamiento y Validación	101
3.3.1. Señales de Tránsito de Alemania	101
3.3.1.1. Análisis del Entrenamiento y Validación del Diseño A	102
3.3.1.2. Análisis del Entrenamiento y Validación del Diseño B	103
3.3.1.3. Análisis del Entrenamiento y Validación del Diseño C	104
3.3.1.4. Análisis del Entrenamiento y Validación del Diseño D	105

3.3.1.5. Análisis del Entrenamiento y Validación del Diseño E	106
3.3.2. Señales de Tránsito de Perú	109
3.3.2.1. Análisis del Entrenamiento y Validación del Diseño A	109
3.3.2.2. Análisis del Entrenamiento y Validación del Diseño B	110
3.3.2.3. Análisis del Entrenamiento y Validación del Diseño C	111
3.3.2.4. Análisis del Entrenamiento y Validación del Diseño D	112
3.3.2.5. Análisis del Entrenamiento y Validación del Diseño E	113
4. Resultados de la tesis	116
4.1. Señales de Tránsito de Alemania	119
4.1.1. Tabla de Resultados	119
4.1.2. Matriz de Confusión	120
4.1.3. Curvas ROC - AUC	121
4.1.4. Curvas PR - AUC	121
4.2. Señales de Tránsito de Perú	122
4.2.1. Tabla de Resultados	122
4.2.2. Matriz de Confusión	123
4.2.3. Curvas ROC - AUC	124
4.2.4. Curvas PR - AUC	124
4.3. Reconocimiento de Señal	125

4.3.1. Interfaz de Aplicación	126
5. Consideraciones finales	128
5.1. Conclusiones	128
5.2. Trabajos futuros y recomendaciones	130
Bibliografia	131
A. Pseudocódigo del Diseño de Red	136

Capítulo 1

Introducción

Al conducir en carreteras congestionadas, a veces es difícil mantener los ojos en todas partes a la vez, comprobando el camino por delante, el tráfico venidero, lo que está detrás de usted, tratar de mantener su velocidad; es por ello, que existen mecanismos destinados a reglamentar el tránsito, advertir o informar a los usuarios mediante palabras, sonidos o símbolos determinados. Un claro ejemplo de ello, es la policía de tránsito o las señalizaciones vehiculares que según sea el caso, en todos los países regulan el tránsito e informan al usuario sobre direcciones, rutas, destinos, así como dificultades existentes en las carreteras y previenen cualquier peligro que podría presentarse en la circulación vehicular.

Sin embargo, cuando estos mecanismos no son conocidos o percibidos pueden ocasionar no solo que la congestión del tráfico aumente, sino que también se produzcan accidentes que en muchos casos derivan en consecuencias fatales, generando inseguridad vial.

La inseguridad vial es un problema de interés mundial, según el último informe de la OMS (Organización Mundial de la salud) anualmente cerca de 1,3 millones de personas mueren alrededor del mundo y entre 20 y 50 millones padecen traumatismos no mortales (OMS, 2017). Esto representa la segunda de las principales causas de muerte a nivel mundial entre los jóvenes de 5 a 29 años de edad, y la tercera entre la población de 30 a 44 años. Son distintas las causas que conllevan a este problema, de las cuales las principales pueden ser la falta de concientización y educación vial.

Es un problema para la economía mundial de los países, así también para los hogares. A pesar de ello, se invierte muy poco dinero en prevenir los accidentes y las lesiones causadas por el tránsito. El sector salud se beneficiaría mucho de una mejor prevención de las lesiones porque se reducirían las hospitalizaciones y la gravedad de los traumatismos (Consejo Nacional de Seguridad Vial, 2014).

Los usuarios vulnerables de la vía pública representan la mitad de todas las muertes por accidente de tránsito a nivel mundial y es mayor en países de ingresos bajos, siendo la causa principal, el aumento de velocidad durante la conducción de los vehículos (OMS, 2017).



Figura 1.1: Señalización de velocidad

Fuente: OMS (2017)

Por otra parte, de los vehículos que se venden en el 80 % de los países no cumplen las normas básicas de seguridad, es por ello que se debe trabajar en obtener vehículos más seguros ya que es un factor fundamental para prevenir de alguna forma los accidentes de tránsito o reducir la probabilidad de traumatismos graves en caso de que estos se produzcan (OMS, 2017).

En lo que respecta al sector nacional, es decir en el Perú, la inseguridad vial es un problema constante ya que los peruanos mueren más por los accidentes de tránsito que por la inseguridad ciudadana según datos mostrados por María Edith Baca de la Organización Panamericana de la Salud (Cordova Rampant, 2017). Adicionalmente, según un estudio realizado por RPPData en base a reportes de la Policía publicados entre el 2010 y el 2016, en el país cada día fallecen 8 personas en accidentes de tránsito, (Romero Benites, 2017). El costo de estas muertes, calculado en S/ 19,165 millones por la consultora Alauda especializada en mantenimiento de infraestructura, representó un 3.1 % del PBI (Gestión.pe, 2016a).

En el 2016 se obtuvo un índice Global de Satisfacción del Conductor (Waze, 2016) en el cual, Perú y la capital Lima se encuentran respectivamente en la lista de peores países y ciudades para conducir en América Latina, esto se ve reflejado en que los últimos años se ha incrementado el índice de mortandad originados por los accidentes de tránsito siendo las principales causas de los mismos el exceso de velocidad, estado de ebriedad del conductor, imprudencia temeraria y el desacato a las señales de tránsito, todas ellas de responsabilidad directa del conductor del vehículo motorizado al no respetar las señales de tránsito(SUTRAN, 2014).



Figura 1.2: Análisis de responsabilidad de accidentes
Fuente acceso: Gestión.pe (2016b)



Figura 1.3: Influencia de velocidad en accidentes
Fuente acceso: Gestión.pe (2016b)

El reconocimiento de estas señales es un problema de clasificación multi-categórica que comúnmente presenta desigualdades en las frecuencias de aparición de sus clases. Además, las señales de tránsito muestran una amplia gama de variaciones entre las clases en términos de color, forma y la presencia de símbolos, leyendas o texto. Además, existen subconjuntos de clases (por ejemplo, signos de límite de velocidad) que son muy similares entre sí, lo que representa un reto para un reconocedor automático, el cual tiene que hacer frente a grandes variaciones en las apariencias visuales debido a cambios de iluminación, occlusiones parciales, rotaciones, condiciones meteorológicas, escalamiento, etc.

1.1. Justificación de la investigación

1.1.1. Justificación Académica

Recientemente las redes convolucionales profundas han superado los métodos tradicionales de aprendizaje en la clasificación de imágenes. Con los rápidos avances de las estructuras de algoritmos de aprendizaje profundo y la factibilidad de su implementación de alto rendimiento con unidades de procesamiento gráfico (GPU), es ventajoso investigar en problemas de clasificación de imágenes desde la perspectiva de un aprendizaje profundo eficiente, (Gu et al., 2015). Sin embargo, realizar esto no es tarea simple, ya que se requiere de un modelo de reconocimiento que funcione para imágenes que se encuentran comúnmente influenciadas por la iluminación, la orientación, la variación de velocidad de los vehículos, entre muchos otros problemas más.

En este sentido, esta investigación pretende la elaboración de un modelo basado en el aprendizaje profundo de redes convolucionales que permita el reconocimiento de señales de tránsito vehicular. La siguiente figura muestra un diagrama de bloques con la secuencia de actividades que demanda el reconocimiento.



Figura 1.4: Diagrama de bloques para el reconocimiento

Fuente: Elaboración propia

Por lo tanto, la importancia de esta investigación en el punto de vista de ciencias de la computación se justifica en poner en práctica los conocimientos adquiridos en la formación académica, siendo los más resaltantes el tema de procesamiento de imágenes e inteligencia artificial con la finalidad de obtener un modelo robusto de redes neuronales convolucionales basadas en el aprendizaje profundo (deep learning) que permita analizar el contenido de imágenes para el reconocimiento de señales de tránsito vehicular.

1.1.2. Justificación Social

Teniendo conocimiento de lo descrito al inicio de este capítulo, la visibilidad y conocimiento de señales de tráfico es crucial para la seguridad de los conductores y es por ello que la introducción de un modelo de reconocimiento de señales de tránsito que funcione en diferentes contextos puede formar parte de la solución a que se puedan evitar constantes infracciones y muertes, y en con-

secuencia reducir estos índices progresivamente. Por ejemplo, el reconocimiento de estas señales en el momento de la conducción puede ofrecer la posibilidad de dar una notificación al no darse cuenta de un cambio en el límite de velocidad, el aviso de que se está cometiendo una infracción al girar o estacionarse donde no se debe o la advertencia de un peligro potencial por delante. Por otro lado, cuando usuarios desconocen de alguna señal de tránsito, una aplicación móvil que dé la posibilidad de reconocer automáticamente aquella señal serviría como aporte en la educación vial.

Por lo tanto, esta investigación es importante porque a través de un modelo que reconozca señales de tránsito vehicular se podría contribuir en la industria automotriz, específicamente en los sistemas avanzados de asistencia al conductor (del inglés, *ADAS*); así como también se ha descrito anteriormente, el modelo pretendido puede ser usado en diferentes plataformas y formar parte de diversos mecanismos que buscan dar soluciones a la inseguridad vial.

Todos los hechos descritos hacen que el reconocimiento de las señales de tránsito sea un reto desafiante y esencial en muchos aspectos, no solo para contribuir en los esfuerzos de la industria automotriz en el campo de la asistencia al conductor, sino también incluso para organismos gubernamentales quienes se dan cuenta de la problemática que representa la inseguridad vial y buscan constantemente introducir nuevos mecanismos y tecnologías que faciliten y mejoren la conducción vehicular para el beneficio propio del conductor y en general para la seguridad vial en la sociedad.

1.2. Formulación del problema

En este trabajo, se propone discutir el modelo de redes neuronales convolucionales basado en el problema del reconocimiento de imágenes para responder a la siguiente pregunta:

¿Cómo se puede reconocer de manera automática señales de tránsito vehicular?

1.3. Hipótesis

Un modelo basado en el aprendizaje profundo de redes neuronales convolucionales permitirá el reconocimiento automático de señales de tránsito vehicular.

1.4. Objetivos

1.4.1. Objetivo general

La investigación tiene por objetivo principal implementar un modelo basado en el aprendizaje profundo de redes neuronales convolucionales para reconocer automáticamente señales de tránsito vehicular.

1.4.2. Objetivos específicos

- a) Obtener dos conjuntos de imágenes(datasets) donde se muestren diferentes señales de tránsito vehicular.
- b) Dividir ambos conjuntos de imágenes en 3 grupos, uno para el entrenamiento, uno para

validación del entrenamiento y un tercero para evaluación del modelo.

- c) Analizar los datasets a través de métodos de procesamiento de imágenes, con el objetivo de aumentar la cantidad de imágenes.
- d) Implementar diferentes modelos de redes convolucionales profundas en las cuales los datasets serán procesados.
- e) Experimentar el uso de diversas funciones de activación, funciones de costo, ajuste de hiperparámetros y métodos de optimización para dichos modelos.
- f) Evaluar individualmente el rendimiento que se obtiene de los modelos implementados en el dataset de entrenamiento y evaluación.
- g) Elaborar un modelo computacional basado en las evaluaciones realizadas.

1.5. Estructura de la tesis

El presente trabajo está dividido en cinco capítulos. El primer capítulo presenta los aspectos generales de la investigación realizada tal como justificación, formulación del problema, hipótesis, los objetivos y la estructura de la tesis.

El capítulo describe los materiales y las técnicas de recolección de datos, se presenta el referencial teórico, soporte del tema, contemplando los conceptos de inteligencia artificial, aprendizaje automático basado en redes neuronales detallando y diferenciando sus características en el área de

aprendizaje profundo. Además, se analiza las características y procesos de una red convolucional a detalle. Finalmente se describe el método empleado durante la investigación.

En el tercer capítulo es el desarrollo de la tesis, diseñándose los modelos propuestos y explicando la implementación de cada uno de ellos.

En el cuarto capítulo se presentan los resultados y discusión obtenidos en la investigación.

En el capítulo cinco se presentan las consideraciones finales obtenidas en esta tesis. Inicialmente se presentan las conclusiones, seguida de las recomendaciones para futuras investigaciones relacionadas al tema en cuestión.

Finalmente, se describen las referencias bibliográficas usadas para la investigación en esta tesis y los anexos donde se presentan pseudocódigos de los programas elaborados.

Capítulo 2

Materiales y Métodos

2.1. Marco Teórico

En este capítulo se presenta una reseña del material bibliográfico investigado con relación a los temas considerados en esta investigación. Los conocimientos investigados son muy amplios, principalmente aquel que ayudó a consolidar las bases del conocimiento científico para elaborar esta tesis, como lo son los temas de aprendizaje artificial profundo, redes neuronales convolucionales, metaheurísticas, procesamiento de imágenes, conocimientos sin los cuales sería difícil de modelar y solucionar computacionalmente cualquier tipo de problema de reconocimiento de imágenes.

2.1.1. Antecedentes de la Investigación

En este capítulo se presentan 8 trabajos previos con relación a al tema abordado en la investigación y que sirven de base para este.

2.1.1.1. INTERNACIONALES:

(Vicen-Bueno et al., 2007) en la investigación denominada "Traffic Sign Classification by Image Preprocessing and Neural Networks", propusieron clasificar 9 tipos de señales de tráfico de circulación (color azul) de España utilizando una combinación de diferentes técnicas de pre-procesamiento de imágenes junto con una red perceptron de 2 capas la cual produjo un acierto de 98.72 % sobre un conjunto de 78 imágenes para la evaluación. El orden de aplicación de técnicas de pre-procesamiento de imágenes es lo más destacable de este antecedente, recomendando usar el filtro mediano para suavizar la imagen, ecualización de histogramas e histogramas vertical y horizontal con un umbral fijo de 185.

(Rocha and Escorcia, 2010) en su investigación "Sistema de Visión Artificial para la Detección y el Reconocimiento de Señales de Tráfico basado en Redes Neuronales", diseñaron un sistema de detección basado en redes neuronales feedforward y descriptores de forma conocidos como momentos invariantes. Este sistema fue capaz de entrenarse con algunas señales de tránsito con la meta de asistir al conductor de no cometer una infracción o en el peor de los casos un accidente, reconociendo una señal de tránsito a cierta distancia para que así el conductor a priori tenga el conocimiento de esta. El sistema fue implementado en MATLAB y presentó mejorías frente a sistemas basados en lógica difusa o basados únicamente en procesamiento de imágenes, sin embargo, la tasa de acierto no es tan buena obteniéndose un 88.6 % y se recomendó buscar otro

método de invarianza que conjuntamente a una red neuronal pueda conseguir mejores resultados.

Así el antecedente contribuye a descartar algunos métodos que no puedan presentar mejoras en el reconocimiento de señales de tránsito.

(Krizhevsky et al., 2012) en su investigación "The ImageNet Classification with Deep Convolutional Neural Networks", desarrollaron una red neuronal convolucional grande y profunda para clasificar 1,2 millones de imágenes de alta resolución del concurso ImageNet LSVRC-2010 que categorizaban 1000 clases diferentes de imágenes. Como dato destacable utilizaron un método de regularización muy efectivo para evitar el overfitting de la red denominado dropout, que permitió alcanzar mejores tasas de error en la clasificación que las anteriores técnicas en el estado del arte.

(Hannan et al., 2014) en la investigación realizada en Malasia "Traffic Sign Classification based on Neural Network for Advance Driver Assistance System", elaboraron un sistema con pasos de pre-procesamiento y extracción de características para clasificar señales de tránsito usando una red perceptron multicapa que funcione en diferentes condiciones de luminosidad. Lo destacable de la investigación, aparte de que el sistema fue capaz de superar la mayor parte del efecto de iluminación en las imágenes, es el tiempo computacional requerido para el análisis de una imagen, siendo en promedio 0.134s, sin embargo la tasa de precisión no fue buena obteniéndose un 84.4 % de acierto para un conjunto de evaluación compuesta por 300 imágenes.

(Hai Nguyen, 2014) en su artículo "Morphological Classification for Traffic Sign Recognition", propone un nuevo método para el Reconocimiento de Señales de Tránsito usando el Análisis de Componentes Principales (PCA) y una red Perceptron de Multi-Capa (MLP). En este método propuesto, las señales se detectan individualmente a partir de dos componentes, uno es el color y luego se clasifican en tres clases según la forma: círculo, cuadrado y triángulo. Las características basadas en PCA de estas señales se utilizarán como entrada para la MLP durante la fase de entrenamiento o para responder a clases previamente determinadas. Como contribución resaltante, este enfoque no sólo redujo el tiempo, sino que también aumentó el rendimiento en el proceso de reconocimiento. En la simulación, el método propuesto fue evaluado con más de 500 imágenes y su tasa de precisión llegó a cerca del 96 %.

Como antecedente más relevante para clasificación de señales de tránsito es el realizado en base a datos de Alemania, GTSRB (German Traffic Sign Benchmark) cuenta con más de 50 mil imágenes a color distribuidas en 43 clases, dicha base de datos ha sido tomada como base en la realización de estudios de diferentes métodos tales como: Clasificación basada en la Representación Dispersa (Sparse Representation-based Classification), el algoritmo de Vecinos Más Cercanos (Nearest Neighbor Classifier), Máquina de Soporte de Vectores (Support Vector Machine), entre otros. El mejor resultado fue usando redes neuronales convolucionales (Cireşan et al., 2012). Este conjunto de datos refleja las fuertes variaciones en la apariencia visual de las señales debido a la

distancia, la iluminación, las condiciones climáticas, las occlusiones parciales y las rotaciones.



Figura 2.1: Algunas muestras de la base de datos GTSRB

Fuente: Stallkamp et al. (2012)

Los resultados sobre la base de datos GTSRB produjeron un acierto del 99.46 % para el conjunto de imágenes evaluadas, (Stallkamp et al., 2012); sin embargo, desde entonces se han propuesto nuevas variantes de redes neuronales convolucionales, ya sea usando funciones de activación diferentes, usando más capas en la red, mas filtros o introduciendo nuevos métodos de optimización. Lo importante del antecedente es que servirá como elemento de comparación para la investigación propuesta.

2.1.1.2. NACIONALES:

(Vargas Romero, 2015) en su tesis "Implementación de un Sistema Inteligente para el Reconocimiento de Señales Preventivas de Seguridad vial", desarrolló un sistema basado enteramente en procesamiento de imágenes para la detección y el reconocimiento de señales de tránsito del tipo preventivas, usando para la detección la técnica de segmentación por color y luego un análisis por la forma y posteriormente para el reconocimiento se usó el algoritmo Speed Up Robust Features (SURF). Todo el proceso fue implementado utilizando la herramienta OpenCV, obteniéndose un acierto del 88 % en un total de 100 señales evaluadas. De este antecedente se analizará la importancia de introducir la segmentación por color al modelo de reconocimiento que se pretende elaborar.

(Ayuque Arenas, 2016) en su tesis "Diseño de un sistema de clasificación de señales de tránsito vehicular utilizando redes neuronales convolucionales", diseñó 6 modelos de arquitecturas de redes convolucionales para clasificar diversas señales de tránsito de Alemania usando el dataset GTSRB, de los cuales el mejor resultado logró un acierto de 95.29 % en un total de 12630 señales evaluadas, cercano al 99.46 % (Cireşan et al., 2012). Lo importante del antecedente es que servirá como elemento de comparación para la investigación propuesta.

2.2. Aprendizaje Profundo

En los primeros días de la inteligencia artificial, el campo atacó y resolvió rápidamente problemas que son intelectualmente difíciles para los seres humanos, pero relativamente sencillos para las computadoras, problemas que pueden describirse mediante una lista de reglas formales y matemáticas. El verdadero desafío para la inteligencia artificial fue y es resolver las tareas que son fáciles de realizar para las personas pero difíciles de describir de manera formal, problemas que resolvemos intuitivamente, que se sienten automáticos, como reconocer palabras, rostros u objetos en las imágenes (Goodfellow et al., 2016).

Si bien aprendizaje automático (del inglés, **machine learning**) se describe a menudo como una subdisciplina de la inteligencia artificial(IA), es mejor considerarla como la mejor técnica de la disciplina; es decir, es el campo de la IA que hoy en día muestra la mayor promesa al proporcionar herramientas que la industria y la sociedad pueden usar para producir algún cambio. En tal sentido, el aprendizaje automático toma algunas de las ideas centrales de la inteligencia artificial y las enfoca en resolver problemas del mundo real con redes neuronales diseñadas para imitar nuestra propia toma de decisiones; sin embargo, el **aprendizaje profundo** o avanzado de máquinas (del inglés, **deep learning**) se centra aún más estrechamente en un subconjunto de herramientas y técnicas del aprendizaje automático y los aplica a la solución de casi cualquier problema que requiera "pensamiento", ya sea humano o artificial (Goodfellow et al., 2016).

El aprendizaje profundo es un enfoque específico utilizado para construir y entrenar redes neuronales para la toma de decisiones altamente prometedoras. Se considera que un algoritmo es profundo si los datos de entrada se pasan a través de una serie de no linealidades o transformaciones no lineales antes de que se emita. Este enfoque permite que las computadoras aprendan de la experiencia y entiendan el mundo en términos de una jerarquía de conceptos, con cada concepto definido a través de su relación con conceptos más simples. Al reunir el conocimiento de la experiencia, este enfoque evita la necesidad de que los operadores humanos especifiquen formalmente todo el conocimiento que necesita la computadora. La jerarquía de conceptos permite que la computadora aprenda conceptos complicados mediante la construcción de los más simples. Si dibujamos un gráfico que muestra cómo estos conceptos se construyen uno encima del otro, el gráfico es profundo, con muchas capas. Por esta razón, este enfoque se conoce como el aprendizaje profundo de la inteligencia artificial (Goodfellow et al., 2016).

El aprendizaje profundo elimina el hecho que se tenga que realizar una identificación manual de las características en los datos y, en cambio, depende del proceso de capacitación que tenga para descubrir los patrones útiles en los ejemplos de entrada. Esto hace que el entrenamiento de la red neuronal sea más fácil y más rápido (Goodfellow et al., 2016). Las imágenes son un gran ejemplo de cómo funciona esto, ya que contienen muchos elementos diferentes y no es fácil para nosotros comprender cómo una computadora, con su mente o proceso centrado en el cálculo y dirigido en un solo sentido, puede aprender a interpretarlas de la misma manera que nosotros.

Pero el aprendizaje profundo se puede aplicar a cualquier forma de datos (señales de máquina, audio, video, voz, palabras escritas) para producir conclusiones que parecen haber sido alcanzadas por simples humanos.

Por el contrario, la mayoría de los algoritmos modernos de machine learning se consideran "poco profundos", debido a que la entrada solo puede abarcar unos pocos niveles de llamadas en subrutinas. El rendimiento de estos algoritmos simples de aprendizaje automático depende en gran medida de la presentación de los datos que se les proporcionan. Por ejemplo, cuando se usa la regresión logística para recomendar una cesárea, el sistema de IA no examina al paciente directamente. En cambio, el médico le dice al sistema varias piezas de información relevante, como la presencia o ausencia de una cicatriz uterina. Cada pieza de información incluida en la representación del paciente se conoce como una característica. La regresión logística aprende cómo cada una de estas características del paciente se correlaciona con diversos resultados. Sin embargo, no puede influir en cómo se definen las características de todos modos. Si a la regresión logística se le realizara una resonancia magnética del paciente, en lugar del informe formal del médico, no sería capaz de hacer predicciones útiles. Los píxeles individuales en una exploración de la resonancia tienen una correlación insignificante con cualquier complicación que pueda ocurrir durante el parto. Esta dependencia de las representaciones es un fenómeno general que aparece no solo en la vida cotidiana sino también en las ciencias de la computación. En tal sentido, operaciones como buscar una colección de datos puede avanzar exponencialmente más rápido si la colección

está estructurada e indexada de forma inteligente. La gente puede realizar fácilmente números aritméticos arábigos, pero encuentra que la aritmética en números romanos consume mucho más tiempo. No es sorprendente que la elección de la representación tenga un enorme efecto en el rendimiento de los algoritmos de aprendizaje automático (Goodfellow et al., 2016).

Para muchas tareas, sin embargo, es difícil saber qué características se deben extraer. Por ejemplo, un programa para detectar automóviles en fotografías. Se sabe que los autos tienen ruedas, por lo que sería importante utilizar la presencia de una rueda como característica. Desafortunadamente, es difícil describir exactamente cómo es una rueda en términos de valores de píxel. Una rueda tiene una geometría simple, pero su imagen puede verse complicada por las sombras que caen sobre la rueda, el sonido de las partes metálicas de la rueda, el guardabarros del automóvil o un objeto en el primer plano que oscurece la rueda, y así sucesivamente.

Una solución a este problema es utilizar el aprendizaje automático para descubrir no solo el mapeo que inicia en la representación hasta el resultado sino también la representación en sí misma. Este enfoque se conoce como aprendizaje representativo (del inglés, **representation learning**). Las representaciones aprendidas a menudo dan como resultado un rendimiento mucho mejor que el que se puede obtener con representaciones diseñadas a mano. También permiten que los sistemas de IA se adapten rápidamente a nuevas tareas, con una intervención humana mínima. Un algoritmo de aprendizaje de representación puede descubrir un conjunto de características para una tarea simple en minutos o para una tarea compleja en horas. El diseño manual de funciones para una tarea

compleja requiere una gran cantidad de tiempo humano y esfuerzo; puede llevar décadas para toda una comunidad de investigadores, (Goodfellow et al., 2016).

Al diseñar algoritmos para el aprendizaje de características, el objetivo suele ser separar los factores de variación que explican los datos observados. En este contexto, se usa la palabra "factores", simplemente para referirse a fuentes de influencia separadas; los factores generalmente no se combinan por multiplicación. Tales factores a menudo no son cantidades que se observan directamente. En cambio, pueden existir como objetos no observados o fuerzas no observadas en el mundo físico que afectan las cantidades observables. También pueden existir como constructores en la mente humana que proporcionan una simplificación útil de las explicaciones o causas inferidas de los datos observados. Pueden considerarse como conceptos o abstracciones que nos ayudan a dar sentido a la rica variabilidad de los datos. Al analizar una grabación de voz, los factores de variación incluyen la voz del hablante, su sexo, su acento y las palabras que están hablando. Al analizar la imagen de un automóvil, los factores de variación incluyen la posición del automóvil, su color y el ángulo y el brillo del sol.

Una fuente importante de dificultad en muchas aplicaciones de inteligencia artificial del mundo real es que muchos de los factores de variación influyen en cada pieza de datos que podemos observar. Los píxeles individuales en una imagen de un automóvil rojo pueden ser muy cercanos al negro en la noche. La forma de la silueta del automóvil depende del ángulo de visión. La mayoría de las aplicaciones nos exigen separar los factores de variación y descartar las que no nos interesan.

Por supuesto, puede ser muy difícil extraer tales características abstractas de alto nivel a partir de datos en bruto. Muchos de estos factores de variación, como el acento de un hablante, solo se identifican mediante una comprensión sofisticada y casi humana de los datos. Cuando es casi tan difícil obtener una representación como para resolver el problema original, el aprendizaje de la representación, a primera vista, no parece ayudarnos.

El aprendizaje profundo resuelve este problema central en el aprendizaje representativo mediante la introducción de representaciones que se expresan en términos de otras representaciones más simples. El aprendizaje profundo permite a la computadora construir conceptos complejos a partir de conceptos más simples. La Figura 2.2 muestra cómo un sistema de aprendizaje profundo puede representar el concepto de una imagen de una persona combinando conceptos más simples, como esquinas y contornos, que a su vez se definen en términos de bordes.



Figura 2.2: Ilustración de un modelo de aprendizaje profundo
 Fuente: Goodfellow et al. (2016)

La idea de aprender la representación correcta de los datos proporciona una perspectiva para el aprendizaje profundo, (Goodfellow et al., 2016). Otra perspectiva del aprendizaje profundo es que la profundidad permite que la computadora aprenda un programa de computadora de varios pasos. Cada capa de la representación se puede considerar como el estado de la memoria del computador después de ejecutar otro conjunto de instrucciones en paralelo. Las redes con mayor profundidad pueden ejecutar más instrucciones en secuencia. Las instrucciones secuenciales ofrecen gran poder porque las instrucciones posteriores pueden referirse a los resultados de instrucciones anteriores. De acuerdo con esta visión del aprendizaje profundo, no toda la información en las activaciones de

una capa necesariamente codifica los factores de variación que explican la entrada. La representación también almacena información del estado que ayuda a ejecutar un programa el cual puede dar sentido a la entrada. Esta información de estado podría ser análoga a un contador o puntero en un programa informático tradicional. No tiene nada que ver con el contenido de la entrada específicamente, pero ayuda al modelo a organizar su procesamiento.

Hay dos formas principales de medir la profundidad de un modelo, (Goodfellow et al., 2016). La primera vista se basa en la cantidad de instrucciones secuenciales que se deben ejecutar para evaluar el modelo. Podemos considerar esto como la longitud de la ruta más larga a través de un diagrama de flujo que describe cómo calcular cada una de las salidas del modelo a partir de sus entradas. Otro enfoque, utilizado por modelos probabilísticos profundos, considera que la profundidad de un modelo no es la profundidad del gráfico computacional sino la profundidad del gráfico que describe cómo los conceptos se relacionan entre sí. En este caso, la profundidad del diagrama de flujo de los cálculos necesarios para calcular la representación de cada concepto puede ser mucho más profunda que la gráfica de los conceptos mismos. Esto se debe a que la comprensión del sistema de los conceptos más simples puede ser refinado dada la información sobre conceptos más complejos.

Debido a que no siempre está claro cuál de estos dos puntos de vista (la profundidad del gráfico computacional o la profundidad del gráfico de modelado probabilístico) es más relevante, y debido

a que diferentes personas eligen diferentes conjuntos de elementos más pequeños para construir sus gráficos, no hay un solo valor correcto para la profundidad de un modelo, así como no hay un solo valor correcto para la longitud de un programa de computadora. Tampoco hay consenso sobre la profundidad que requiere un modelo para calificar como "profundo", (Goodfellow et al., 2016). Sin embargo, el aprendizaje profundo puede considerarse con seguridad como el estudio de modelos que implican una mayor cantidad de composición de funciones o conceptos aprendidos en comparación con el aprendizaje automático tradicional.

En resumen, el aprendizaje profundo, es un enfoque a la IA. Específicamente, es un tipo de aprendizaje automático, una técnica que permite que los sistemas computacionales mejoren con la experiencia y los datos. En este sentido, se puede sostener que el aprendizaje automático es el único enfoque viable para construir sistemas de inteligencia artificial que puedan operar en entornos complicados del mundo real y el aprendizaje profundo es un tipo particular de aprendizaje automático que logra gran poder y flexibilidad al representar el mundo como una jerarquía de conceptos anidados, con cada concepto definido en relación a conceptos más simples y representaciones más abstractas calculadas en base a representaciones menos abstractas. La siguiente imagen es un diagrama de Venn que muestra cómo el aprendizaje profundo es una especie de aprendizaje de representación, que a su vez es una especie de aprendizaje automático que se utiliza para muchos (pero no todos) los enfoques de la IA.

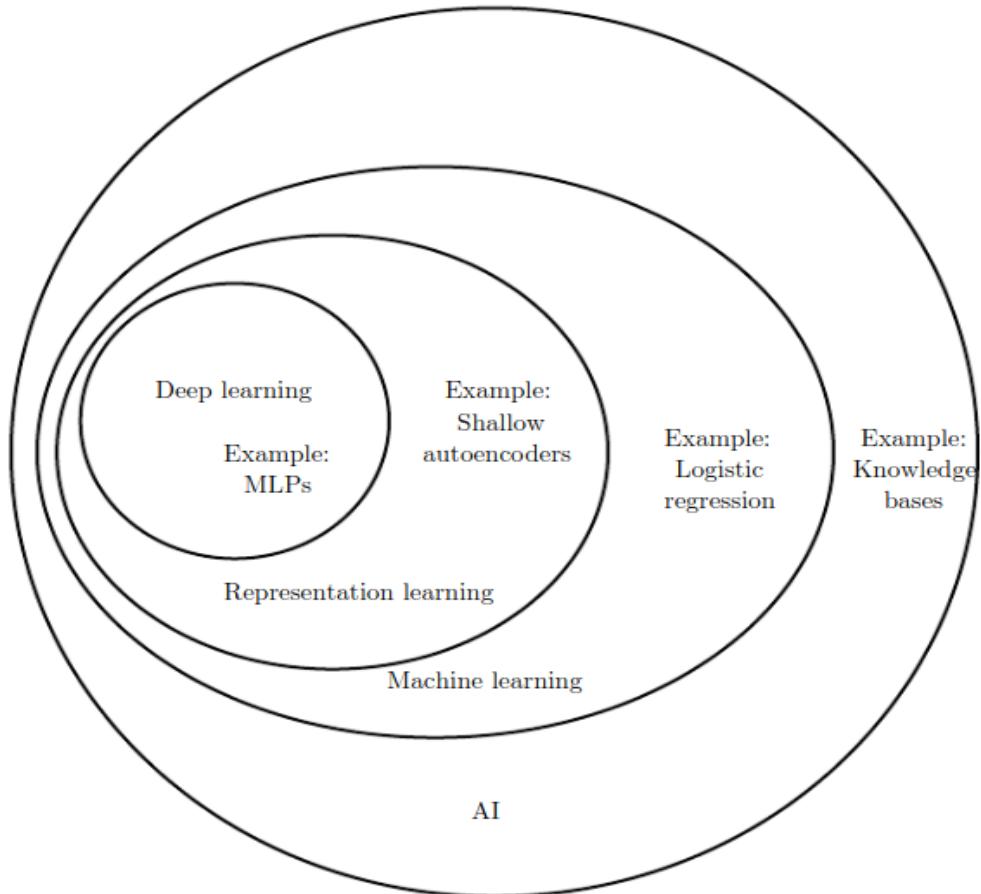


Figura 2.3: Diagrama de Venn donde cada sección incluye un ejemplo de una tecnología de IA (ilustra la relación entre estas diferentes disciplinas de la IA)

Fuente: Goodfellow et al. (2016)

2.3. Red Convolucional

Nueve de cada diez veces, cuando se escucha que el aprendizaje profundo rompe una nueva barrera tecnológica, las redes neuronales convolucionales están involucradas. También llamados CNN (del inglés, **Convolutional Neural Networks**) o **ConvNets**, estas son las preferidas del cam-

po de redes neuronales profundas. Han aprendido a clasificar las imágenes en categorías incluso mejor que los humanos en algunos casos, (Rohrer, 2016).

Los modelos de redes convolucionales siempre suponen explícitamente que las entradas deben ser mapeadas en forma de imágenes, lo que nos permite configurar parte inicial de las propiedades o características del modelo a diseñar. Debido a esta característica la limitación de las ConvNets radica en que solo capturan patrones espaciales locales en datos. Es decir, si no se puede hacer que los datos se vean como una imagen, las ConvNets son prácticamente muy poco útiles, (Rohrer, 2016).

Las redes neuronales convolucionales son muy similares a las redes neuronales ordinarias, están formadas por neuronas que tienen pesos y biases (sesgos) que se pueden aprender durante el entrenamiento de estas. Cada neurona recibe algunas entradas y realiza operaciones matemáticas. Toda la red expresa una única función de puntuación diferenciable: desde el contenido de los píxeles de la imagen en un extremo(entrada) hasta los puntajes que definen la clase o resultado correspondiente en el otro extremo(salida).

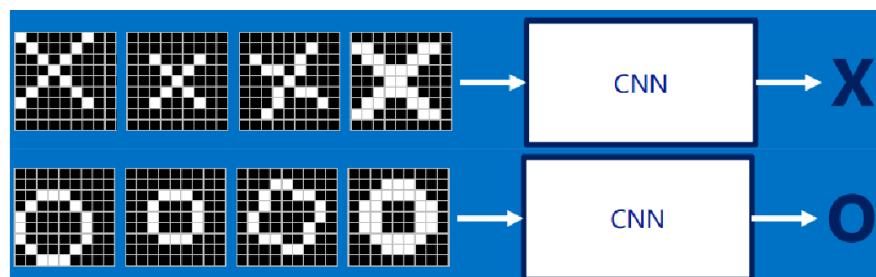


Figura 2.4: Entrada y salida de una CNN
Fuente: Rohrer (2016)

El resultado de las CNNs es que pueden encontrar si una característica está en una imagen sin preocuparse exactamente de donde está. Esto ayuda a resolver el problema de las computadoras al comparar imágenes de manera híper-literal, es decir, que coincida pixel a pixel para que se trate de imágenes iguales.

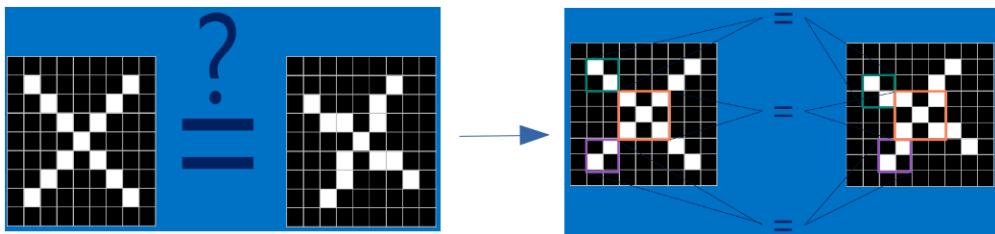


Figura 2.5: Análisis de una CNN
Fuente: Rohrer (2016)

Una ConvNet se caracteriza por tener una secuencia de capas donde cada una de estas transforma un volumen de activaciones en otro nuevo a través de funciones y el aprendizaje profundo se produce cuando se utilizan varias de estas capas variando los parámetros de configuración dentro y entre dichas capas.

Existen dos conjuntos de terminologías para describir estas capas. Una es cuando la red convolucional es vista como un número largo de capas simples y cada paso del procesamiento se considera como una capa en sí misma. Otra terminología es cuando la red convolucional es vista como un número pequeño de capas relativamente complejas, donde cada capa tiene múltiples etapas. En esta terminología, existe un mapeo directo entre los volúmenes de activaciones y las capas de red. En esta investigación se usará esta terminología.



Figura 2.6: Terminología de capas complejas(izquierda) y de capas simples(derecha)

Fuente: Goodfellow et al. (2016)

2.3.1. Capa Convolucional

Los parámetros de la capa convolucional consisten básicamente en dos datos. La entrada y todo lo que respecta a un conjunto de filtros (también denominados kernels) cuyos valores se aprenden, es decir, empiezan con datos aleatorios y conforme avance el entrenamiento se van alterando.

2.3.1.1. Etapa de Convolución

Para el proceso convolucional cada filtro es pequeño espacialmente (a lo ancho y alto), incluso se extiende a través de la profundidad total del volumen de entrada(imagen). Por ejemplo, un filtro

típico en una primera capa de una ConvNet podría tener un tamaño de 5x5x3 (es decir, 5 píxeles de ancho y alto, y 3 de profundidad debido a que los canales de color - RGB). Durante el proceso hacia adelante, se desliza (más precisamente, convoluciona) cada filtro a través del ancho y alto (incluso profundidad) del volumen de entrada para calcular los productos de puntos entre las entradas del filtro y la entrada en cualquier posición.

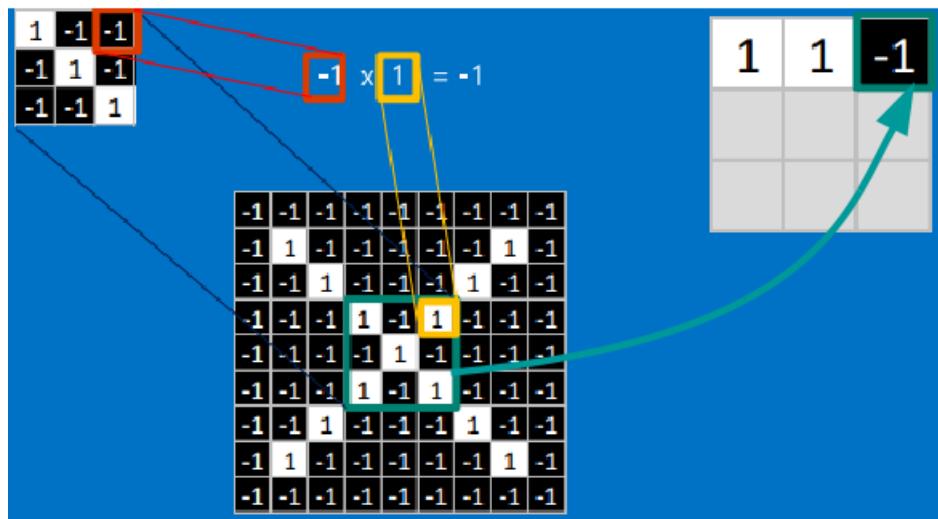


Figura 2.7: Convolución entre el filtro y parte de la imagen
Fuente: Rohrer (2016)



Figura 2.8: Posicionamiento del kernel/filtro por pixel

Fuente: Elaboración propia

A medida que deslizamos el filtro sobre el ancho y la altura del volumen de entrada produciremos un mapa de activación bidimensional que proporciona las respuestas de ese filtro en cada posición espacial. Es decir, el proceso en esta capa consiste en calcular la coincidencia de un filtro con una parte de la imagen, y para conseguirlo simplemente se multiplica cada píxel en el filtro por el valor del píxel en la imagen. Para luego, sumar las respuestas y dividirlas por el número total de píxeles en el filtro.

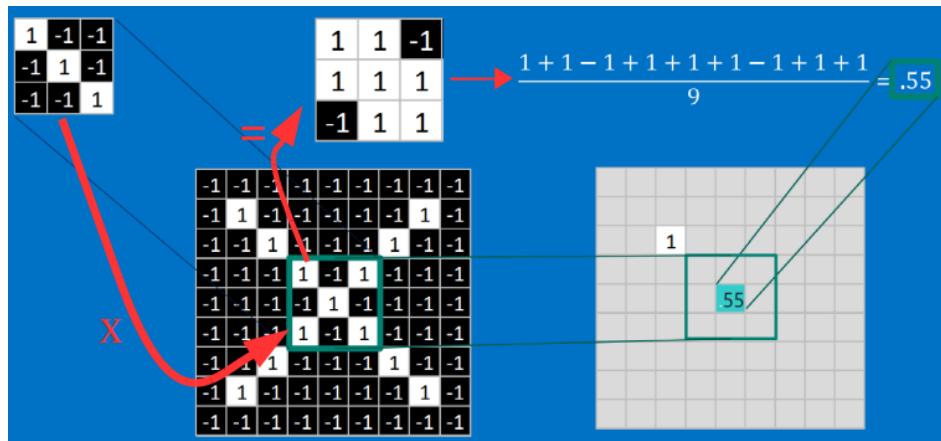


Figura 2.9: Proceso matemático convolucional

Fuente: Rohrer (2016)

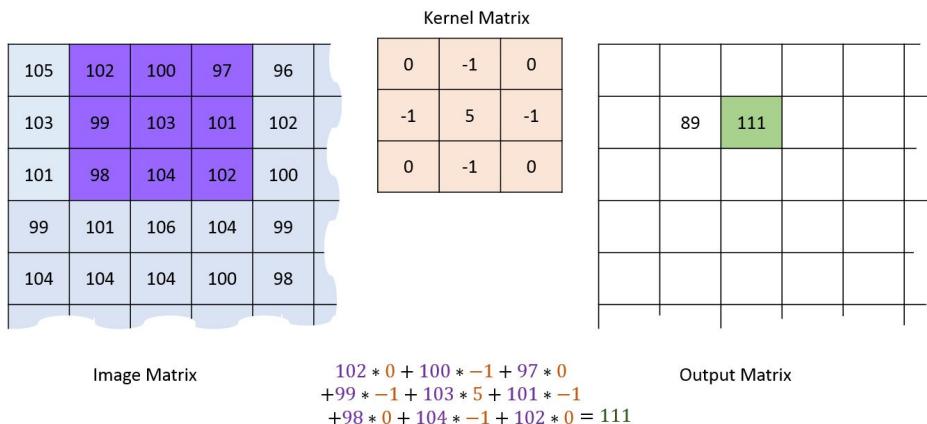


Figura 2.10: Cálculo Convolucional

Fuente: Elaboración propia

Cada filtro puede ser representado como una neurona de salida, cuyo valor se halla usando la sumatoria de pesos como se muestra en la figura 11. Intuitivamente, la red aprenderá los filtros que se activan cuando ven algún tipo de característica visual, como un borde o contorno en alguna orientación específica.

La convolución aprovecha tres ideas importantes que pueden ayudar a mejorar un sistema de aprendizaje automático: **interacciones dispersas, uso compartido de parámetros y representaciones equivalentes**.

Las capas de redes neuronales tradicionales usan la multiplicación de matrices mediante una matriz de parámetros con un parámetro separado que describe la interacción entre cada unidad de entrada y cada unidad de salida. Esto significa que cada unidad de salida interactúa con cada unidad de entrada. Sin embargo, las redes convolucionales suelen tener **interacciones dispersas** (también conocidas como conectividad dispersa o ponderaciones dispersas), en las cuales el kernel es más pequeño que la entrada. Por ejemplo, al procesar una imagen, la imagen de entrada puede tener miles o millones de píxeles, pero podemos detectar características pequeñas y significativas, como bordes con núcleos que ocupan solo decenas o cientos de píxeles, necesitando almacenar menos parámetros, lo que reduce los requisitos de memoria del modelo, mejora su eficiencia estadística y también conlleva a que el cálculo de la salida requiera menos operaciones.

Estas mejoras en la eficiencia suelen ser bastante grandes. Si hay entradas(n) y salidas(m), la multiplicación de la matriz requiere $n \times m$ parámetros, y los algoritmos utilizados en la práctica por ejemplo tienen complejidad de tiempo de ejecución $O(n \times m)$. Si limitamos el número de conexiones que cada salida puede tener a k , entonces el enfoque dispersamente conectado requiere solo $k \times m$ parámetros y $O(k \times m)$ en tiempo de ejecución. Para muchas aplicaciones prácticas, es

possible obtener un buen rendimiento en la tarea de aprendizaje automático mientras se mantienen k distintos órdenes de magnitud menores que m . Para demostraciones gráficas de conectividad dispersa, vea la figura 2.11, donde se resalta una unidad de entrada x_3 , y las unidades de salida que son afectadas por esta unidad. (Arriba) Cuando s está formado por convolución con un kernel de ancho 3, solo tres salidas se ven afectadas por x . (Abajo) Cuando s está formado por la multiplicación de la matriz, la conectividad ya no es dispersa, por lo que todos los resultados se ven afectados por x_3 .

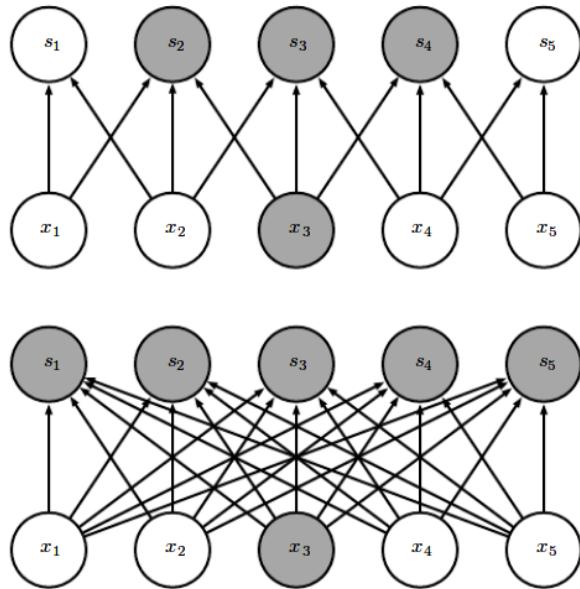


Figura 2.11: Conectividad dispersa vs No Dispersa
Fuente: Goodfellow et al. (2016)

En una red convolucional profunda, las unidades en las capas más profundas pueden interactuar indirectamente con una porción más grande de la entrada. Esto permite que la red describa de manera eficiente las interacciones complicadas entre muchas variables mediante la construcción de

tales interacciones a partir de bloques de construcción simples que describen cada una de ellas. Por lo tanto, aunque las conexiones directas en una red convolucional son muy dispersas, las unidades en las capas más profundas pueden conectarse indirectamente a la totalidad o a la mayoría de la imagen de entrada, (Goodfellow et al., 2016).



Figura 2.12: Capacidad receptiva en capas más profundas en una red convolucional
Fuente: Goodfellow et al. (2016)

El uso **compartido de parámetros** hace referencia al uso del mismo parámetro para más de una función en un modelo. En una red neuronal tradicional, cada elemento de la matriz de pesos se usa exactamente una vez cuando se calcula la salida de una capa. Se multiplica por un elemento de la entrada y luego nunca se vuelve a visitar. Como sinónimo de compartición de parámetros, se puede decir que una red tiene ponderaciones vinculadas, porque el valor de la ponderación aplicada a una entrada está vinculado al valor de una ponderación aplicada a otra parte. En una red neuronal convolucional, cada miembro del kernel se utiliza en cada posición de la entrada (excepto tal vez algunos de los píxeles de los bordes, dependiendo de las decisiones de diseño con respecto al límite). El uso compartido de parámetros utilizado por la operación de convolución significa

que, en lugar de aprender conjuntos de parámetros para cada ubicación por separado, aprendemos un solo conjunto, (Goodfellow et al., 2016).

En el caso de la convolución, la forma particular de compartir los parámetros hace que la capa tenga una propiedad llamada **representaciones equivalentes**. Decir que una función es equivalente significa que, si la entrada cambia, la salida cambia de la misma manera. La convolución crea un mapa en 2-D de donde aparecen ciertas características en la entrada. Si movemos el objeto en la entrada, su representación se moverá la misma cantidad en la salida. Esto es útil cuando sabemos que se utiliza alguna función de un número pequeño de píxeles vecinos cuando se aplica a ubicaciones de entrada múltiples. Por ejemplo, al procesar imágenes, es útil detectar bordes en la primera capa de una red convolucional. Los mismos bordes aparecen más o menos en todas partes en la imagen, por lo que es práctico compartir los parámetros en toda la imagen.

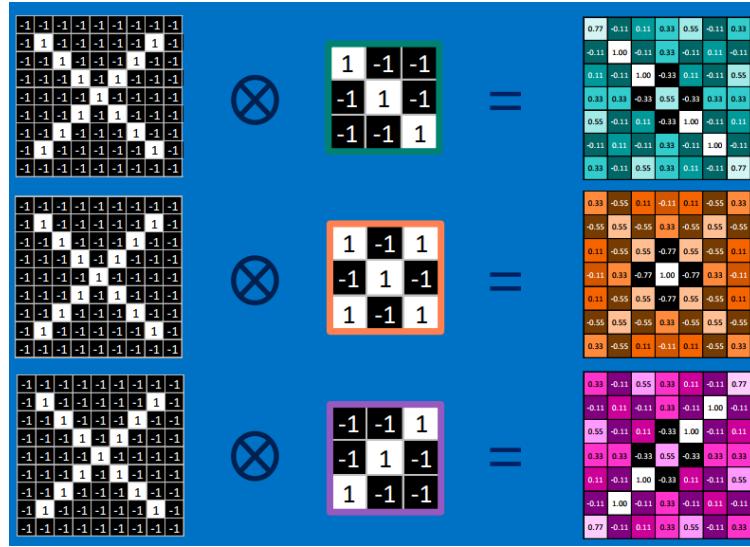


Figura 2.13: Resultado de Convolución (conjunto de mapas de activación. generado a partir de 3 filtros para 3 características: diagonal derecha, cruzamiento central y diagonal izquierda).

Símbolo de convolución: \otimes

Fuente: Rohrer (2016)

En resumen, se tiene un conjunto completo de 'n' filtros en cada capa convolucional (determinando la profundidad de la capa) y cada uno de estos filtros producirá un mapa de activación bidimensional por separado. Apilaremos estos mapas de activación a lo largo de la dimensión de profundidad y produciremos el volumen de salida, es decir el resultado es un conjunto de imágenes que muestran una versión filtrada de la imagen original resaltando características o patrones importantes de ella, como puede ser visto en la figura 2.13.

Cabe resaltar que para la construcción de un filtro o kernel, es necesario considerar tres aspectos importantes: la extensión espacial (spatial extent), el paso (stride) y la cantidad de zero a llenar (zero-padding).

1. La extensión espacial es el tamaño del filtro, comúnmente es de tamaño impar tanto en largo y ancho.
2. El stride es otra pieza del bloque de construcción básico de los filtros convolucionales. Este representa el 'paso' en la operación de convolución indicando cuánto es que se debe desplazar un filtro en una imagen con cada paso. El filtro se desliza sobre la imagen, se detiene en cada longitud de salto y realiza las operaciones necesarias en ese paso.

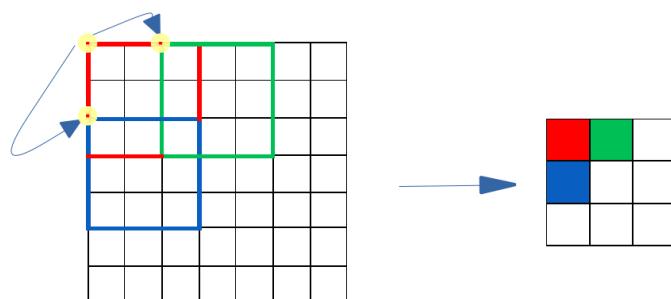


Figura 2.14: Imagen con stride igual a 2, para el filtro tanto en largura como anchura
Fuente: Elaboración propia

3. Zero-padding agrega ceros alrededor del borde de una imagen.

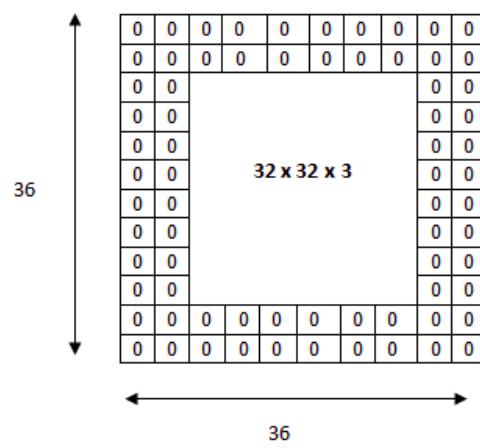


Figura 2.15: Ejemplo de zero-padding con tamaño 2
Fuente: Elaboración propia

Los principales beneficios del relleno son los siguientes:

- Le permite usar una capa de convolución sin necesariamente reducir la altura y el ancho de los volúmenes. Esto es importante para construir redes más profundas, ya que de lo contrario la altura / ancho se reduciría a medida que se avanza hacia capas más profundas.
- Nos ayuda a mantener más información en el borde de una imagen. Sin relleno, muy pocos valores en la siguiente capa se verían afectados por los píxeles como los bordes de una imagen.

Cada capa convolucional recibe como dato de entrada los parámetros: $W_0 \times H_0 \times D_0$

Produce un dato de salida con los siguientes parámetros: $W_1 \times H_1 \times D_1$

Estas salidas, son influenciadas por la manera de configuración de los filtros.

En el que:

$$W_1 = \frac{W_0 - F + 2P}{S} + 1$$

$$H_1 = \frac{H_0 - F + 2P}{S} + 1$$

$$D_1 = K$$

Donde:

W es el ancho (width) de la imagen,

H es la altura (height) de la imagen,

D es la profundidad (depth) de la imagen,

F es la extensión espacial (spatial extent) del filtro,

S es el paso (stride) del filtro,

P es la cantidad de zero padding del filtro,

K es el número de filtros (filters).

La ecuación de convolución de manera generalizada es:

$$conv_j^n = \sum_{k=1}^k x_k^n \times w_{kj}^n + b_n$$

En el que:

- x, w, b son valores de entrada, pesos y sesgos (biases), respectivamente.
- n es el número de la capa
- j es el número del filtro de salida
- k es la cantidad de filtros en la capa $n - 1$ o n

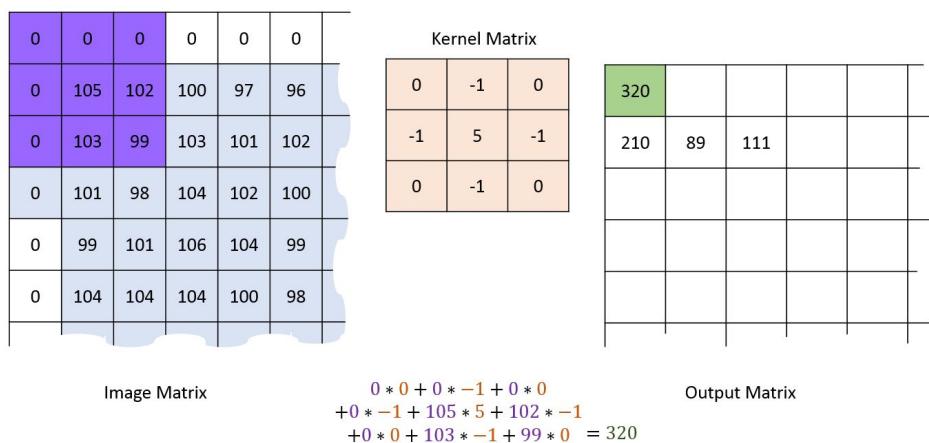


Figura 2.16: Ejemplo de filtro creado a partir de los 3 aspectos mencionados

Fuente: Elaboración propia

2.3.1.2. Capa ReLU (Rectified Linear Units)

Debido al hecho que todas las capas en una red neuronal no son lineales, después de calcular los valores para cada una de las neuronas en la red neuronal, colocamos estos valores a través de

una función de activación. Una red neuronal artificial consiste básicamente en multiplicaciones y suma de matrices. Si solo utilizáramos estos cálculos lineales, podríamos apilarlos uno encima del otro y esa no sería una red muy profunda. Por lo tanto, a menudo se utiliza funciones de activación no lineales en cada capa de la red. Apilando capas de funciones lineales y no lineales una encima de la otra, teóricamente podemos modelar cualquier problema.

Estas son las tres funciones de activación no lineal más populares:

- 1) Sigmoid (analiza un valor entre 0 y 1)
- 2) TanH (analiza un valor entre -1 y 1)
- 3) ReLU (si el valor es negativo, se convierte en 0, de lo contrario, permanece igual)

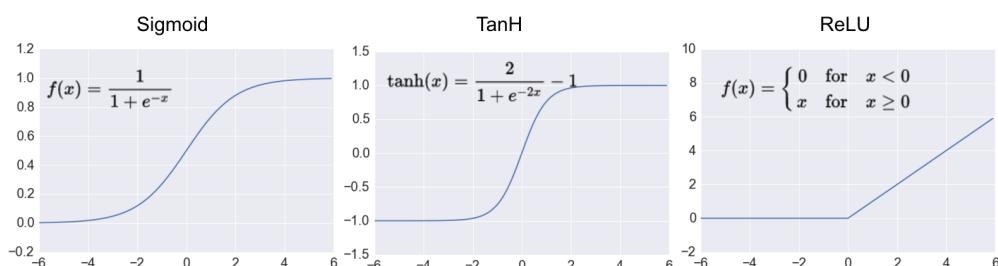


Figura 2.17: Funciones de activación
Fuente: Elaboración propia

Actualmente, ReLU es la función de activación no lineal más utilizada (Karpthy, 2016). La razón principal de esto es porque la red puede entrenar mucho más rápido (debido a la eficiencia computacional) sin hacer una diferencia significativa en la precisión. También ayuda a aliviar el problema del gradiente de fuga, que es el problema donde las capas inferiores de la red entran

muy lentamente porque el gradiente de optimización disminuye exponencialmente a través de las capas. La capa ReLU aplica la función $f(x) = \max(0, x)$ a todos los valores en el volumen de entrada. En términos básicos, esta capa simplemente cambia todas las activaciones negativas a 0. Esta capa aumenta las propiedades no lineales del modelo y la red global sin afectar los campos receptivos de la capa convolucional. El hecho de que entradas en la función de activación de valores menores o iguales a cero resulten cero induce a la dispersión en las unidades ocultas, que según lo comentado anteriormente, produce representaciones dispersas las cuales se consideran más valiosas, (Nair and Hinton, 2010).



Figura 2.18: Procedimiento de la función ReLU
Fuente: Rohrer (2016)

Es por ello que entre la capa de convolución y la capa de pooling puede encontrarse la capa ReLU, la cual está compuesta por neuronas que poseen una función de activación llamada Función Lineal Rectificada que deriva de la función de activación sigmoidal, pero tiene mayores ventajas

que esta última y también de la tangencial. Las activaciones de ReLU se sobreponen más fácilmente que los sigmoides, esto los prepara muy bien para ser utilizados en combinación con la técnica DROPOUT.

2.3.1.3. Técnica Dropout

Combinar las predicciones de muchos modelos diferentes es una forma muy exitosa de reducir los errores de prueba, pero parece ser demasiado costoso para las redes neuronales de gran tamaño debido a que pueden tardar varios días en entrenar. Sin embargo, dropout es una técnica de regularización que tiene por objetivo reducir el sobreajuste que puede darse durante el entrenamiento de una red neuronal. Esta técnica consiste en establecer a cero la salida de ciertas neuronas denominadas neuronas ocultas, las cuales son seleccionadas aleatoriamente en cada capa con una probabilidad comúnmente del 50 %. Las neuronas que se abandonan de esta manera no contribuyen al pase directo y no participan en las siguientes etapas de entrenamiento (Romero, 2015a).

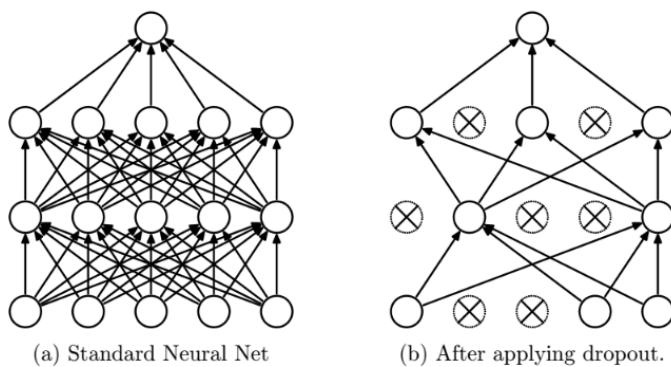


Figura 2.19: En la izquierda la red neuronal común y a la derecha la red neuronal diluida producida por la aplicación de dropout

Romero (2015a)

Por lo tanto, cada vez que se presenta una entrada, la red neuronal muestra una arquitectura diferente, pero todas estas arquitecturas comparten ponderaciones. Esta técnica reduce las co-adaptaciones complejas de las neuronas, ya que una neurona no puede depender de la presencia de otras neuronas particulares. Por lo tanto, se ve forzada a aprender características más robustas que son útiles en conjunción con muchos otros subconjuntos aleatorios provenientes de otras neuronas.

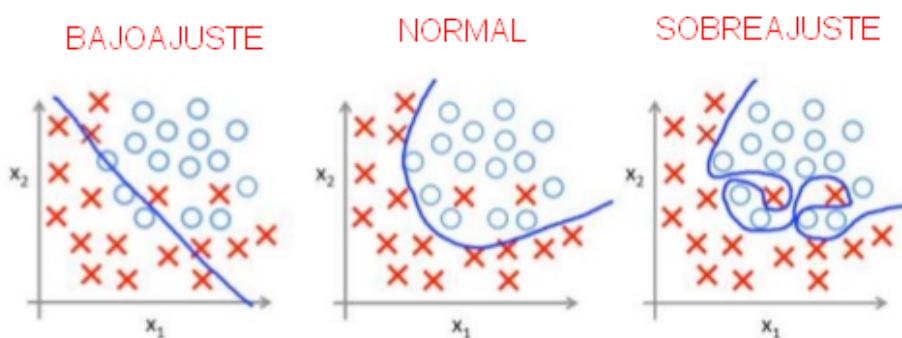


Figura 2.20: Procesos que pueden ocurrir durante entrenamiento. Dropout evita el sobreajuste
Fuente: Elaboración propia

2.3.1.4. Capa de Agrupación(Pooling)

Una variedad de cálculos que reducen la dimensión de un mapa de características se conocen como agrupación. La agrupación, que se aplica a cada canal por separado, permitiendo que la red sea robusta e invariantes a pequeños cambios y distorsiones. La capa Pooling (también conocida como una capa de reducción de resolución) combina o agrupa un conjunto de valores en su campo receptivo generando un menor número de valores. Puede ser configurado en función del tamaño de su campo receptivo (por ejemplo, 2 x 2) y en función a la operación de agrupamiento (por ejemplo,

máximo-max o promedio-average), como se muestra en Fig. 2.21. Normalmente, la agrupación se produce en bloques que no se solapan (es decir, el paso o stride es igual al tamaño de la agrupación) y usualmente se usa una stride mayor que uno cuando se quiere que haya una reducción en la dimensión de la representación (mapa de características).

2x2 pooling, stride 2

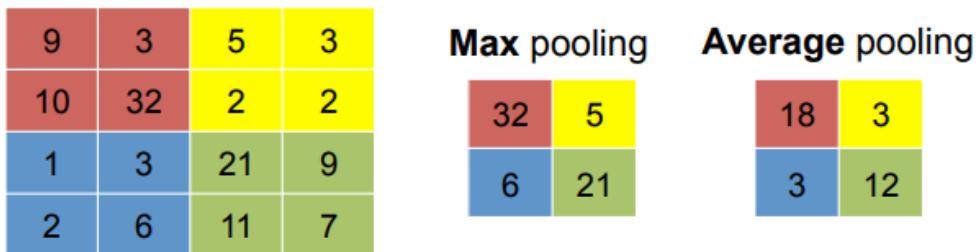


Figura 2.21: Formas de agrupamiento(pooling)
Fuente: Jia et al. (2014)

Después de algunas capas ReLU, es común optar por aplicar una capa de agrupamiento. En esta categoría, la operación MAX (conocida como Max-pooling) es la más popular reduciendo progresivamente el tamaño espacial de la representación de una imagen (mapa de características) mientras conserva la información más importante en ella, esto se ejecuta con dos objetivos, el primero de reducir la cantidad de parámetros y reducir el cálculo en la red. Este último de gran importancia en el control del sobreajuste, (Rohrer, 2016).

La capa de agrupación opera independientemente en cada segmento de profundidad de la entrada y la cambia de tamaño espacialmente (reduce su resolución). El proceso matemático consiste en

pasar una pequeña ventana(kernel) a través de una imagen y tomar el valor máximo de la ventana en cada paso.



Figura 2.22: Operación MAX-pooling en capa de Agrupación
Fuente: Rohrer (2016)

Este proceso es aplicado para cada mapa de activación (salida de la capa de convolución). Análogamente con la capa de convolución, el resultado en esta capa es un conjunto de imágenes que muestran una versión agrupada de las imágenes de entrada.

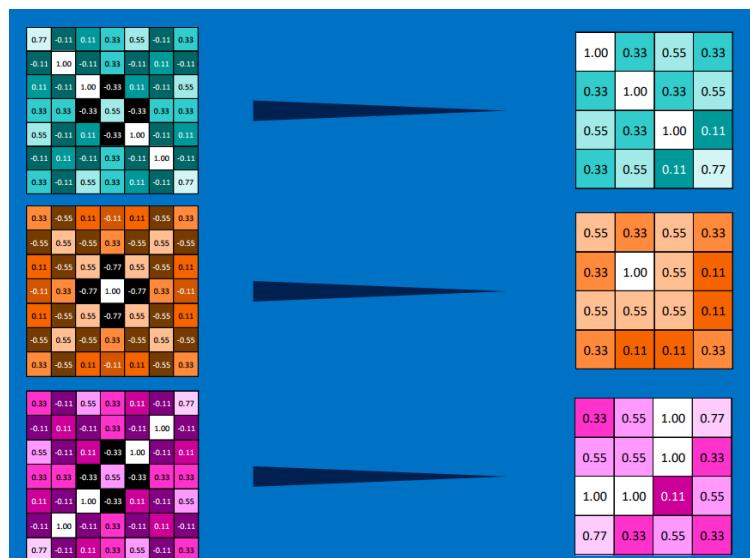


Figura 2.23: Resultado de Agrupación
Fuente: Rohrer (2016)

Esta capa pooling recibe como dato de entrada los parámetros: $W_0 \times H_0 \times D_0$

Produce un dato de salida con los siguientes parámetros: $W_1 \times H_1 \times D_1$

En el que:

$$W_1 = \frac{W_0 - F}{S} + 1$$

$$H_1 = \frac{H_0 - F}{S} + 1$$

$$D_1 = D_0$$

Donde:

W es el ancho (width) de la imagen,

H es la altura (height) de la imagen,

D es la profundidad (depth) de la imagen,

F es la extensión espacial (spatial extent) del kernel,

S es el paso (stride) del kernel,

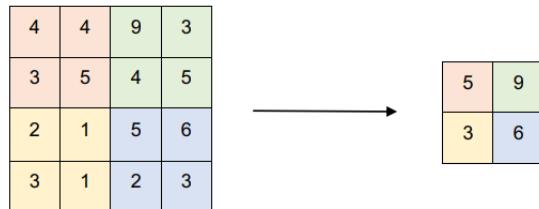


Figura 2.24: Max-pooling con filtro 2x2 y paso 2

Fuente: Elaboración propia

2.3.2. Capa totalmente conectada (Fully-connected layer)

Esta capa por lo general aparece al final del modelo y es similar al Perceptrón Multicapa (MultilayerPerceptron - MLP), en el cual la neurona de salida se conecta a todas las neuronas de entrada y el peso de las conexiones son actualizadas usando el método de retropropagación.

Eventualmente, con un mapa de características lo suficientemente pequeño, el contenido se aplastará en un vector de una sola dimensión y será entrada para en un MLP totalmente conectado para su procesamiento.

Habiendo detallado que la función ReLU es la más utilizada en las capas anteriores, en esta última capa del modelo generalmente se utiliza una función de activación diferente, porque en esta capa se pretende que tenga una salida determinada. La **función Softmax** es muy popular cuando se hace la clasificación y es por ello que se utilizará en esta última capa (Krizhevsky et al., 2012).

En resumen, un modelo de red convolucional estará compuesto de varias capas y ejecutará las siguientes actividades:

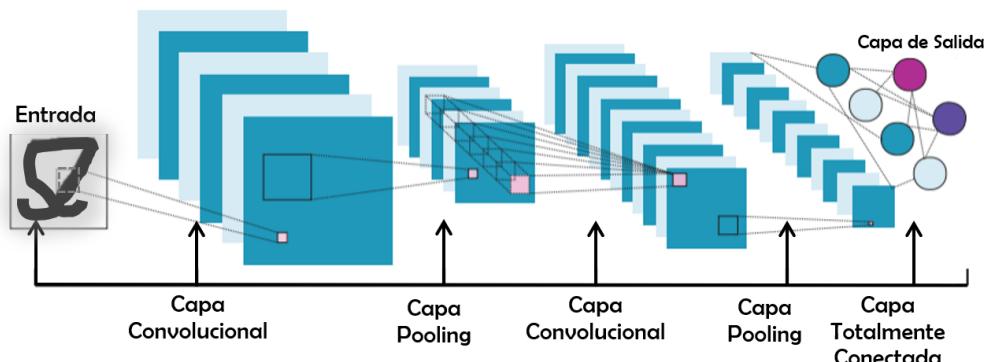


Figura 2.25: Modelo de red neuronal convolucional profunda

Fuente: Elaboración propia

- Se pasa una imagen de entrada a la primera capa convolucional, cuya salida se obtiene como un mapa de activación. Los filtros aplicados en la capa de convolución extraen características relevantes de la imagen de entrada para pasar más lejos.
- Los valores de cada filtro se irán ajustando durante el entrenamiento y en caso de que necesitemos retener el tamaño de la imagen, usamos el mismo relleno (padding cero), de lo contrario se usa el relleno válido ya que ayuda a reducir el número de características.

- Las capas de agrupamiento se agregan para reducir aún más el número de parámetros.
- Se agregan varias capas de convolución y agrupación antes de realizar la clasificación.
- A medida que profundizamos en la red convolucional, se extraen características más específicas en comparación con una red lineal donde las características extraídas son más genéricas.
- La capa de salida en una CNN, como se mencionó anteriormente, es una capa totalmente conectada, donde la entrada de las otras capas se aplana y se envía para transformar la salida en el número de clases que desea la red.
- La salida se genera a través de la capa de salida y se compara con una vector de resultados deseados para la generación de errores. Una función de pérdida se define en la capa de salida totalmente conectada para calcular el gradiente de error.
- El error se retroproyecta para actualizar los valores de filtro (pesos) y sesgo (bias).
- Finalmente, un ciclo de entrenamiento se completa en un solo pase hacia adelante y hacia atrás. Por lo que se tiene que repetir todo el proceso durante varias iteraciones para obtener mejores resultados.

2.4. Entrenamiento y Validación

La red procesa los registros en los datos de entrenamiento uno a la vez, usando los pesos, biases y las funciones en las capas ocultas, luego compara las salidas resultantes con las salidas deseadas.

Los errores se propagan a través del sistema, lo que hace que el sistema ajuste los pesos y biases que serán procesados en la siguiente iteración de entrenamiento. Este proceso ocurre una y otra vez para el mismo conjunto de datos, a medida que los pesos se ajustan (refinan) continuamente.

Para realizar este proceso, existe un método de optimización muy popular denominado **Descenso de gradiente** (Gradient Descent).

2.4.1. Descenso de Gradiente

Un gradiente mide cuánto cambia la salida de una función si se cambia un poco las entradas.

En el caso del entrenamiento de una red neuronal, el gradiente simplemente mide el cambio en todos los pesos con respecto al cambio en el error. El gradiente se puede representar como la pendiente de una función. Cuanto más alto es el gradiente, más pronunciada es la pendiente y más rápido puede aprender un modelo. Pero si la pendiente es cero, el modelo deja de aprender. Dicho matemáticamente, un gradiente es una derivada parcial con respecto a sus entradas, (Dong and Zhou, 2008).

El descenso de gradiente es un algoritmo de optimización iterativa utilizado al entrenar un

modelo de aprendizaje automático, basado en una función convexa, que ajusta sus parámetros iterativamente para minimizar la función de pérdida (error) hasta su mínimo local, (Dong and Zhou, 2008).

La idea detrás del descenso de gradiente es disminuir de forma gradual, pero constante, el error de salida ajustando los pesos y biases. Intuitivamente, se conoce que si un cambio en un peso aumentará o disminuirá el error, entonces queremos disminuir o aumentar ese peso. Matemáticamente, representa el cambio en el error dado un cambio de unidad en el peso:

$$\frac{\partial E}{\partial w_{ij}}$$

Ecuación de la derivada del error con respecto al peso

Una vez que encontramos esta derivada, actualizamos el peso a través de la siguiente fórmula:

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}}$$

Representación del ajuste del peso, donde η es la tasa de aprendizaje

La tasa de Aprendizaje suele disminuir gradualmente durante las épocas de la fase de entrenamiento. Si actualizamos todos los pesos usando esta misma fórmula, esto equivale a moverse en la dirección de descenso más pronunciado a lo largo de la superficie de error, de ahí el nombre, descenso de gradiente.



Figura 2.26: Descenso de Gradiente a través de ajustes en los pesos y biases
Donges (2018)

Se tienen diversos tipos de descenso de gradiente, estos son:

2.4.1.1. Gradiente Descendiente por Lotes (Batch Gradient Descent)

Calcula el error para cada ejemplo dentro del conjunto de datos de entrenamiento, pero el modelo se actualiza solo después de que se hayan evaluado todos los ejemplos de entrenamiento. Todo este proceso es como un ciclo y se denomina época de entrenamiento.

2.4.1.2. Gradiente Descendiente Estocástico (Stochastic Gradient Descent-SGD)

Por el contrario, hace esto para cada ejemplo de entrenamiento dentro del conjunto de datos. Esto significa que actualiza los parámetros para cada ejemplo de entrenamiento, uno por uno. Esto puede hacer que el SGD sea más rápido que el Descenso de gradiente por lotes, dependiendo del problema. Una ventaja es que las actualizaciones frecuentes nos permiten tener una tasa de mejora bastante detallada. El hecho es que las actualizaciones frecuentes son más costosas desde el punto de vista computacional que el enfoque del BGD y la frecuencia de esas actualizaciones

también puede generar gradientes ruidosos, lo que puede hacer que la tasa de error salte, en lugar de disminuir lentamente.

El MOMENTUM es otro argumento en el optimizador SGD que podríamos ajustar para obtener una convergencia más rápida. Ayuda al vector de parámetros a aumentar la velocidad en cualquier dirección con un descenso constante del gradiente para evitar oscilaciones. Una elección típica de momento es entre 0.5 a 0.9.

2.4.1.3. Mini-batch Gradient Descent

Es el método de gradiente más usado, ya que es una combinación de los conceptos de SGD y Batch Gradient Descent. Simplemente divide el conjunto de datos de entrenamiento en pequeños lotes y realiza una actualización para cada uno de estos lotes. Por lo tanto, crea un equilibrio entre la robustez del descenso del gradiente estocástico y la eficiencia del descenso del gradiente discontinuo. Este método es el usado en esta investigación.

2.4.2. Tasa de Aprendizaje (Learning Rate)

Uno de los hiperparámetros clave durante el entrenamiento de una red neuronal es la velocidad/tasa de aprendizaje para el descenso del gradiente. Este parámetro puede entenderse como el tamaño del paso en la optimización para minimizar la función de pérdida de la red, es decir, es un parámetro que determina cuánto influye un paso de actualización en el valor actual de los pesos, (Jordan, 2018).

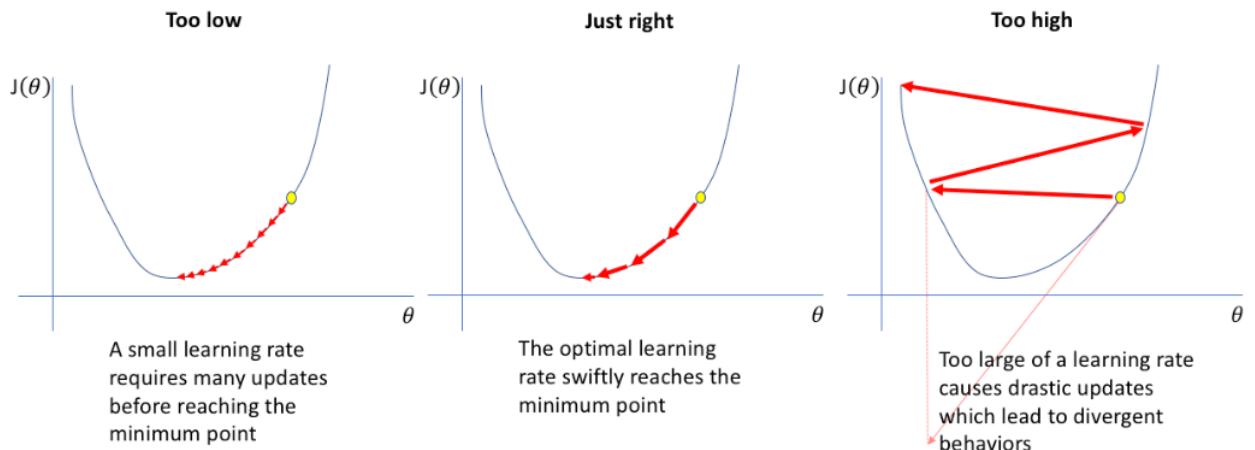


Figura 2.27: Establecimiento de la Tasa de Aprendizaje
Jordan (2018)

Cuando la tasa de aprendizaje es demasiado pequeña, es necesario realizar muchas iteraciones de aprendizaje; pero cuando la tasa de aprendizaje es demasiado grande, el resultado se moverá hacia adelante y hacia atrás en ambos sentidos de los valores extremos(oscila), y no se podrá lograr la solución óptima como puede ser observado en la figura 2.27.

Debido a esto es que al entrenar redes neuronales profundas, a menudo es útil reducir la tasa de aprendizaje a medida que avanza el entrenamiento y no mantenerlo constante y así evitar la divergencia (punto a partir del cual la pérdida ya no se reduce y en lugar de eso comienza a incrementarse).

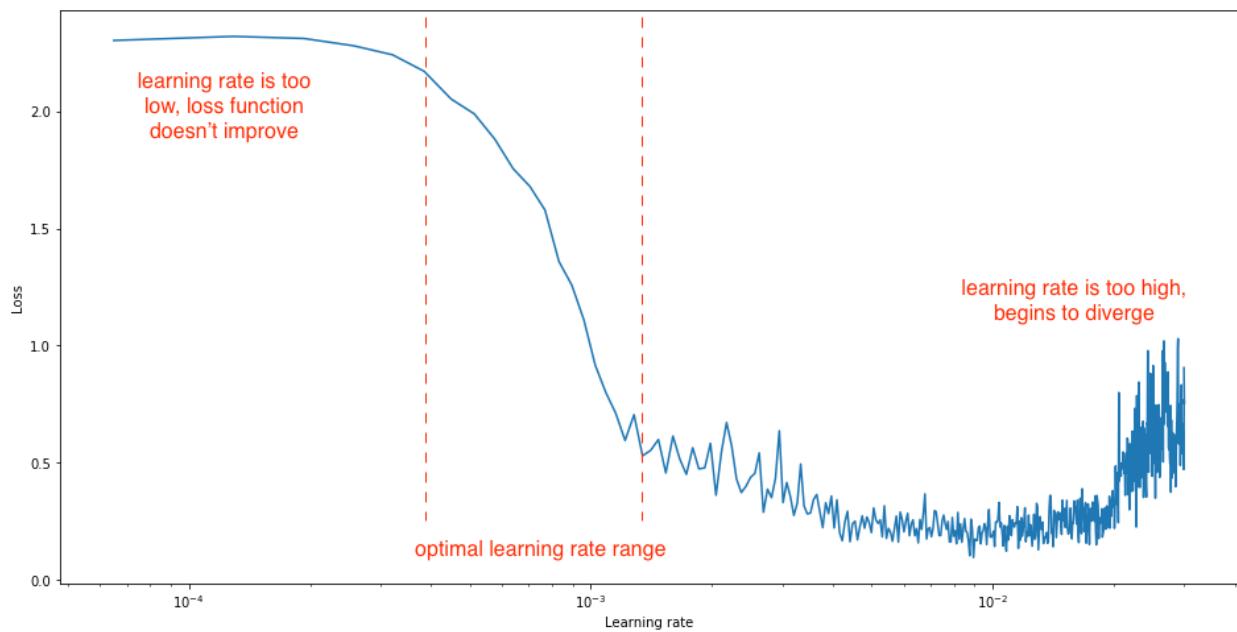


Figura 2.28: Comportamientos en la reducción de la Tasa de Aprendizaje
Jordan (2018)

Reduciendo lentamente la tasa de aprendizaje a lo largo del tiempo ayuda a acelerar el aprendizaje. Esto es lo que se conoce como tasa de decaimiento de aprendizaje (**learning rate decay**).

entrenamiento/learning_rate_decay



Figura 2.29: Ejemplo de Tasa de decaimiento de aprendizaje por época
Fuente: Elaboración propia

Sin embargo, el optimizador Gradient Descent (en cualquiera de sus tipos) con una tasa de decaimiento de aprendizaje, no es usado a menudo para entrenar redes neuronales profundas ya que genera un desafío el determinar los hiperparámetros que deben definirse de antemano y dependen en gran medida del tipo de modelo y problema. Además de que la misma tasa de aprendizaje se aplica a todas las actualizaciones de los parámetros. Por lo que opta en usar optimizadores más avanzados que tienen una tasa de convergencia más rápida y adaptable a diversas situaciones. Proporcionan una alternativa a los clásicos y son conocidos como optimizadores de descenso de gradiente de segundo orden; tales como optimizador Adagrad, Adadelta, RMSprop, Adam, entre otros.

En esta investigación se usarán los métodos de Descenso de Gradiente Mini-batch y el método Adam como optimizador de segundo orden.

2.4.3. Optimizador ADAM

El método calcula las tasas de aprendizaje individuales de manera adaptativa para diferentes parámetros a partir de las estimaciones de los primeros y segundos momentos de los gradientes; el nombre Adam deriva de Estimación del Momento Adaptativo (Kingma and Ba, 2014),

Permite que la tasa de aprendizaje se adapte según los parámetros y realiza actualizaciones más grandes para parámetros infrecuentes y actualizaciones más pequeñas para los frecuentes. Debido a esto, es adecuado para datos dispersos, como lo es en el procesamiento de lenguaje natural o

en el reconocimiento de imágenes. Otra ventaja es que básicamente simplifica la necesidad de ajustar la tasa de aprendizaje debido a que cada parámetro tiene su propia tasa de aprendizaje, sin embargo, la tasa de aprendizaje no disminuye monótonamente. Además de almacenar las tasas de aprendizaje para cada uno de los parámetros, también almacena los cambios de momento para cada uno de ellos por separado.

2.4.4. Validación Cruzada

Para el entrenamiento, el conjunto de datos se divide en un subconjunto de entrenamiento y otro para validación. En esta investigación se dividirá el 75 % para entrenamiento y 25 % para validación, (Elkan, 2012). La división en subgrupos de entrenamiento y validación generalmente se realiza de manera aleatoria, para garantizar que ambos subconjuntos sean muestras aleatorias de la misma distribución. Puede ser razonable realizar un muestreo estratificado, lo que significa asegurar que cada clase esté presente en la misma proporción en los subconjuntos de entrenamiento y prueba. En esta investigación se trabajará con 2 tipos de muestreos (balanceado/estratificado y no balanceado)

El subconjunto de validación es utilizado para evaluar el desempeño del entrenamiento en diferentes arquitecturas del modelo (diferentes topologías), para luego escoger una de ellas. Este conjunto de validación se usa para verificar si el error está dentro de algún rango y no se usa directamente para ajustar los pesos, por el contrario, **se usa para indicar el número óptimo de**

neuronas ocultas o determina el punto de parada para el entrenamiento. Por lo tanto, el modelo es entrenado sobre el conjunto de entrenamiento completo y la capacidad de generalización es medida en el conjunto de validación.

La validación cruzada o cross-validation es una técnica utilizada para evaluar los resultados de un análisis estadístico y garantizar que son independientes de la partición entre datos de entrenamiento y validación, (Moore, 2001). En problemas de clasificación, debido a que los datos se dividen en clases finitas, es natural pensar que la categoría al cual pertenece cierto resultado se dará a través de probabilidades. En probabilidad, la entropía cruzada es la distancia entre las dos distribuciones de probabilidad y se usa generalmente como la función de pérdida. Es por ello que el resultado de la validación cruzada representa la pérdida de la entropía del conjunto de datos. La entropía se suele utilizar en la teoría de la información para medir la pureza o impureza de un conjunto determinado. La pregunta a la que responde es: ¿Cuánto diferente esos elementos son entre sí?

$$H(p, q) = - \sum_{\forall x} p(x) \log(q(x))$$

Fórmula de entropía cruzada con dos distribuciones sobre la variable discreta x , donde $q(x)$ es la estimación para la clasificación verdadera $p(x)$

Para una red neuronal, normalmente verá la ecuación escrita en una forma donde y es el vector de verdad y la variable \hat{y} (o algún otro valor tomado directamente de la última salida de la capa) es

la estimación, y se vería así para un solo ejemplo:

$$L = -\mathbf{y} \cdot \log(\hat{\mathbf{y}})$$

A menudo esta ecuación es promediada sobre todos los ejemplos como una función de costo. No siempre se cumple estrictamente en las descripciones, pero generalmente una función de pérdida es de nivel inferior y describe cómo una sola instancia o componente determina un valor de error, mientras que una función de costo es de nivel superior y describe cómo se evalúa un sistema completo para la optimización. Una función de costo basada en pérdida de clasificación de múltiples clases para un conjunto de datos de tamaño N podría verse así:

$$J = -\frac{1}{N} \left(\sum_{i=1}^N \mathbf{y}_i \cdot \log(\hat{\mathbf{y}}_i) \right)$$

Cálculo de la función de costo en la validación de clasificación de datos

Es por ello que la función de costo/pérdida dada por la validación cruzada ayuda para decidir cuándo se debe terminar el entrenamiento de una red, (Romero, 2015b). Usualmente se utiliza el error cuadrático medio como función de pérdida para medir el rendimiento de un modelo de regresión. Sin embargo, en casos de clasificación, la función de costo de pérdida a través del cálculo de la entropía cruzada es preferible, ya que tiende a no causar saturación de las neuronas de salida, produciendo un aprendizaje más rápido, (Romero, 2015a). La pérdida de entropía cruzada aumentará a medida que la probabilidad prevista diverge de la clasificación real. Por lo tanto, un modelo casi perfecto tendría una pérdida de entropía cercana a cero, (Golik et al., 2013).

2.4.5. Función Softmax

Conocida también como función exponencial normalizada, es la función de activación utilizada en la última capa de una red neuronal. La salida de una red neuronal completamente conectada no es una distribución de probabilidad, sin embargo, el uso de esta función ayuda a obtenerla. Por lo que esta función permitirá estimar la probabilidad de que la imagen de entrada pertenezca a cada una de las clases al realizar una normalización con el objetivo que el valor de cada neurona esté limitado entre cero y uno y así permitir que los resultados de las neuronas de salida sumen a uno.

$$P(y = j \mid \mathbf{x}) = \frac{e^{\mathbf{x}^\top \mathbf{w}_j}}{\sum_{k=1}^K e^{\mathbf{x}^\top \mathbf{w}_k}}$$

Función Softmax

Esto se puede ver como la composición de K funciones lineales $x \mapsto \mathbf{x}^\top \mathbf{w}_1, \dots, x \mapsto \mathbf{x}^\top \mathbf{w}_K$, donde $\mathbf{x}^\top \mathbf{w}$ denota el producto interno de x (entrada) y w (peso). La operación es equivalente a aplicar un operador lineal definido por w a vectores x , transformando así la entrada original, probablemente altamente dimensional (reales arbitrarios), en vectores K -dimensional de valores reales en el rango $[0, 1]$ que suman 1. (Bishop, 2006).

Esta función permite que al estar normalizadas las salidas de la segunda capa totalmente conectada, pueda ser aplicada la validación cruzada y conocer la pérdida de entropía.

2.4.6. Método de Regularización L2

En la regularización, lo que se hace normalmente es mantener el mismo número de funciones, pero se reduce la magnitud de los coeficientes, sin embargo, el método de Regularización L2 o también conocido como Regresión Lasso (Lasso Regression - Least Absolute Shrinkage and Selection Operator), trata de agregar un término adicional(λ) a la función de pérdida, la cual penaliza parametrizaciones con pesos más elevados, es decir, reduce el coeficiente de la función menos importante a cero, eliminando así algunas características. Por lo tanto, esto funciona bien para la selección de funciones en caso de que tengamos una gran cantidad de funciones, (Romero, 2015b).

Si $L(\theta, D)$ es la función de pérdida, θ es el conjunto de parámetros libres y D es un ejemplo de entrenamiento, entonces a la función de pérdida regularizada será:

$$E(\theta, D) = L(\theta, D) + \lambda R(\theta)$$

Donde $R(\theta)$ caracteriza la complejidad del modelo y λ representa la proporción de la pérdida total del modelo, generalmente λ es una constante y estará cerca de 0 porque si λ es demasiado grande, conducirá a un ajuste insuficiente (underfitting). Por otro lado, para evitar el exceso de ajuste (overfitting) cuando se tiene una gran cantidad de características en su conjunto de datos, no optimizamos directamente $L(\theta, D)$ (la entropía cruzada), sino que optimizamos $L(\theta, D) + \lambda R(\theta)$.

En esta investigación se usará una constante **lambda(λ) = 0.0001**. La regularización de pérdida L2 solo debe incluir los pesos de las capas totalmente conectadas, y normalmente no incluye a los sesgos (biases). La intuición detrás de esto es que el sesgo contribuye al overfitting (sobreajuste), y no estaría agregando ningún nuevo grado de libertad al modelo.

2.5. Método de la investigación

2.5.1. Tipo de investigación

De acuerdo al fin que se persigue es una investigación de tipo tecnológica y de acuerdo al diseño es una investigación experimental de tipo cuantitativa donde se analizará el rendimiento del modelo algorítmico.

2.5.2. Variables de la Investigación

2.5.2.1. Variable Dependiente

Reconocimiento automático de señales de tránsito vehicular

2.5.2.2. Variable Independiente

Modelo basado en el aprendizaje profundo de redes neuronales convolucionales

2.5.3. Indicadores

Los indicadores nos permiten realizar mediciones y a su vez determinan la validez de la hipótesis planteada en la presente investigación. El desempeño de la clasificación se puede verificar con los siguientes seis indicadores:

Tabla 2.1: Indicadores para la investigación

<u>Indicador</u>	<u>Descripción</u>	<u>Instrumento</u>
Tasa de Verdaderos Positivos (Efectividad - Sensibilidad)	$\frac{Verdaderos\ Positivos}{Verdaderos\ Positivos+Falsos\ Negativos}$	Modelo de Reconocimiento
Tasa de Verdaderos Negativos (Especificidad)	$\frac{Verdaderos\ Negativos}{Verdaderos\ Negativos+Falsos\ Positivos}$	Modelo de Reconocimiento
Valor Predictivo Positivo (Precisión)	$\frac{Verdaderos\ Positivos}{Verdaderos\ Positivos+Falsos\ Positivos}$	Modelo de Reconocimiento
Curvas PR (Precision - Recall)	Relación entre Efectividad(Recall) y Precision	Modelo de Reconocimiento
Curvas ROC (Receiver Operating Characteristic)	Relación entre Efectividad y Especificidad	Modelo de Reconocimiento
Tasa de Acierto (Exactitud)	$\frac{Verdaderos\ Positivos+Verdaderos\ Negativos}{Total\ de\ Imagenes}$	Modelo de Reconocimiento

2.5.4. Recolección de Datos para la Construcción del Modelo

2.5.4.1. Técnica de Recolección

Revisión de la literatura (análisis de documentos).

2.5.4.2. Población

Existen diversas arquitecturas de redes neuronales convolucionales usadas para propósitos específicos. La idea principal es que al principio la arquitectura de la red neuronal toma como entrada una imagen y su especificación de dimensiones en 3 valores (largo, ancho y profundidad). Capas convolucionales y capas de activación(optimización) se apilan juntas y luego son seguidas por capas de agrupamientos. Esta estructura se usa comúnmente y se repite hasta que la entrada (imagen) se fusiona espacialmente a un tamaño pequeño. Después de eso, se envía a capas completamente conectadas y la salida de la última capa completamente conectada, que está al final de la arquitectura, produce los puntajes de clase de la imagen de entrada.

2.5.4.3. Muestra

Para el proceso de diseño e implementación de arquitecturas a elaborar se tomarán en cuenta dos modelos mundialmente conocidos e importantes:

Arquitectura AlexNet

Esta arquitectura hizo que las Redes Convolucionales fueran populares en el campo de Visión por Computadora. AlexNet fue desarrollado por (Krizhevsky et al., 2012). La entrada consiste en una imagen de 224x224 pixeles en formato RGB (3 canales). La arquitectura consta de 8 capas, las primeras cinco capas son convolucionales y el resto son capas totalmente conectadas. La primera y segunda capa convolucional son seguidas por capas de normalización de respuesta local, luego estas capas de normalización de respuesta son seguidas por capas de agrupación máxima. La salida de cada capa convolucional y cada capa completamente conectada se activa a través de una función no linear conocida como RE-LU. Similar a LenNet-5(LeCun et al., 1998) pero más grande, teniendo aproximadamente 60 millones de parámetros.

Arquitectura Inception

Es una arquitectura de red neuronal convolucional profunda, creada por un grupo de investigación de Google que fue responsable de establecer el nuevo estado del arte de la técnica para la clasificación y detección en la competencia de reconocimiento visual a gran escala ImageNet 2014(ILSVRC14).

El principal sello distintivo de esta arquitectura es la utilización mejorada de los recursos informáticos dentro de la red. Esto fue logrado gracias al aumento de la profundidad y el

ancho de la red mientras se mantiene el costo computacional constante. Para optimizar la calidad, las decisiones para elaborar la arquitectura se basaron en el principio Hebbiano y la intuición de procesamiento a escala múltiple. Una encarnación particular utilizada en la competencia ILSVRC14 es llamada GoogLeNet, una red de 22 capas de profundidad, cuya calidad se evalúa en el contexto de reconocimiento y detección, (Szegedy et al., 2014).

2.5.5. Recolección de Datos para el Entrenamiento y Evaluación del Modelo

2.5.5.1. Técnica de Recolección

Revisión de la literatura (análisis de documentos) y captura de imágenes del Perú a través de la aplicación Google Maps.

2.5.5.2. Población

***) Área:** Imágenes de Señales de Seguridad Vial.

***) Categoría:** Tránsito Vehicular Vertical.

***) Subcategoría:** Señales reguladoras, preventivas e informativas.

La población es infinita para esta investigación, debido al número infinito de formas distintas en que una señal de tránsito puede ser capturada. Se tomarán en cuenta imágenes donde se muestren señales de tránsito vehicular del tipo vertical en sus 3 subcategorías.

2.5.5.3. Muestra

Existe una colección(dataset) que se ajusta a las características de nuestra población y será usada para conseguir el objetivo de la investigación, principalmente porque cuenta con abundantes imágenes lo que conforma una muestra representativa y necesaria para hacer generalizaciones.

Señales de Tránsito de Alemania

Conjunto de datos creados a partir de aproximadamente 10 horas de video grabados durante el día mientras se conducía en diferentes tipos de carreteras en Alemania. De las secuencias del video fueron extraídas imágenes de señales de tránsito en formato RGB cuyas dimensiones varían entre 15x15 y 250x250 pixeles. En esta colección se obtuvieron un total de 39209 imágenes distribuidas en 43 clases. (Stallkamp et al., 2011)

Señales de Tránsito de Perú

Conjunto de aproximadamente 614 imágenes originalmente tomadas de la aplicación Google Maps agrupadas en 7 categorías. Dado que la población de imágenes es infinita, se optó por usar un muestreo aleatorio simple para poblaciones desconocidas con un nivel de confianza del 95 % y un error de muestreo del 4 %.

$$n = \frac{z^2 pq}{e^2}$$

En el que:

- n = tamaño de muestra
- z = Coeficiente de confiabilidad 95 % al que corresponde (1.96)
- pq = Varianza de la población, ponemos la varianza mayor posible porque a mayor varianza hará falta una muestra mayor (0.25)
- e = Error muestral (0.04)

2.5.6. Etapas de la investigación

El desarrollo de la investigación comprenderá las siguientes etapas de trabajo a saber:

- a) Investigación bibliográfica de los diferentes temas necesarios para la elaboración de la investigación, tales como dispositivos de control de tránsito, vehículos autónomos, sistemas avanzados de asistencia al conductor, aprendizaje profundo (deep learning), procesamiento de imágenes, entre otros.
- b) Búsqueda de los principales casos de éxito en el reconocimiento de señales de tránsito a través de trabajos de investigación en el Perú como también en otros países.
- c) Recolección de imágenes de señales de tránsito vehicular que se utilizarán en el desarrollo de la investigación.
- d) Análisis e implementación de diversas técnicas de procesamiento de imágenes que serán

aplicadas al dataset recolectado con la finalidad de contribuir en el diseño inicial del modelo convolucional.

- e) Puesto a que las redes neuronales son heurísticas por naturaleza (requieren un constante ajuste de hiper-parámetros para obtener un óptimo resultado), se realizarán diferentes diseños de arquitecturas de modelos convolucionales y se escogerá el modelo que otorgue los mejores resultados.
- f) Para el análisis de resultados, además de los indicadores conocidos con tasa de sensibilidad, especificidad, precisión, se pretende usar una Matriz de confusión. A través de esta matriz se pueden conocer valores estadísticos del espacio ROC (Receiver Operating Characteristic), métrica usada para evaluar la calidad del modelo reconocedor.

2.5.6.1. Metodología para el diseño del Modelo

Se utilizará la metodología propuesta por Tammy Noergard (T. Noergaard, 2005) , la cual toma mucha de las claves de las metodologías conocidas como Rational Unified Process (RUP), Attribute Driven Desing (ADD), Object Oriented Process (OOP) y el Model Driven Architecture (MDA), pero las simplifica. La siguiente figura muestra las diferentes fases de la metodología.

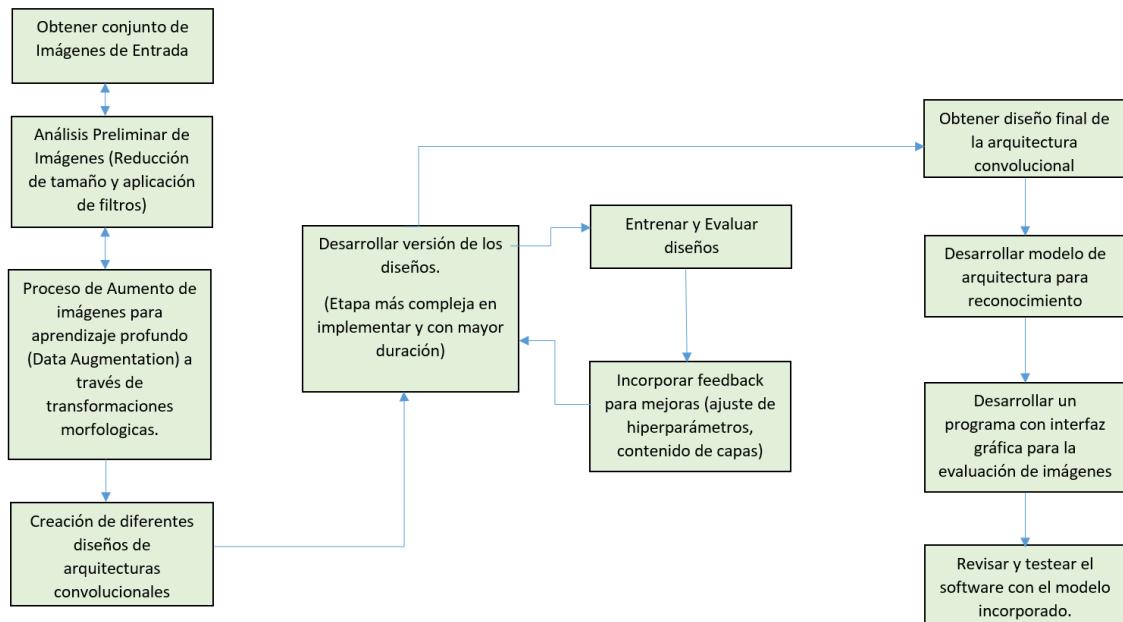


Figura 2.30: Diseño del Modelo del Ciclo de Vida del Desarrollo

Fuente: Elaboración propia

Capítulo 3

Desarrollo de la Investigación

En la siguiente imagen se detallan los procesos de implementación computacional más importantes que fueron realizados durante el desarrollo de la investigación:

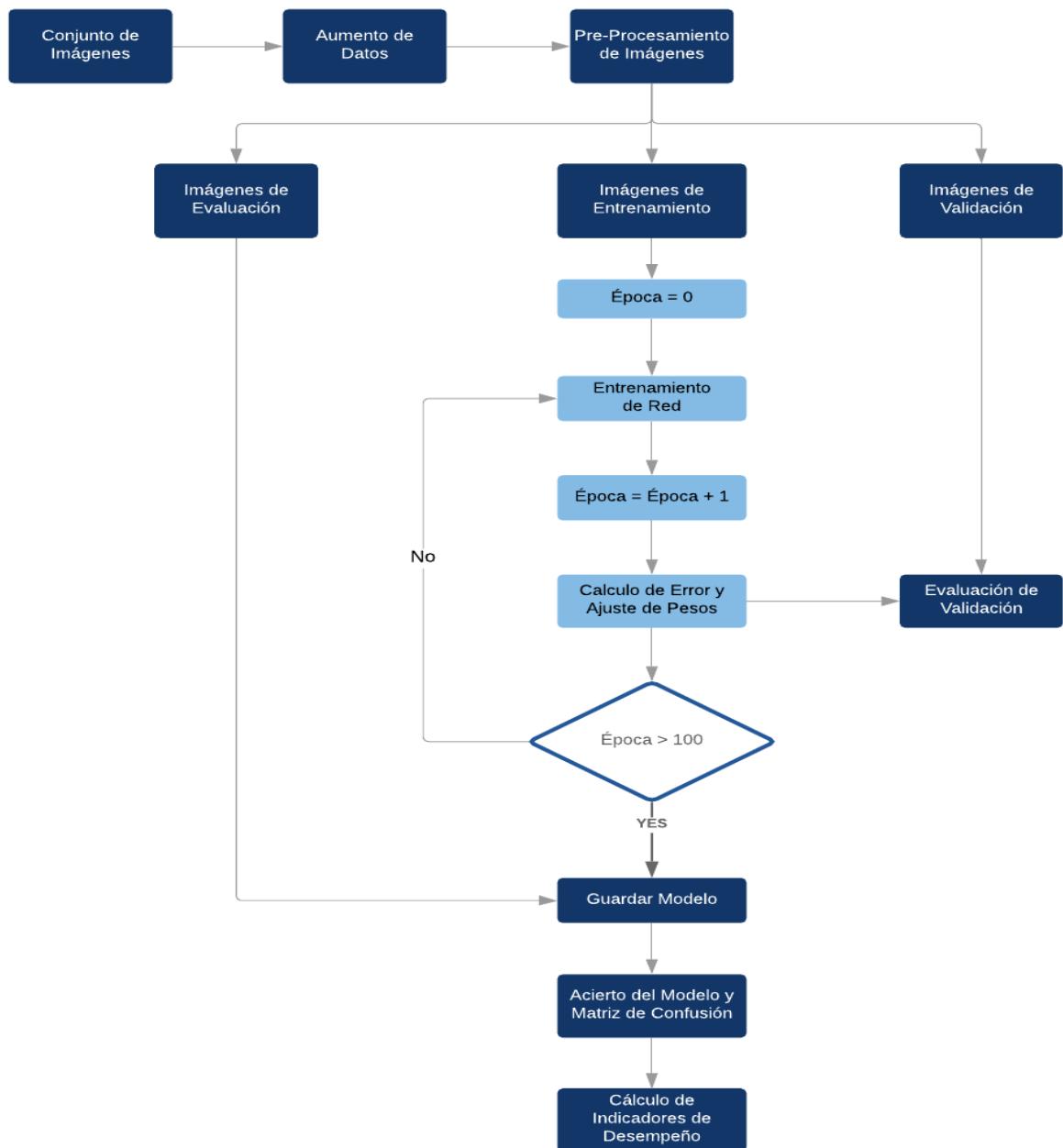


Figura 3.1: Diagrama de flujo para el desarrollo de la Investigación
 Fuente: Elaboración propia

3.1. Análisis del conjunto de Imágenes

En esta etapa de la investigación se busca encontrar imágenes de señales de tránsito, las cuales conforman la principal fuente de los modelos que se pretenden crear. Estas serán usadas durante el entrenamiento, validación y evaluación del modelo pretendido.

3.1.1. Datos Iniciales para el entrenamiento

Se obtienen un conjunto de imágenes iniciales de señales de tránsito de Alemania y Perú, las cuales servirán para entrenar y validar los modelos creados.

3.1.1.1. Señales de Tránsito de Alemania

En esta colección consta de un total de 51839 imágenes distribuidas en 43 clases no necesariamente balanceadas con un tamaño de 32 x 32 pixeles.



Figura 3.2: Speed limit (20km/h)
Fuente: Elaboración propia



Figura 3.3: Speed limit (50km/h)
Fuente: Elaboración propia



Figura 3.4: Bumpy road
Fuente: Elaboración propia



Fuente: Elaboración propia
Figura 3.5: Children crossing



Figura 3.6: Turn left ahead
Fuente: Elaboración propia

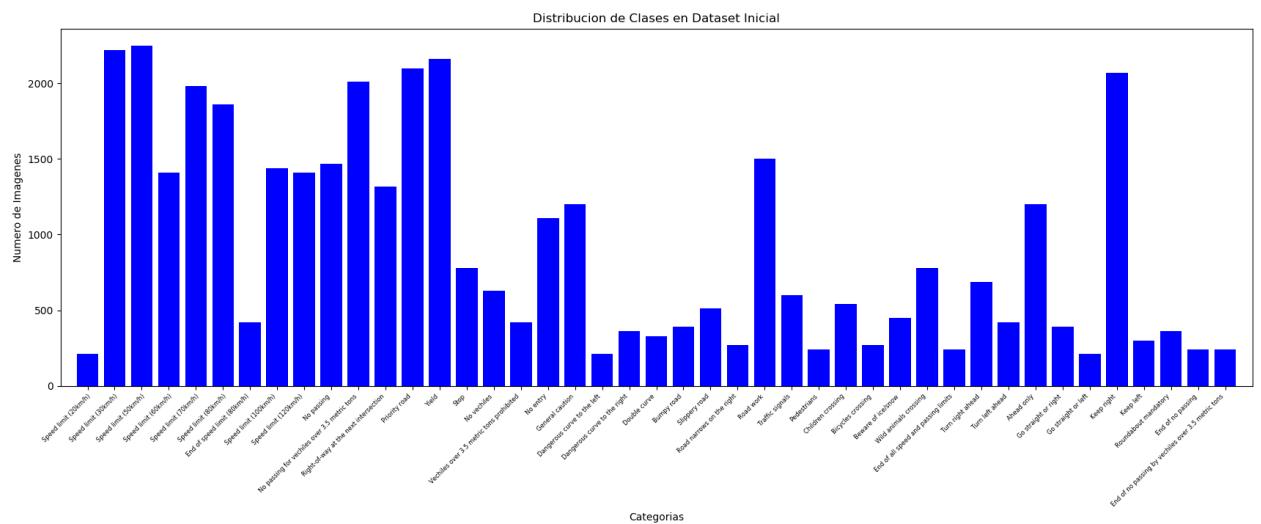


Figura 3.7: Distribución de ejemplos por señal para el entrenamiento (Total 39209)

Fuente: Elaboración propia

3.1.1.2. Señales de Tránsito de Perú

En esta colección consta de un total de 614 imágenes distribuidas en 7 clases no necesariamente balanceadas con un tamaño de 60 x 60 píxeles.



Figura 3.8: Pare
Fuente: Elaboración propia



Figura 3.9: Resalto
Fuente: Elaboración propia



Figura 3.10: Zona Escolar
Fuente: Elaboración propia



Figura 3.11: Peatones
Fuente: Elaboración propia



Figura 3.12: Paradero
Fuente: Elaboración propia



Figura 3.13: No Estacionar
Fuente: Elaboración propia



Figura 3.14: Disminuir Velocidad
Fuente: Elaboración propia

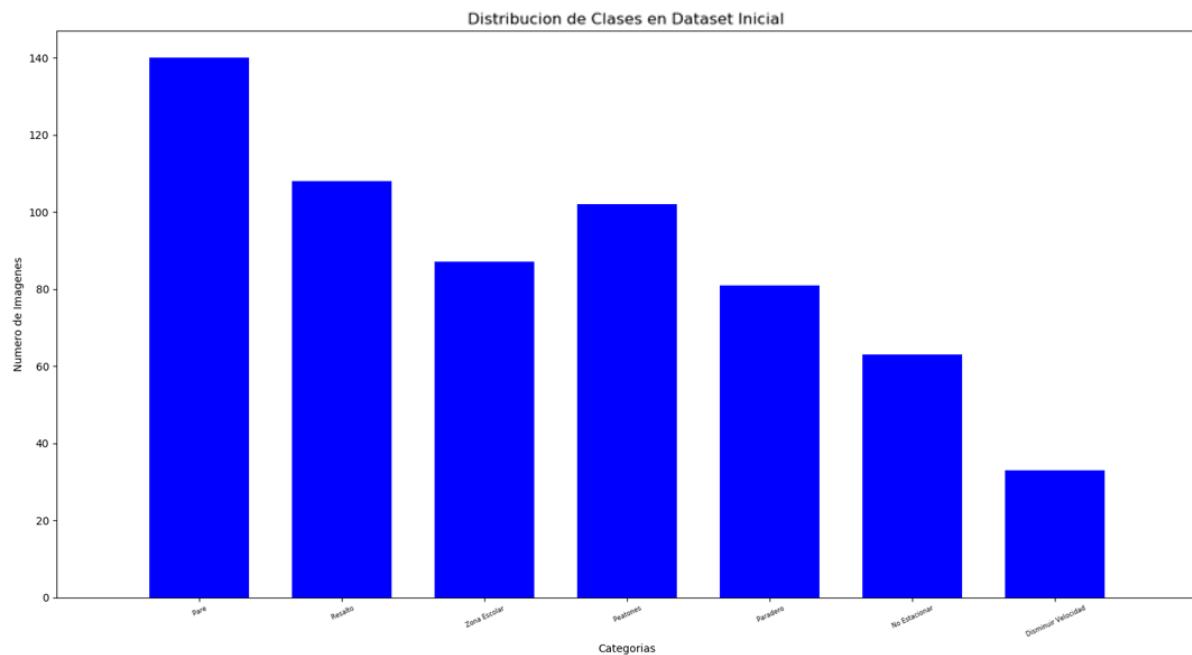


Figura 3.15: Distribución de ejemplos por señal para el entrenamiento (Total 614)
Fuente: Elaboración propia

3.1.2. Datos para la evaluación

Al igual que en el anterior punto, se obtienen un conjunto de menor cantidad de imágenes de señales de tránsito alemán y peruano, las cuales servirán para evaluar los modelos creados.

3.1.2.1. Señales de Tránsito de Alemania

Para la etapa de evaluación se cuenta con un conjunto de 12630 imágenes, de igual manera no necesariamente balanceadas y que no son utilizadas durante el proceso de entrenamiento para obtener resultados confiables a la hora de evaluar el resultado del entrenamiento.

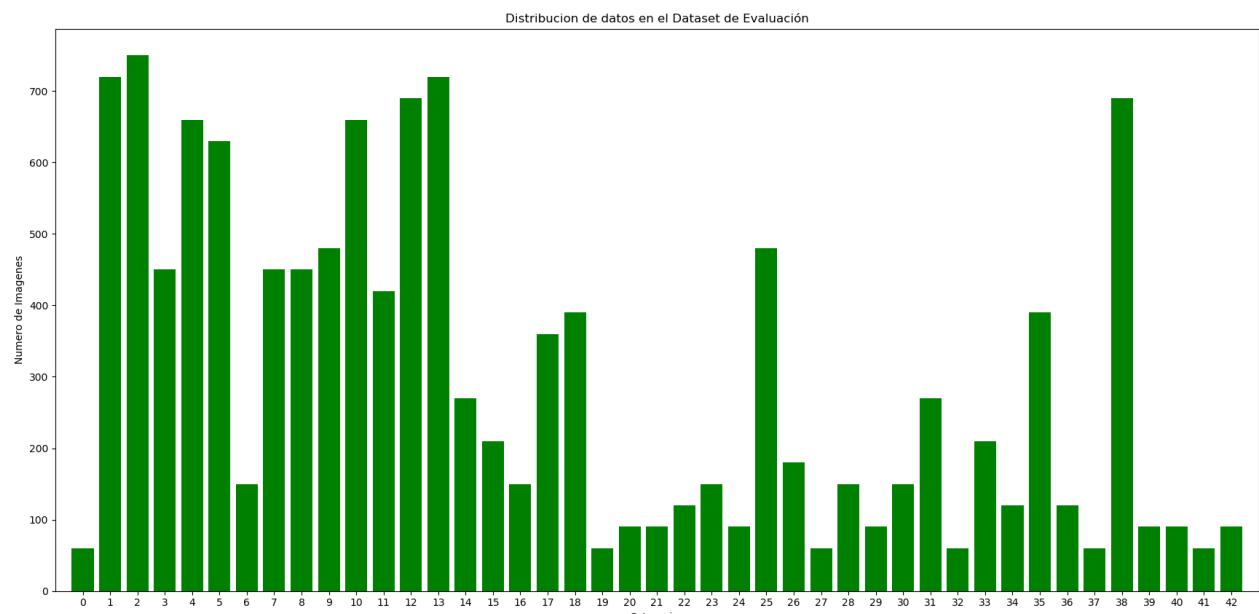


Figura 3.16: Distribución de ejemplos por señal para la evaluación - Alemania

Fuente: Elaboración propia

3.1.2.2. Señales de Tránsito de Perú

Para la etapa de evaluación se cuenta con un conjunto de 4698 imágenes, de igual manera no necesariamente balanceadas y que no son utilizadas durante el proceso de entrenamiento para obtener resultados confiables a la hora de evaluar el resultado del entrenamiento.

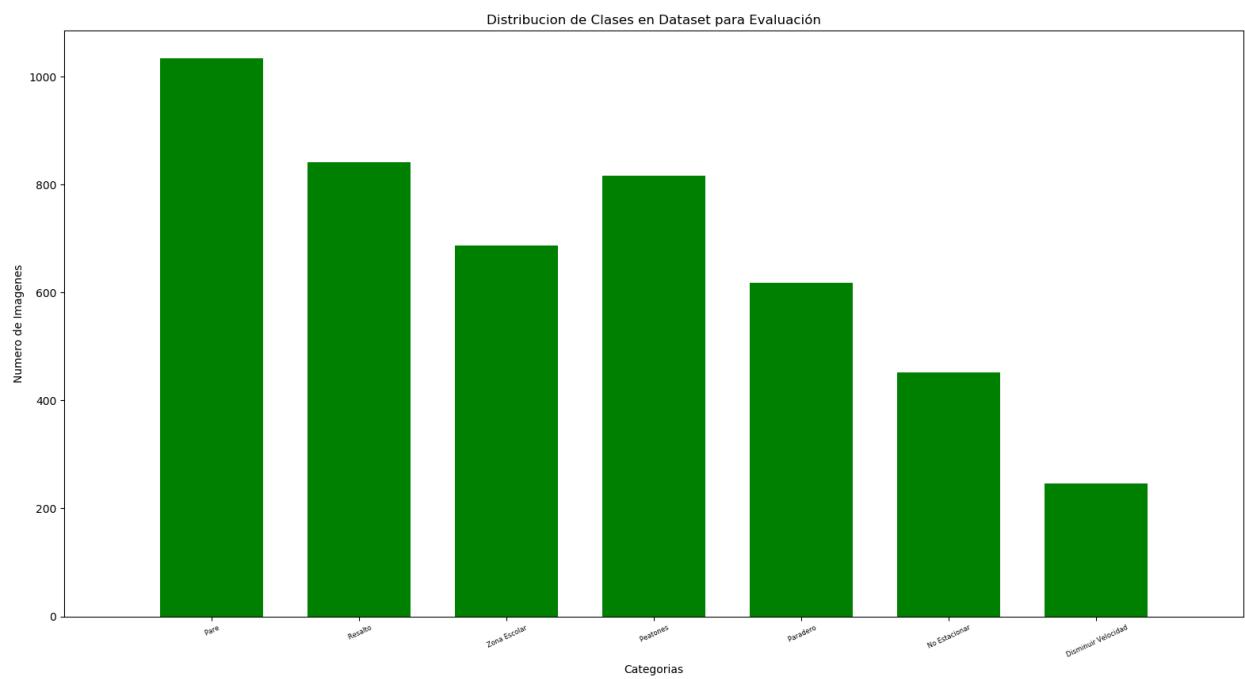


Figura 3.17: Distribución de ejemplos por señal para la evaluación - Perú

Fuente: Elaboración propia

3.1.3. Proceso de Aumento de Datos (Data Augmentation)

Las redes convolucionales durante el aprendizaje profundo requieren una gran cantidad de datos para conseguir realizar un mejor entrenamiento(aprendizaje) y obtener un modelo que generalice eficazmente. Muchas veces esta recopilación de datos suele ser costosa y laboriosa es por eso que

el proceso de Data Augmentation ayuda a superar este problema a través del uso de métodos o técnicas de procesamiento de imágenes. Recientemente se ha utilizado ampliamente el aumento de datos genéricos para mejorar el rendimiento de las Redes Neuronales Convolucionales, (Taylor and Nitschke, 2017).

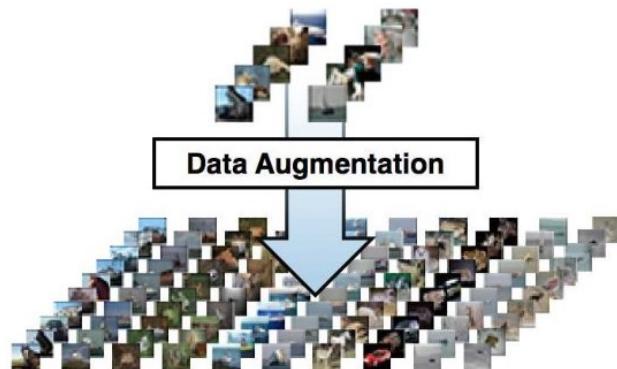


Figura 3.18: El aumento de datos infla artificialmente los conjuntos de datos usando transformaciones que preservan las categorías de los objetos

Taylor and Nitschke (2017)

Son utilizadas las siguientes técnicas:

- 1) Flipping (Horizontal/Vertical)
- 2) Proyección
- 3) Rotación
- 4) Zoom (Out/In)
- 5) Ecualización de Histograma

3.1.3.1. Flipping

Primero, vamos a aplicar un par de trucos para extender nuestros datos volteando. Algunas señales de tráfico son invariantes para voltear horizontal y / o verticalmente, lo que básicamente significa que podemos voltear una imagen y todavía debe clasificarse como perteneciente a la misma clase.

Esta técnica solo fue utilizada para el dataset de señales de Tránsito de Alemania.

Flipping horizontal:



Figura 3.19: Imágenes volteadas horizontalmente

Fuente: Elaboración propia

Flipping Vertical:



Figura 3.20: Imágenes volteadas verticalmente

Fuente: Elaboración propia

Flipping Horizontal y Vertical:



Figura 3.21: Imágenes volteadas primero horizontal y luego verticalmente

Fuente: Elaboración propia

Incluso, hay signos que luego de volverse, deben clasificarse como un signo de alguna otra clase. Esto sigue siendo útil, ya que podemos utilizar los datos de estas clases para ampliar sus contrapartes.

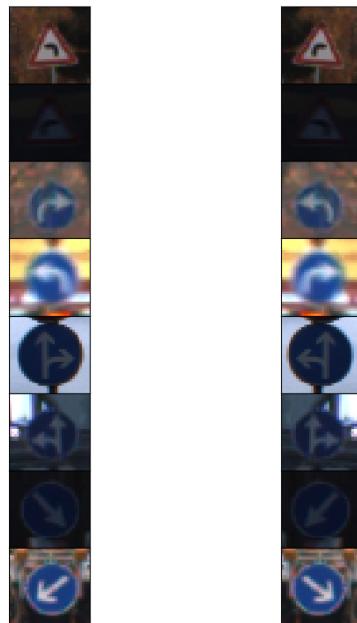


Figura 3.22: Imágenes que volteadas horizontal o verticalmente, cambian su categoría

Fuente: Elaboración propia

Finalmente obtenemos una nueva distribución de datos luego de haber aplicado flipping a ciertas imágenes. Esta distribución consta de 63538 imágenes.

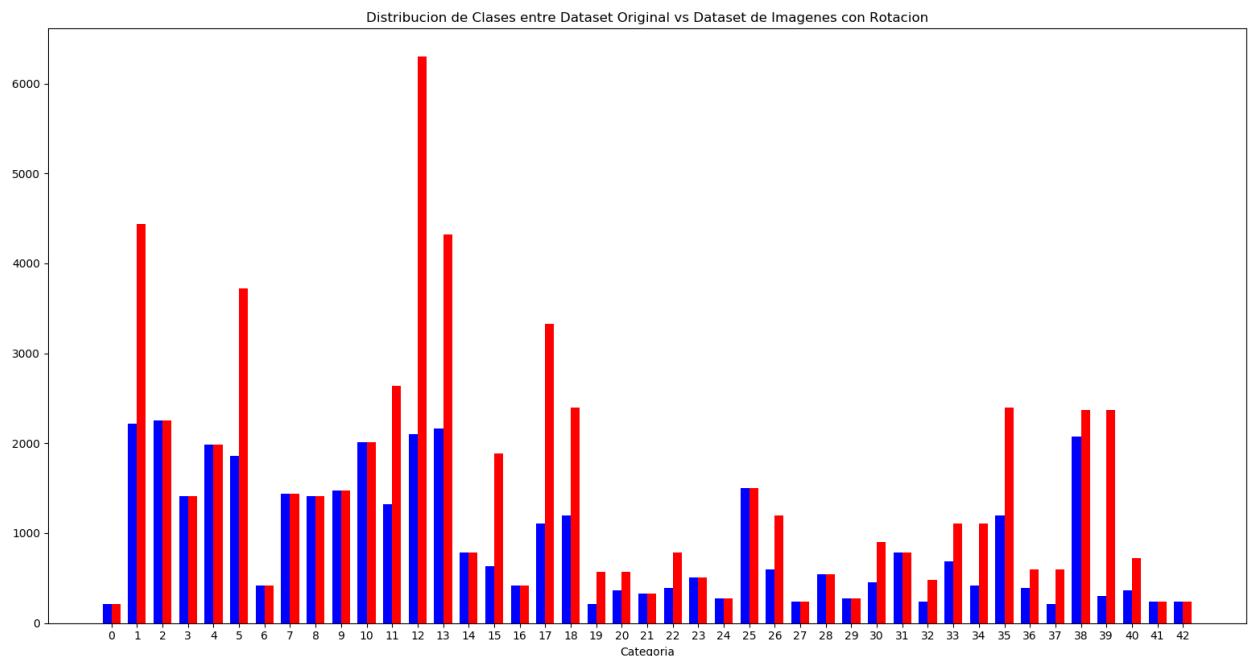


Figura 3.23: Distribución de categoría, luego de aplicar el Flip (en sus distintos tipos)- Señales de Alemania

Fuente: Elaboración propia

A excepción de esta técnica, las siguientes fueron utilizadas para ambos datasets de imágenes (Alemania y Perú).

3.1.3.2. Projection (Proyección)

Implica mover la imagen a lo largo de la dirección X o Y (o ambas). Este método de aumento es muy útil ya que la mayoría de los objetos se pueden ubicar en casi cualquier lugar de la imagen, obligando a la red neuronal convolucional a buscar en todas partes. Los márgenes de proyección se

asignan al azar en un rango que depende del tamaño de la imagen. La transformación de proyección también se ocupar de escalar al azar a medida que colocamos al azar las esquinas de la imagen en un rango [-delta, +delta].

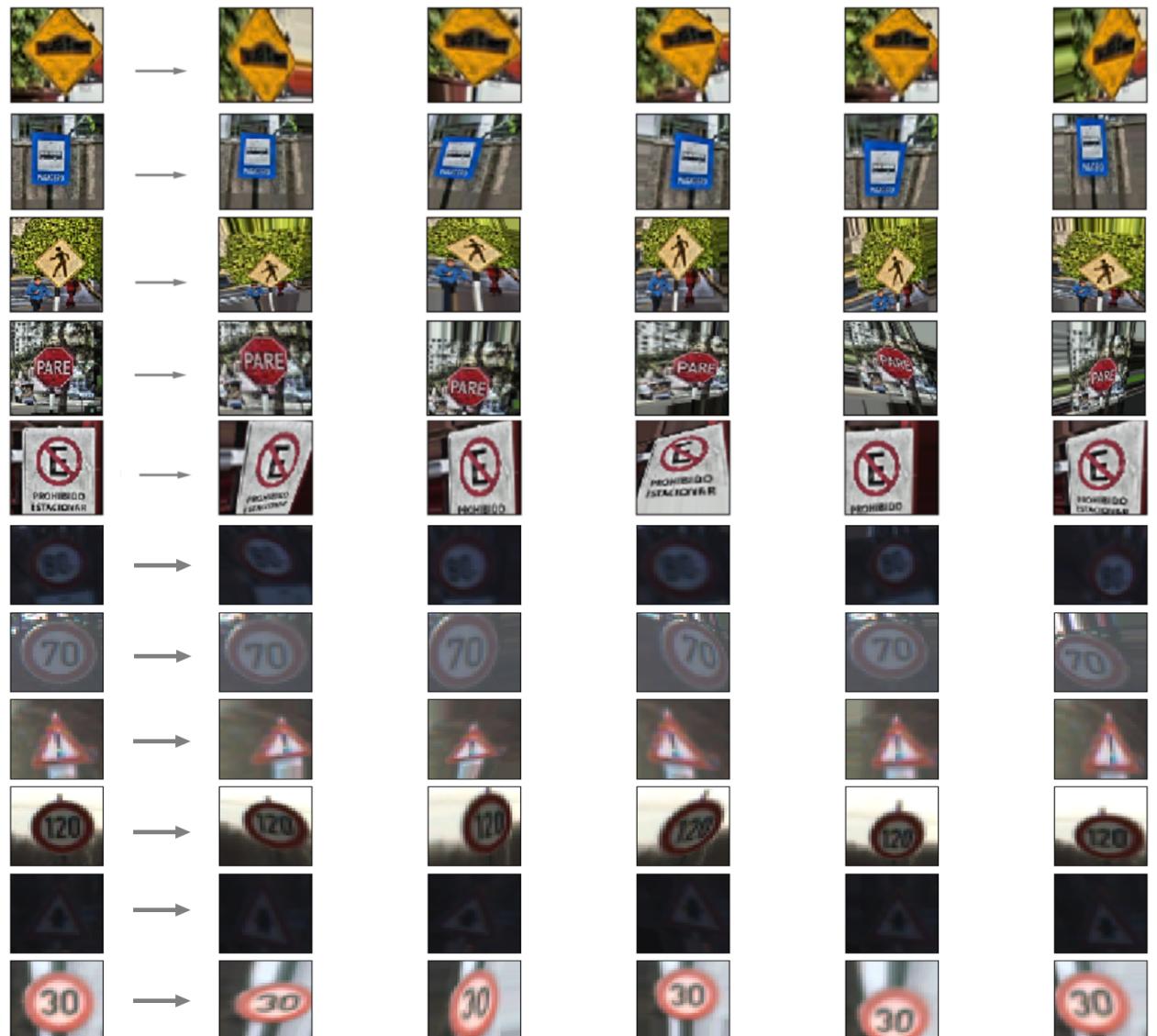


Figura 3.24: Ejemplo de cinco proyecciones por imagen - Dataset Perú y Alemania

Fuente: Elaboración propia

3.1.3.3. Rotation (Rotación)

Aplica la rotación aleatoria en un rango de grados definido (hasta 30°) a un subconjunto aleatorio de imágenes. El rango en sí está sujeto a escala dependiendo de la intensidad de aumento.



Figura 3.25: Ejemplo de cinco rotaciones por imagen - Dataset Alemania y Perú

Fuente: Elaboración propia

3.1.3.4. Zoom

Acerca (zoom in) o aleja (zoom out) la imagen tratando de preservar el fondo.



Figura 3.26: Ejemplo de cinco aplicaciones de zoom(in/out) por cada imagen - Dataset Alemania y Perú

Fuente: Elaboración propia

3.1.3.5. Ecualización del histograma

Aumenta el contraste global de muchas imágenes, especialmente cuando los datos utilizables de la imagen están representados por valores de contraste cercanos. Esto permite que áreas de menor contraste local puedan obtener un mayor contraste. La ecualización del histograma logra esto al distribuir eficazmente los valores de intensidad más frecuentes.

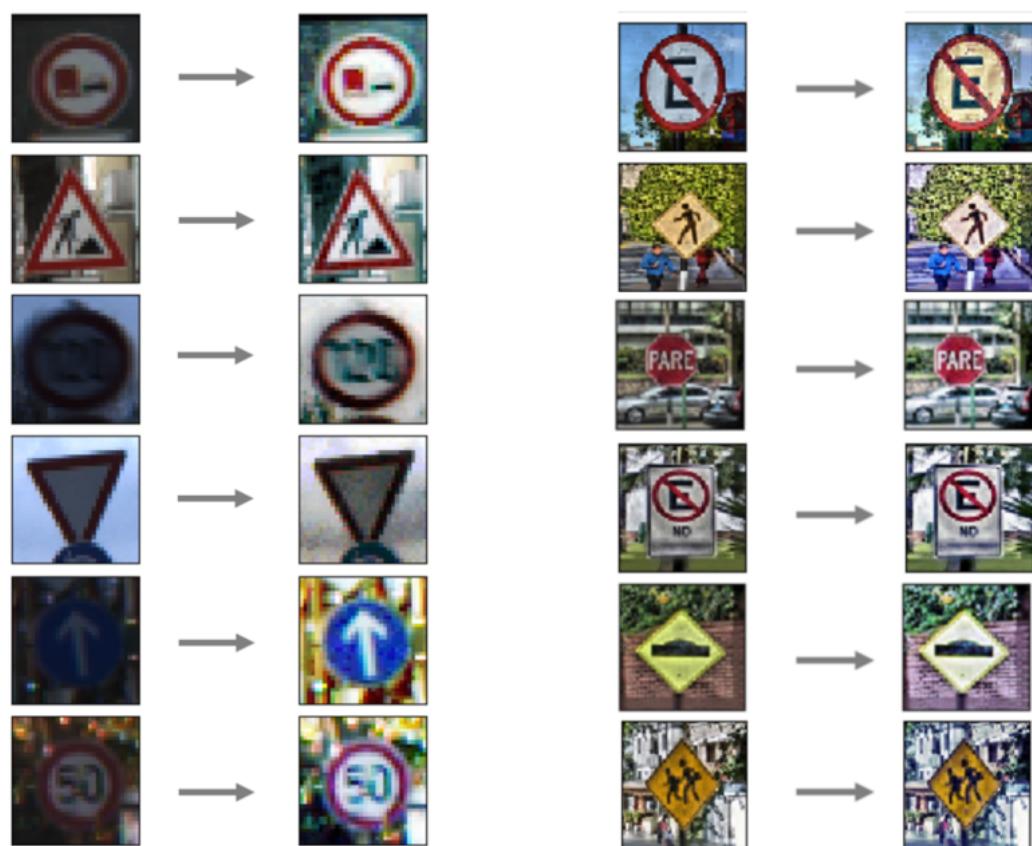


Figura 3.27: Distribución eficaz de los valores de intensidad más frecuentes

Fuente: Elaboración propia

3.1.4. Dataset final para el Entrenamiento

Finalmente, luego de haber aplicado de manera secuencial y/o aleatoriamente estas técnicas a cada una de las imágenes, obtenemos nuevas distribuciones de datos.

3.1.4.1. Señales de Tránsito de Alemania - Dataset Balanceado

En una se tiene un conjunto balanceado de datos con **270900 imágenes**.

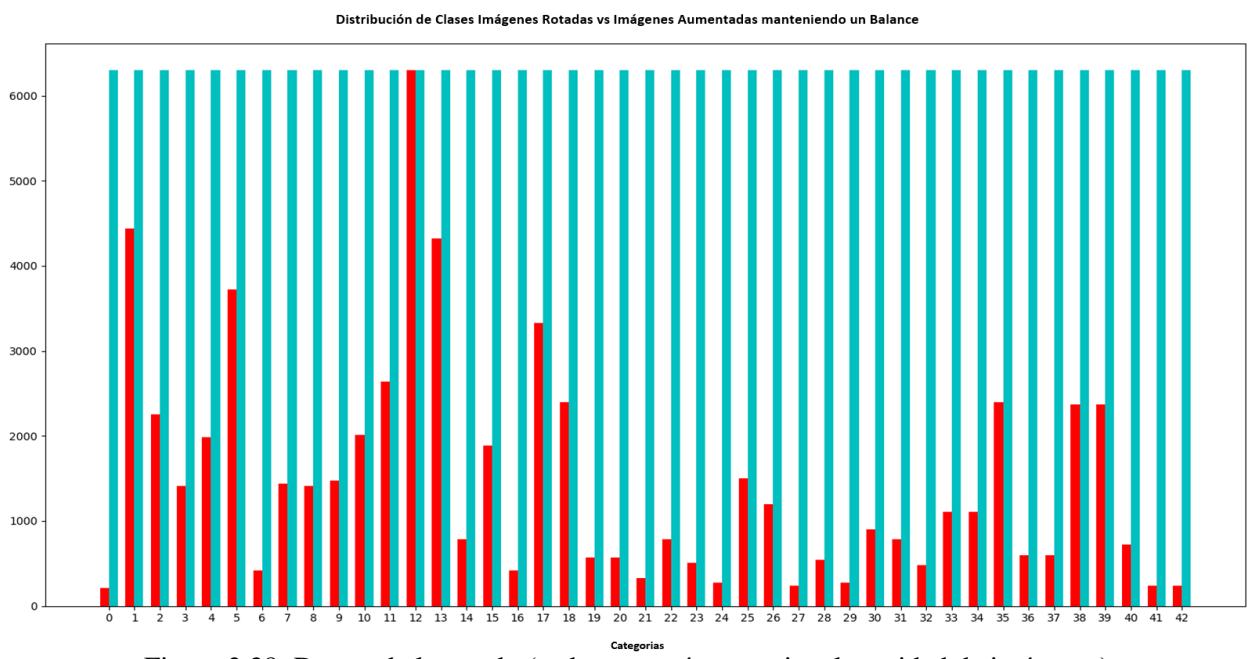


Figura 3.28: Dataset balanceado (cada categoría posee igual cantidad de imágenes)

Fuente: Elaboración propia

Teniendo en cuenta lo descrito en la **Sección 2.3.4 (Validación Cruzada)**, para la etapa del entrenamiento, el conjunto de datos Balanceados compuesto por 270900 imágenes fue dividido en un subconjunto de entrenamiento y otro para validación, con la siguiente distribución:

Tabla 3.1: Distribución Entrenamiento y Validación Dataset Balanceado - Señales de Tránsito de Alemania

<u>CONJUNTO DE DATOS</u>	<u>CANTIDAD IMÁGENES</u>
Entrenamiento	203175 (75 %)
Validación	67725 (25 %)

3.1.4.2. Señales de Tránsito de Perú - Dataset No Balanceado

Con el objetivo de obtener una gran cantidad de datos, para el Dataset de señales de Tránsito del Perú, por cada imagen fueron creadas 50 nuevas imágenes, obteniéndose **31314 imágenes**.

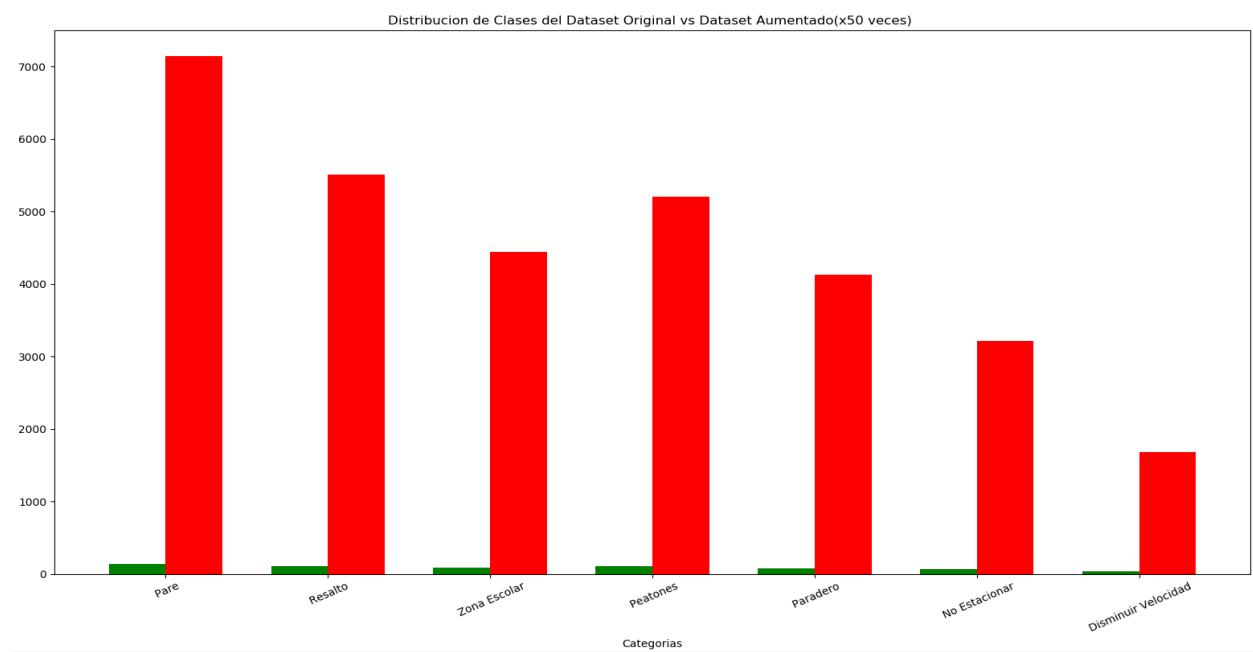


Figura 3.29: Dataset no balanceado - Señales de Tránsito de Perú

Fuente: Elaboración propia

Para este caso, el dataset aumentado se configuró con una distribución de datos para entrenamiento, validación y evaluación:

Tabla 3.2: Distribución Entrenamiento y Validación Dataset no Balanceado - Señales de Tránsito de Perú

<u>CONJUNTO DE DATOS</u>	<u>CANTIDAD IMÁGENES</u>
Entrenamiento	23485 (75 %)
Validación	3131 (10 %)
Evaluación	4698 (15 %)

3.1.5. Pre-procesamiento de Imágenes (Normalization)

Existe una técnica de procesamiento de imágenes por computadora que se utiliza para mejorar el contraste en las imágenes denominada Ecualización Adaptativa del Histograma (AHE por sus siglas en inglés). Difiere de la ecualización de histograma ordinaria en el sentido de que el método adaptativo computa varios histogramas, cada uno correspondiente a una sección distinta de la imagen, y los utiliza para redistribuir los valores de luminosidad de la imagen. Por lo tanto, es adecuado para mejorar el contraste local y mejorar las definiciones de los bordes en cada región de una imagen. Sin embargo, AHE tiene una tendencia a amplificar el ruido en regiones relativamente homogéneas de una imagen. Una variante de la ecualización de histograma adaptativo llamada ecualización de histograma adaptativo limitado por contraste (CLAHE por sus siglas en inglés) evita que se genere esto al limitar la amplificación.

En esta investigación, aplicaremos la técnica CLAHE, un algoritmo para la mejora del contraste local, que utiliza histogramas calculados sobre diferentes regiones en una imagen. Utilizado frecuentemente para mejorar el nivel de visibilidad de una imagen o video con niebla ya que permite

mejorar los detalles locales incluso en regiones que son más oscuras o más claras que la mayoría de regiones de la imagen. Este algoritmo mejorará aún más la extracción de características de cada imagen, (Yadav et al., 2014)



Figura 3.30: Imágenes a las cuales se le aplicaron la técnica CLAHE

Fuente: Elaboración propia

Por otra parte, para convertir un color de un espacio de color basado en un modelo de color RGB típico de gamma comprimido (no lineal) a una representación en escala de grises de su luminancia, primero se debe eliminar la función de compresión gamma mediante la expansión gamma para transformar la imagen en un RGB lineal espacio de color, de modo que la suma ponderada apropiada se puede aplicar a los componentes de color lineales (R_{linear} , G_{linear} , B_{linear}) para calcular la luminancia lineal Y_{linear} .

Para imágenes en espacios de color como Y'UV y sus derivados, que se usan en sistemas de TV y video a color estándar como PAL, SECAM y NTSC, un componente de luma no lineal (Y') se

calcula directamente a partir de intensidades primarias comprimidas con gamma como una suma ponderada, que aunque no es una representación perfecta de la luminancia colorimétrica, puede calcularse más rápidamente sin la expansión gamma y sin la compresión utilizadas en los cálculos fotométricos o colorimétricos,(Poynton, 2003). En los modelos utilizados por PAL y NTSC para conseguir tener las imágenes en escala de grises, el componente de luma no lineal (Y') se calcula como: $Y' = 0,299R' + 0,587G' + 0,114B'$ (Cook, 2009)

Pierre Sermanet y Yann LeCun mencionaron en su artículo (LeCun et al., 1998), que el uso de canales de color no pareció mejorar mucho las cosas. Además, debido a diversas condiciones o problemas de iluminación, no es adecuado procesar directamente las imágenes que se capturan a través de la cámara o sensores de imágenes, es por ello que en esta investigación **se usará un solo canal** en el modelo, es decir las imágenes estarán en escala de grises en lugar de tener 3 canales de colores.

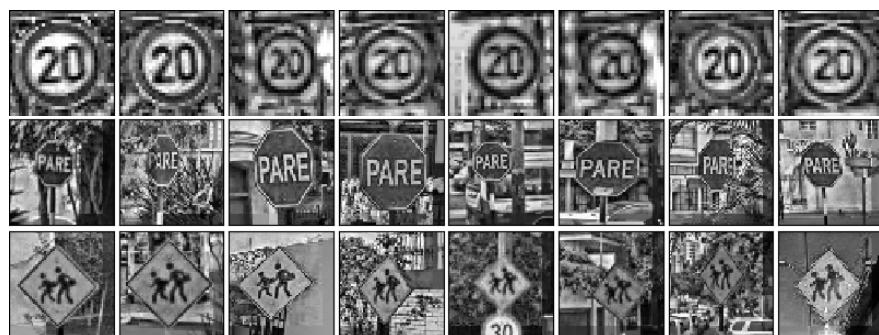


Figura 3.31: Imágenes procesadas en escala de grises

Fuente: Elaboración propia

3.2. Arquitectura del Modelo

Los hiperparámetros engloban funciones, variables y constantes utilizadas durante la construcción de las diferentes arquitecturas; estas varían, sin embargo, siguiendo conceptos teóricos, antecedentes (investigaciones previas) y sobre todo después de algunas pruebas realizadas, los siguientes hiperparámetros fueron seleccionados de manera específica:

Tabla 3.3: Hiperparámetros del Modelo

HIPERPARÁMETROS	TIPO	HIPERPARÁMETROS	TIPO
Inicialización de Pesos	Xavier	Tasa de Aprendizaje	entre 0.0005 y 0.0001
Alg. de Optimización	Optimizador Adam	Método de Validación	Entropía Cruzada
Fun. Activ. Capas Convolucionales	RELU y DropOut	Fun. Activ. Capas Totalmente Conectadas	Func. Softmax
Método de Regularización	$L2\ Lasso(\lambda = 0.0001)$	Épocas	100

Recordemos que en cada época se procesa el dataset completo de imágenes, por lo que cada época es igual: $epoch = \text{Tamaño del Minibatch} \times \text{iteraciones}$

Sin embargo, debido a que se tiene dos distintos datasets para el entrenamiento (**Sección 3.1.4 - Dataset final para el Entrenamiento**), el tamaño del Minibatch cambiará respecto al tamaño del dataset y por ende también será diferente la cantidad de iteraciones ejecutadas durante 100 épocas de entrenamiento.

A continuación, se presentan 2 cuadros con las distribuciones del tamaño del Mini-batch y la cantidad de iteraciones necesarias durante las 100 épocas para ambos tipos de dataset (Alemania y Perú).

Tabla 3.4: Mini-batch e iteraciones en el Dataset de Señales de Tránsito de Alemania Balanceado

Imágenes totales	203175
Iteraciones por Época	387
Tamaño del Mini-batch	525 imágenes analizadas por iteración
Épocas entrenadas	100 épocas (38700 iter.)

Tabla 3.5: Mini-batch e iteraciones en el Dataset de Señales de Tránsito de Perú No Balanceado

Imágenes totales	23485
Iteraciones por Época	77
Tamaño del Mini-batch	305 imágenes analizadas por iteración
Épocas entrenadas	100 épocas (7700 iter.)

En esta investigación se implementó el modelo de redes neuronales convolucionales(CNN) donde las capas convolucionales iniciales de la red extraen características de la imagen (feature extractor), mientras que las capas totalmente conectadas predicen las probabilidades de salida(classifier).

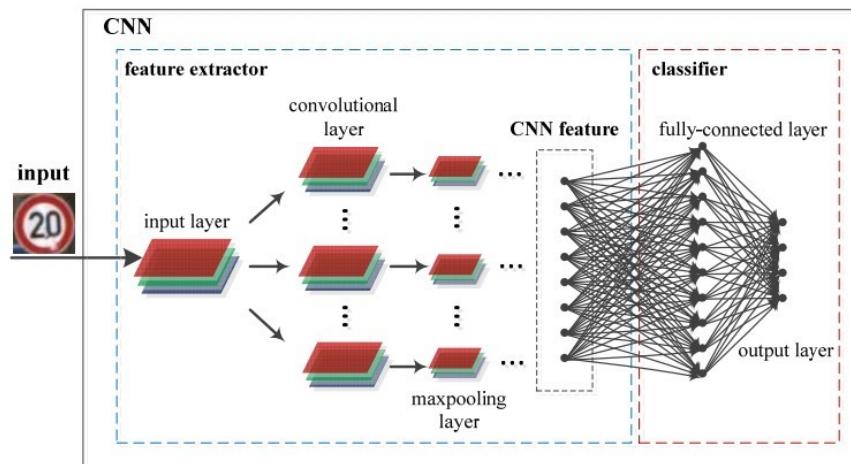


Figura 3.32: Arquitectura de la CNN

Fuente: Zeng et al. (2015)

La arquitectura de la red está inspirada en la arquitectura Inception (Szegedy et al., 2014) y en la arquitectura AlexNet (Krizhevsky et al., 2012) para la clasificación de imágenes. En la arquitectura Inception, el modelo creado es denominado GoogLeNet, similar a AlexNet donde varios módulos iniciales son apilados uno sobre el otro para producir el resultado final. En el módulo de inicio, en ese tipo de red se usaron diversos tamaños de filtros convolucionales para capturar características de diferente abstracción. El alto nivel de abstracción se captura con filtros de mayor tamaño y el de un nivel inferior con filtros de menor tamaño, procesando información visual a diferentes escalas y al concatenarlas se obtiene un nivel eficiente de abstracción.

Para la clasificación de señales de tránsito en esta investigación se utilizó una versión modificada de las arquitecturas antes mencionadas. Se incorporó **funciones de escala-múltiple** (Sermanet and LeCun, 2011), lo que significa que la salida de las capas convolucionales no solo se envía a la capa posterior, sino que también se ramifica y se introduce al clasificador (capa totalmente conectada). La razón detrás de esto es que cuando el clasificador está tomando una decisión basada en convoluciones, podría encontrar que la salida de la primera o segunda capa convolucional también es útil. Básicamente con las características de escala múltiple depende del clasificador qué nivel de abstracción usar, ya que tiene acceso a las salidas de todas las capas convolucionales, es decir, características en todos los niveles de abstracción. Estas capas ramificadas se someten a un pooling máximo adicional, de modo que todas las convoluciones se submuestrean proporcionalmente antes de entrar en el clasificador. Así se garantiza que todas las funciones de escala múltiple

experimenten la misma cantidad de máximo aprovechamiento.

3.2.1. Diseño de la Red

Teniendo en cuenta lo descrito en la **sección 2.2 (Figura 2.6)**, para describir las capas convolucionales se utilizará la terminología compleja (cada capa tiene múltiples etapas).

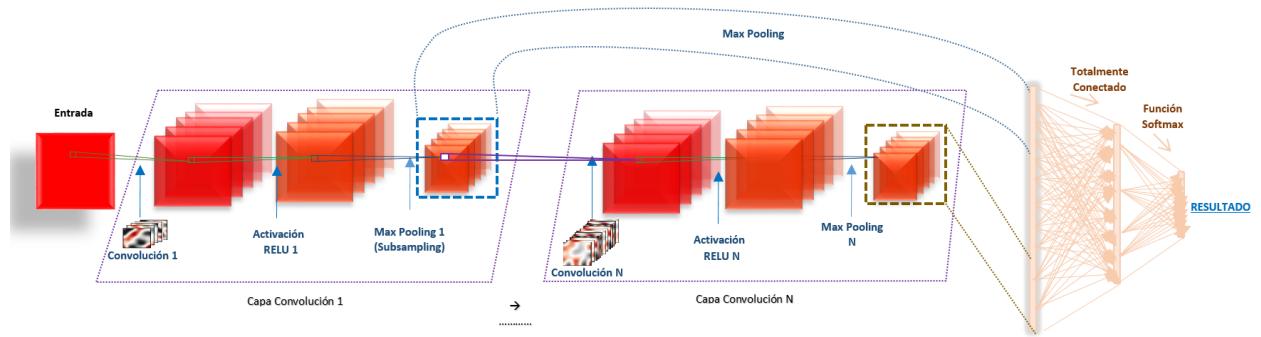


Figura 3.33: Modelo del diseño de la Red propuesta

Fuente: Elaboración propia

Para el proceso de entrenamiento fueron tomados en cuenta 5 diferentes diseños. Estos fueron ejecutados durante el entrenamiento del dataset que contiene señales de tránsito de Alemania y Perú.

Estos 5 modelos son descritos a continuación.

3.2.1.1. Diseño A

Diseño compuesto de 2 capas convolucionales y 2 capas totalmente conectadas.

- Dataset de Alemania

Tabla 3.6: Diseño A (73995 neuronas) - Dataset de Imágenes de Alemania

Capa	Entrada	Tipo	Número de (kernels/filtros)	Padding	Salida	Func.Esc.Múltiple
1	1 de 32 x 32 neuronas	Conv(DropOut : 0.8)	32 de 3 x 3	Activo	32 de 32 x 32 neuronas	–
	32 de 32 x 32 neuronas	Max Pool	32 de 2 x 2	Inactivo	32 de 16 x 16 neuronas	(Kernel = 2) 32 de 8x8
2	32 de 16 x 16 neuronas	Conv(DropOut : 0.7)	64 de 5 x 5	Activo	64 de 16 x 16 neuronas	–
	64 de 16 x 16 neuronas	Max Pool	64 de 2 x 2	Inactivo	64 de 8 x 8 neuronas	(Kernel = 1) 64 de 8x8
3	6144 neuronas	F.C.(DropOut : 0.5)	3072 neuronas	–	43 neuronas	–

- Dataset del Perú

Tabla 3.7: Diseño A (250679 neuronas) - Dataset de Imágenes de Perú

Capa	Entrada	Tipo	Número de (kernels/filtros)	Padding	Salida	Func.Esc.Múltiple
1	1 de 60 x 60 neuronas	Conv(DropOut : 0.8)	32 de 3 x 3	Activo	32 de 60 x 60 neuronas	–
	32 de 60 x 60 neuronas	Max Pool	32 de 2 x 2	Inactivo	32 de 30 x 30 neuronas	(Kernel = 2) 32 de 15x15
2	32 de 30 x 30 neuronas	Conv(DropOut : 0.7)	64 de 5 x 5	Activo	64 de 30 x 30 neuronas	–
	64 de 30 x 30 neuronas	Max Pool	64 de 2 x 2	Inactivo	64 de 15 x 15 neuronas	(Kernel = 1) 64 de 15x15
3	21600 neuronas	F.C.(DropOut : 0.5)	7200 neuronas	–	7 neuronas	–

3.2.1.2. Diseño B

Diseño compuesto de 3 capas convolucionales y 2 capas totalmente conectadas.

- Dataset de Alemania

Tabla 3.8: Diseño B (83339 neuronas) - Dataset de Imágenes de Alemania

Capa	Entrada	Tipo	Número de (kernels/filtros)	Padding	Salida	Func.Esc.Múltiple
1	1 de 32 x 32 neuronas	Conv(DropOut : 0.8)	32 de 3 x 3	Activo	32 de 32 x 32 neuronas	–
	32 de 32 x 32 neuronas	Max Pool	32 de 2 x 2	Inactivo	32 de 16 x 16 neuronas	(Kernel = 4) 32 de 4x4
2	32 de 16 x 16 neuronas	Conv(DropOut : 0.7)	64 de 5 x 5	Activo	64 de 16 x 16 neuronas	–
	64 de 16 x 16 neuronas	Max Pool	64 de 2 x 2	Inactivo	64 de 8 x 8 neuronas	(Kernel = 2) 64 de 4x4
3	64 de 8 x 8 neuronas	Conv(DropOut : 0.6)	128 de 5 x 5	Activo	128 de 8 x 8 neuronas	–
	128 de 8 x 8 neuronas	Max Pool	128 de 2 x 2	Inactivo	128 de 4 x 4 neuronas	(Kernel = 1) 128 de 4x4
4	3584 neuronas	F.C.(DropOut : 0.5)	1024 neuronas	–	43 neuronas	–

- Dataset del Perú

Tabla 3.9: Diseño B (274339 neuronas) - Dataset de Imágenes de Perú

Capa	Entrada	Tipo	Número de (kernels/filtros)	Padding	Salida	Func.Esc.Múltiple
1	1 de 60 x 60 neuronas	Conv(DropOut : 0.8)	32 de 3 x 3	Activo	32 de 60 x 60 neuronas	–
	32 de 60 x 60 neuronas	Max Pool	32 de 2 x 2	Inactivo	32 de 30 x 30 neuronas	(Kernel = 4) 32 de 7x7
2	32 de 30 x 30 neuronas	Conv(DropOut : 0.7)	64 de 5 x 5	Activo	64 de 30 x 30 neuronas	–
	64 de 30 x 30 neuronas	Max Pool	64 de 2 x 2	Inactivo	64 de 15 x 15 neuronas	(Kernel = 2) 64 de 7x7
3	64 de 15 x 15 neuronas	Conv(DropOut : 0.6)	128 de 5 x 5	Activo	128 de 15 x 15 neuronas	–
	128 de 15 x 15 neuronas	Max Pool	128 de 2 x 2	Inactivo	128 de 7 x 7 neuronas	(Kernel = 1) 128 de 7x7
4	10976 neuronas	F.C.(DropOut : 0.5)	2700 neuronas	–	7 neuronas	–

3.2.1.3. Diseño C

Diseño compuesto de 3 capas convolucionales y 2 capas totalmente conectadas. Variando el número de kernels en la 2da capa convolucional.

- Dataset de Alemania

Tabla 3.10: Diseño C (82315 neuronas) - Dataset de Imágenes de Alemania

Capa	Entrada	Tipo	Número de (kernels/filtros)	Padding	Salida	Func.Esc.Múltiple
1	1 de 32 x 32 neuronas	Conv(DropOut : 0.8)	32 de 3 x 3	Activo	32 de 32 x 32 neuronas	–
	32 de 32 x 32 neuronas	Max Pool	32 de 2 x 2	Inactivo	32 de 16 x 16 neuronas	(Kernel = 4) 32 de 4x4
2	32 de 16 x 16 neuronas	Conv(DropOut : 0.7)	64 de 3 x 3	Activo	64 de 16 x 16 neuronas	–
	64 de 16 x 16 neuronas	Max Pool	64 de 2 x 2	Inactivo	64 de 8 x 8 neuronas	(Kernel = 2) 64 de 4x4
3	64 de 8 x 8 neuronas	Conv(DropOut : 0.6)	128 de 5 x 5	Activo	128 de 8 x 8 neuronas	–
	128 de 8 x 8 neuronas	Max Pool	128 de 2 x 2	Inactivo	128 de 4 x 4 neuronas	(Kernel = 1) 128 de 4x4
4	3584 neuronas	F.C.(DropOut : 0.5)	1024 neuronas	–	43 neuronas	–

- Dataset del Perú

Tabla 3.11: Diseño C (273315 neuronas) - Dataset de Imágenes de Perú

Capa	Entrada	Tipo	Número de (kernels/filtros)	Padding	Salida	Func.Esc.Múltiple
1	1 de 60 x 60 neuronas	Conv(DropOut : 0.8)	32 de 3 x 3	Activo	32 de 60 x 60 neuronas	–
	32 de 60 x 60 neuronas	Max Pool	32 de 2 x 2	Inactivo	32 de 30 x 30 neuronas	(Kernel = 4) 32 de 7x7
2	32 de 30 x 30 neuronas	Conv(DropOut : 0.7)	64 de 3 x 3	Activo	64 de 30 x 30 neuronas	–
	64 de 30 x 30 neuronas	Max Pool	64 de 2 x 2	Inactivo	64 de 15 x 15 neuronas	(Kernel = 2) 64 de 7x7
3	64 de 15 x 15 neuronas	Conv(DropOut : 0.6)	128 de 5 x 5	Activo	128 de 15 x 15 neuronas	–
	128 de 15 x 15 neuronas	Max Pool	128 de 2 x 2	Inactivo	128 de 7 x 7 neuronas	(Kernel = 1) 128 de 7x7
4	10976 neuronas	F.C.(DropOut : 0.5)	2700 neuronas	–	7 neuronas	–

3.2.1.4. Diseño D

Diseño compuesto de 3 capas convolucionales y 2 capas totalmente conectadas. Variando el número de kernels en la 3ra capa convolucional.

- Dataset de Alemania

Tabla 3.12: Diseño D (86411 neuronas) - Dataset de Imágenes de Alemania

Capa	Entrada	Tipo	Número de (kernels/filtros)	Padding	Salida	Func.Esc.Múltiple
1	1 de 32 x 32 neuronas	Conv(DropOut : 0.8)	32 de 3 x 3	Activo	32 de 32 x 32 neuronas	–
	32 de 32 x 32 neuronas	Max Pool	32 de 2 x 2	Inactivo	32 de 16 x 16 neuronas	(Kernel = 4) 32 de 4x4
2	32 de 16 x 16 neuronas	Conv(DropOut : 0.7)	64 de 5 x 5	Activo	64 de 16 x 16 neuronas	–
	64 de 16 x 16 neuronas	Max Pool	64 de 2 x 2	Inactivo	64 de 8 x 8 neuronas	(Kernel = 2) 64 de 4x4
3	64 de 8 x 8 neuronas	Conv(DropOut : 0.6)	128 de 7 x 7	Activo	128 de 8 x 8 neuronas	–
	128 de 8 x 8 neuronas	Max Pool	128 de 2 x 2	Inactivo	128 de 4 x 4 neuronas	(Kernel = 1) 128 de 4x4
4	3584 neuronas	F.C.(DropOut : 0.5)	1024 neuronas	–	43 neuronas	–

- Dataset del Perú

Tabla 3.13: Diseño D (277411 neuronas) - Dataset de Imágenes de Perú

Capa	Entrada	Tipo	Número de (kernels/filtros)	Padding	Salida	Func.Esc.Múltiple
1	1 de 60 x 60 neuronas	Conv(DropOut : 0.8)	32 de 3 x 3	Activo	32 de 60 x 60 neuronas	–
	32 de 60 x 60 neuronas	Max Pool	32 de 2 x 2	Inactivo	32 de 30 x 30 neuronas	(Kernel = 4) 32 de 7x7
2	32 de 30 x 30 neuronas	Conv(DropOut : 0.7)	64 de 5 x 5	Activo	64 de 30 x 30 neuronas	–
	64 de 30 x 30 neuronas	Max Pool	64 de 2 x 2	Inactivo	64 de 15 x 15 neuronas	(Kernel = 2) 64 de 7x7
3	64 de 15 x 15 neuronas	Conv(DropOut : 0.6)	128 de 7 x 7	Activo	128 de 15 x 15 neuronas	–
	128 de 15 x 15 neuronas	Max Pool	128 de 2 x 2	Inactivo	128 de 7 x 7 neuronas	(Kernel = 1) 128 de 7x7
4	10976 neuronas	F.C.(DropOut : 0.5)	2700 neuronas	–	7 neuronas	–

3.2.1.5. Diseño E

Diseño compuesto de 4 capas convolucionales y 2 capas totalmente conectadas.

- Dataset de Alemania

Tabla 3.14: Diseño E (90185 neuronas) - Dataset de Imágenes de Alemania

Capa	Entrada	Tipo	Número de (kernels/filtros)	Padding	Salida	Func.Esc.Múltiple
1	1 de 32 x 32 neuronas	Conv(DropOut : 0.8)	32 de 3 x 3	Activo	32 de 32 x 32 neuronas	–
	32 de 32 x 32 neuronas	Max Pool	32 de 2 x 2	Inactivo	32 de 16 x 16 neuronas	(Kernel = 8) 32 de 2x2
2	32 de 16 x 16 neuronas	Conv(DropOut : 0.7)	64 de 5 x 5	Activo	64 de 16 x 16 neuronas	–
	64 de 16 x 16 neuronas	Max Pool	64 de 2 x 2	Inactivo	64 de 8 x 8 neuronas	(Kernel = 4) 64 de 2x2
3	64 de 8 x 8 neuronas	Conv(DropOut : 0.6)	128 de 5 x 5	Activo	128 de 8 x 8 neuronas	–
	128 de 8 x 8 neuronas	Max Pool	128 de 2 x 2	Inactivo	128 de 4 x 4 neuronas	(Kernel = 2) 128 de 2x2
4	128 de 4 x 4 neuronas	Conv(DropOut : 0.6)	128 de 7 x 7	Activo	128 de 4 x 4 neuronas	–
	128 de 4 x 4 neuronas	Max Pool	128 de 2 x 2	Inactivo	128 de 2 x 2 neuronas	(Kernel = 1) 128 de 2x2
5	1408 neuronas	F.C.(DropOut : 0.5)	702 neuronas	–	43 neuronas	–

- Dataset del Perú

Tabla 3.15: Diseño E (279623 neuronas) - Dataset de Imágenes de Perú

Capa	Entrada	Tipo	Número de (kernels/filtros)	Padding	Salida	Func.Esc.Múltiple
1	1 de 60 x 60 neuronas	Conv(DropOut : 0.8)	32 de 3 x 3	Activo	32 de 60 x 60 neuronas	–
	32 de 60 x 60 neuronas	Max Pool	32 de 2 x 2	Inactivo	32 de 30 x 30 neuronas	(Kernel = 8) 32 de 3x3
2	32 de 30 x 30 neuronas	Conv(DropOut : 0.7)	64 de 5 x 5	Activo	64 de 30 x 30 neuronas	–
	64 de 30 x 30 neuronas	Max Pool	64 de 2 x 2	Inactivo	64 de 15 x 15 neuronas	(Kernel = 4) 64 de 3x3
3	64 de 15 x 15 neuronas	Conv(DropOut : 0.6)	128 de 5 x 5	Activo	128 de 15 x 15 neuronas	–
	128 de 15 x 15 neuronas	Max Pool	128 de 2 x 2	Inactivo	128 de 7 x 7 neuronas	(Kernel = 2) 128 de 3x3
4	128 de 7 x 7 neuronas	Conv(DropOut : 0.6)	128 de 7 x 7	Activo	128 de 7 x 7 neuronas	–
	128 de 7 x 7 neuronas	Max Pool	128 de 2 x 2	Inactivo	128 de 3 x 3 neuronas	(Kernel = 1) 128 de 3x3
5	3168 neuronas	F.C.(DropOut : 0.5)	1584 neuronas	–	7 neuronas	–

En el apéndice A se muestran bloques de pseudocódigo que fueron implementados para la construcción de los diseños de Red.

3.3. Entrenamiento y Validación

Para estimar el error de clasificación, como fue mencionado en la **Sección 2.3**, se utilizó el método de validación cruzada. En la estimación del error se usa el conjunto de muestras disponible, el cual se divide en el conjunto de entrenamiento y el de validación. El clasificador se diseña usando las muestras de entrenamiento y luego se evalúa obteniendo el error de clasificación para las muestras de validación. Con base en el error obtenido se puede predecir el desempeño del clasificador ante nuevas muestras. Para obtener una medida confiable del desempeño, el conjunto de muestras es lo suficientemente grande y los conjuntos de entrenamiento y de validación son independientes.

3.3.1. Señales de Tránsito de Alemania

Luego de haber entrenado las diferentes arquitecturas durante 100 épocas (38700 iteraciones), obtenemos los siguientes resultados:

3.3.1.1. Análisis del Entrenamiento y Validación del Diseño A

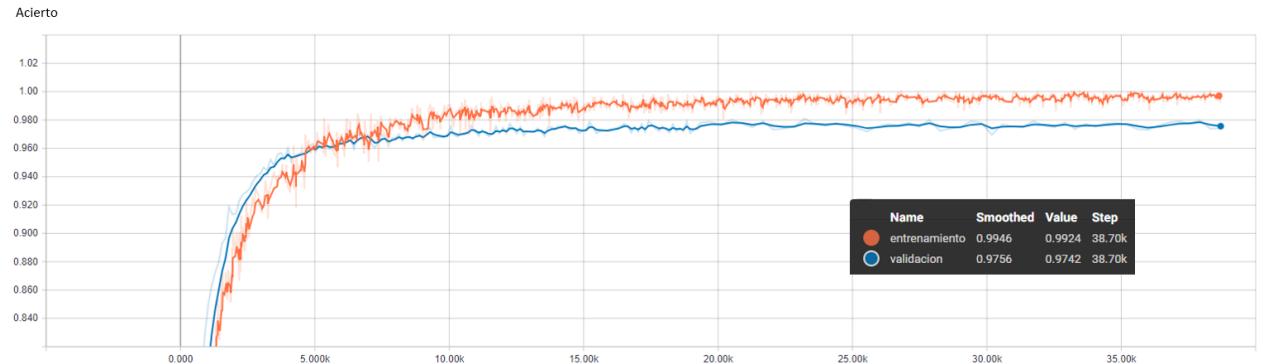


Figura 3.34: Modelo A Tasa de Acierto por iteración - Dataset de imágenes de Alemania

Fuente: Elaboración propia

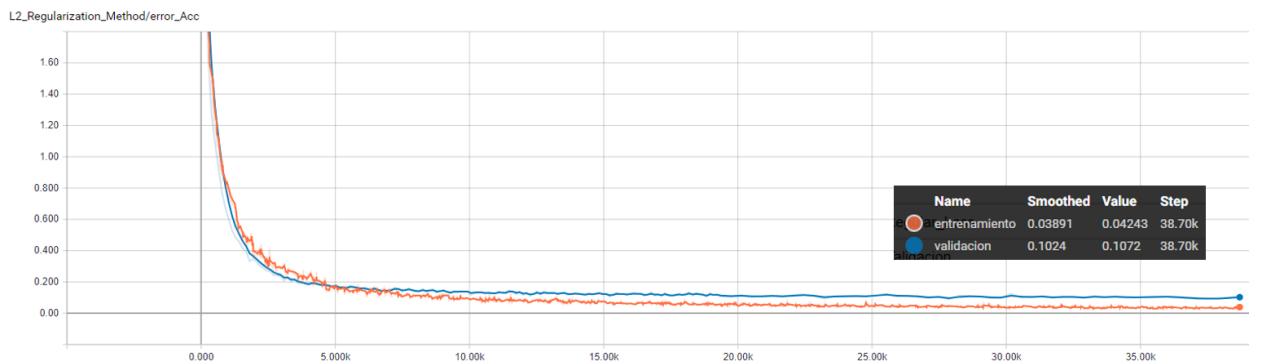


Figura 3.35: Modelo A Tasa de pérdida por iteración - Dataset de imágenes de Alemania

Fuente: Elaboración propia

Luego de 20 épocas iteradas (aproximadamente 7000 iteraciones), se observa que rápidamente el modelo se sobre-ajusta al dataset de entrenamiento. Es decir, rápidamente el valor de pérdida es muy cercano a cero, lo que a su vez hace que se separe de manera abrupta de los resultados de validación y lo mismo sucede con la tasa de acierto, en la cual la de validación luego de estas 20 épocas es mucho menor al de entrenamiento. Iteraciones posteriores no permitirán mejores resultados.

3.3.1.2. Análisis del Entrenamiento y Validación del Diseño B

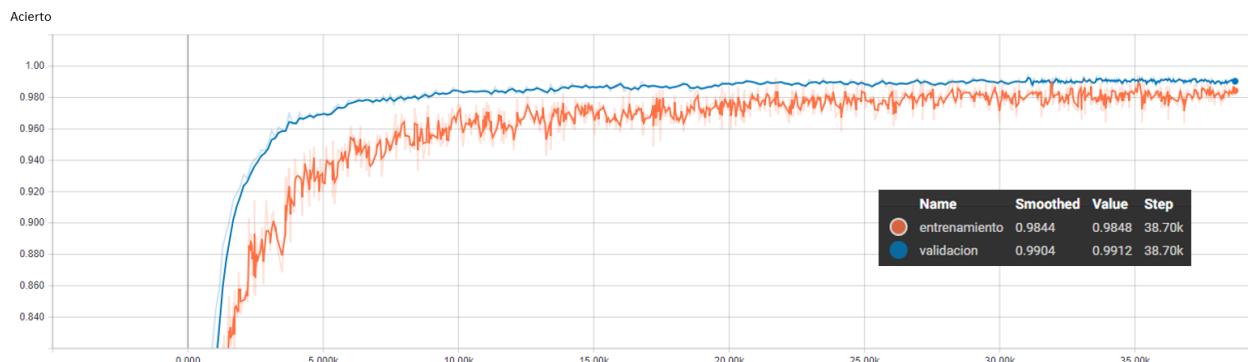


Figura 3.36: Modelo B Tasa de Acierto por iteración - Dataset de imágenes de Alemania

Fuente: Elaboración propia

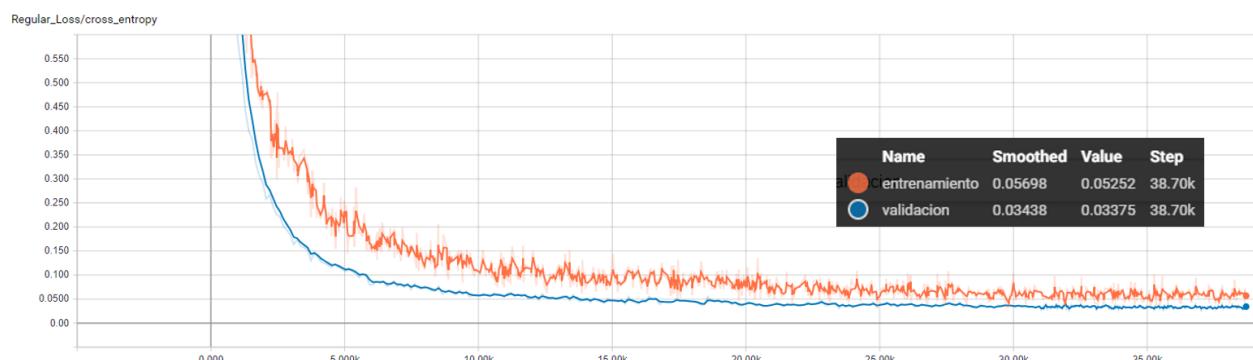


Figura 3.37: Modelo B Tasa de pérdida por iteración - Dataset de imágenes de Alemania

Fuente: Elaboración propia

Durante 100 épocas iteradas (38700 iteraciones), a partir aproximadamente de **34800 iteraciones (80 épocas)** se obtiene los mejores promedios de acierto y error, logrando resultados muy similares entre la tasa de acierto y pérdida del entrenamiento y validación.

3.3.1.3. Análisis del Entrenamiento y Validación del Diseño C

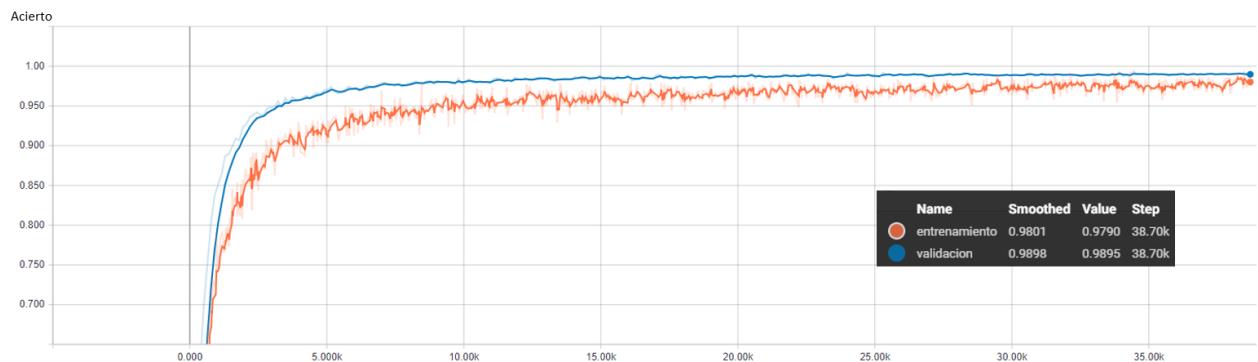


Figura 3.38: Modelo C Tasa de Acierto por iteración - Dataset de imágenes de Alemania

Fuente: Elaboración propia

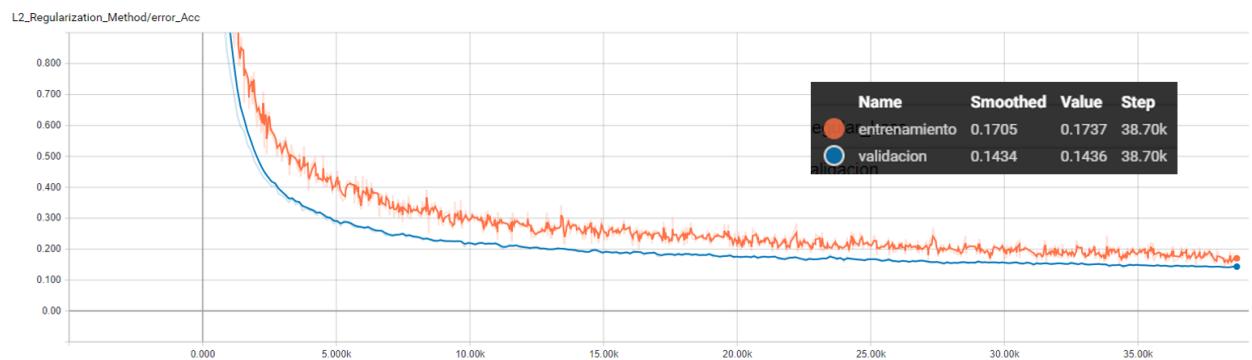


Figura 3.39: Modelo C Tasa de pérdida por iteración - Dataset de imágenes de Alemania

Fuente: Elaboración propia

Al igual que el modelo anterior, luego de 100 épocas iteradas (38700 iteraciones), se obtiene las mejores tasas de acierto y pérdida en la validación logrando resultados muy similares entre el entrenamiento y la validación, se podría seguir entrenando en iteraciones posteriores, es muy probable que el modelo obtenga mejoras en los resultados pero a su vez, haría que se sobreajuste al conjunto de entrenamiento y no permita evaluar correctamente imágenes previamente no analizadas.

3.3.1.4. Análisis del Entrenamiento y Validación del Diseño D

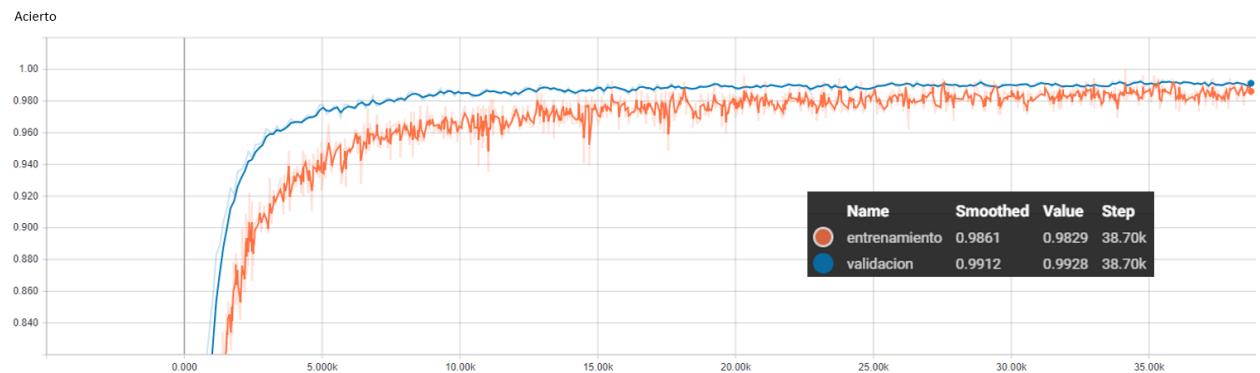


Figura 3.40: Modelo D Tasa de Acierto por iteración - Dataset de imágenes de Alemania

Fuente: Elaboración propia

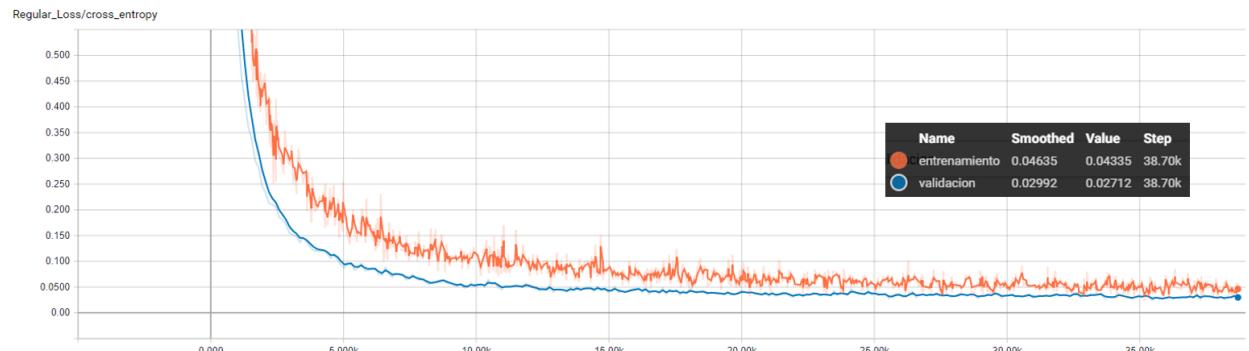


Figura 3.41: Modelo D Tasa de pérdida por iteración - Dataset de imágenes de Alemania

Fuente: Elaboración propia

Este modelo presenta características muy similares a los anteriores, sin embargo, a diferencia del Modelo C y muy similar al modelo B, se observa que a la mitad de épocas entrenadas (20000 iteraciones) el promedio de acierto y pérdida no se aumenta/reduce significativamente lo que resulta en un modelo que demora más en aprender las características importantes durante el entrenamiento.

3.3.1.5. Análisis del Entrenamiento y Validación del Diseño E

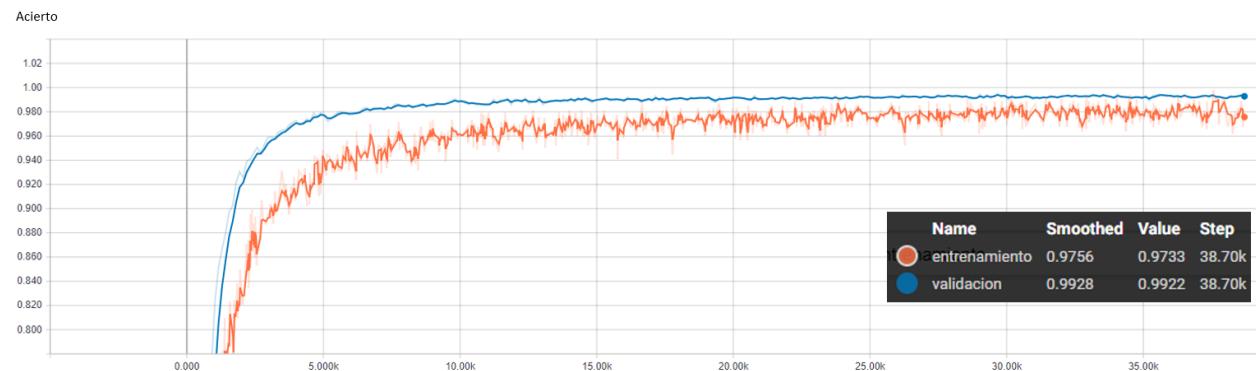


Figura 3.42: Modelo E Tasa de Acierto por iteración - Dataset de imágenes de Alemania

Fuente: Elaboración propia

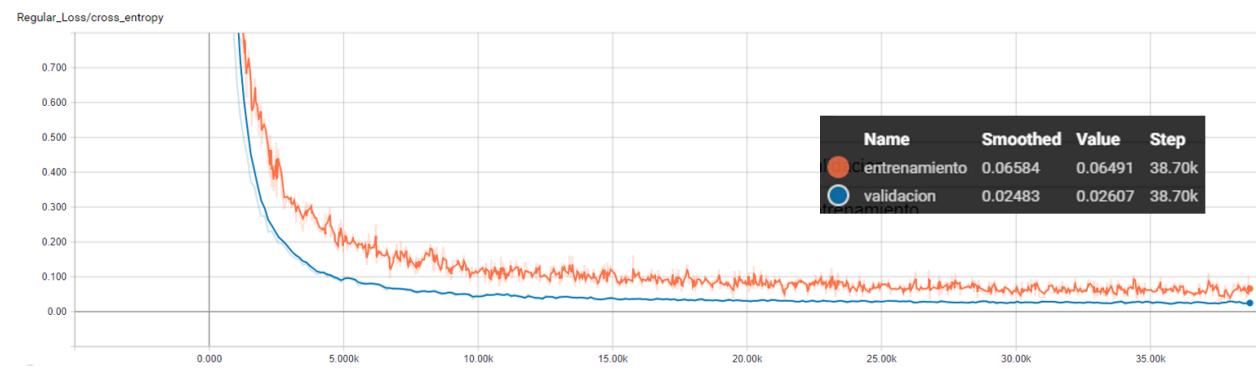


Figura 3.43: Modelo E Tasa de pérdida por iteración - Dataset de imágenes de Alemania

Fuente: Elaboración propia

Poco antes de finalizar las 100 épocas (38700 iteraciones) se obtiene una de las mejores tasas de acierto de validación y una de las tasas más mínimas de pérdida (error).

En resumen, los 5 modelos lograron similares resultados de acierto y error durante el entrenamiento, siendo el modelo A el que obtuvo los mayores resultados observados en la tasa de acierto y error y el modelo E el peor de estos. Sin embargo, esto no indicaría que el modelo A sea el seleccionado como mejor modelo ya que por lo antes mencionado se observa que el modelo A se sobre-ajusta rápidamente al conjunto de entrenamiento siendo la razón principal de la obtención de estos resultados.

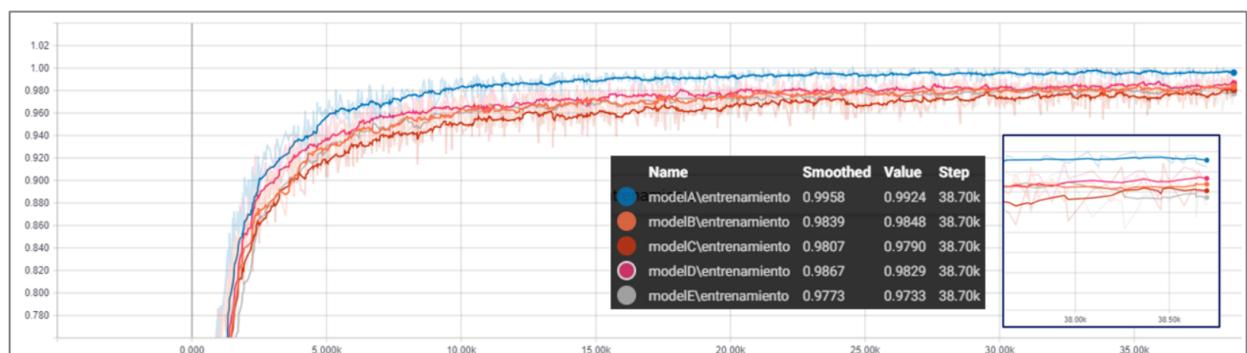


Figura 3.44: Análisis del Acierto durante el Entrenamiento de los modelos - Dataset Señales de Tránsito de Alemania

Fuente: Elaboración propia

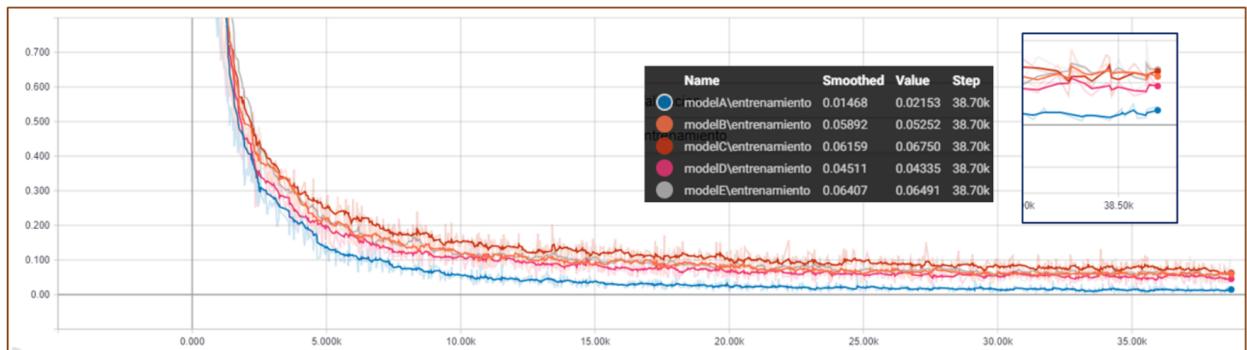


Figura 3.45: Análisis del Error durante el Entrenamiento de los modelos - Dataset Señales de Tránsito de Alemania

Fuente: Elaboración propia

Es por ello que a través de la validación del modelo (usando dataset de imágenes no visto/analizado durante el entrenamiento), se puede apreciar que el Modelo E a pesar que obtuvo las más bajas tasas de acierto y error durante el entrenamiento, es el que mejor generaliza el reconocimiento al presentar las mejores tasas de Acierto y Error ante el conjunto de validación. Esto será corroborado por segunda vez cuando se realice la evaluación del modelo con el dataset de evaluación, el cual presenta mayor cantidad de imágenes que el dataset de validación.

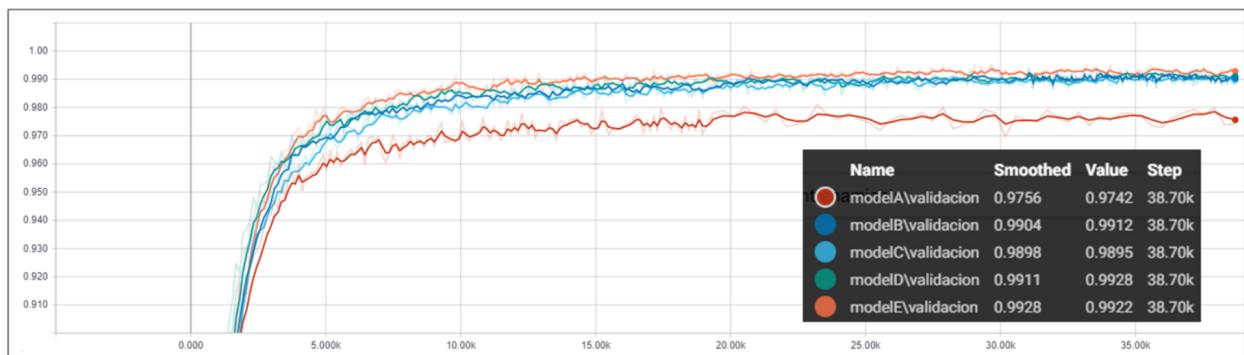


Figura 3.46: Análisis del Acierto en la Validación de los modelos - Dataset Señales de Tránsito de Alemania

Fuente: Elaboración propia

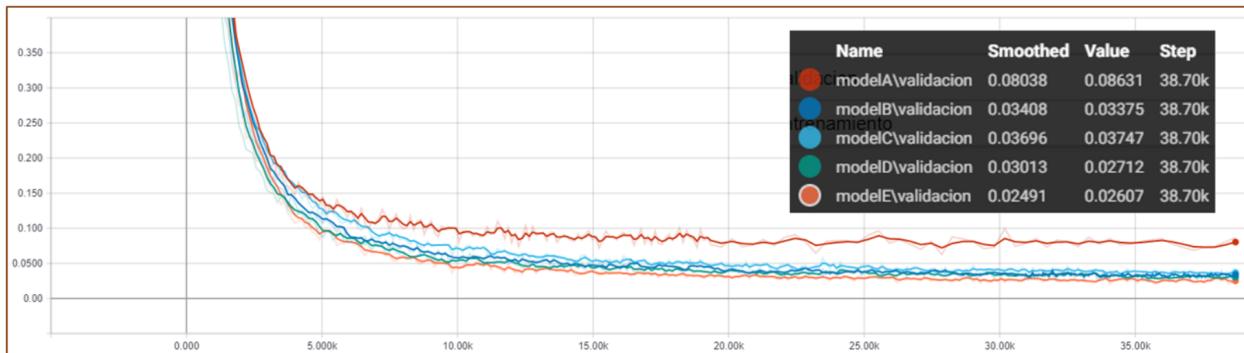


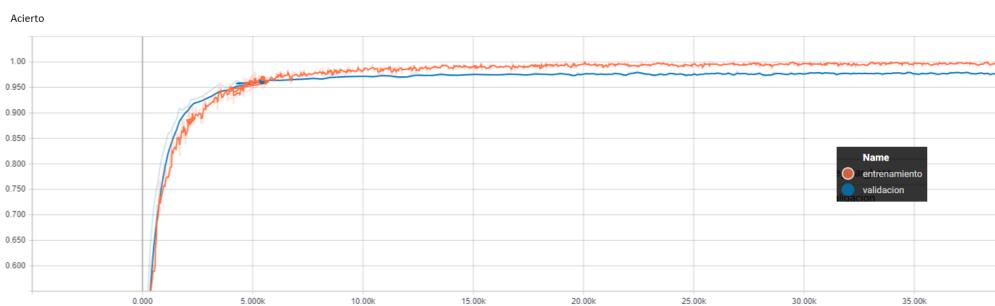
Figura 3.47: Análisis del Error en la Validación de los modelos - Dataset Señales de Tránsito de Alemania

Fuente: Elaboración propia

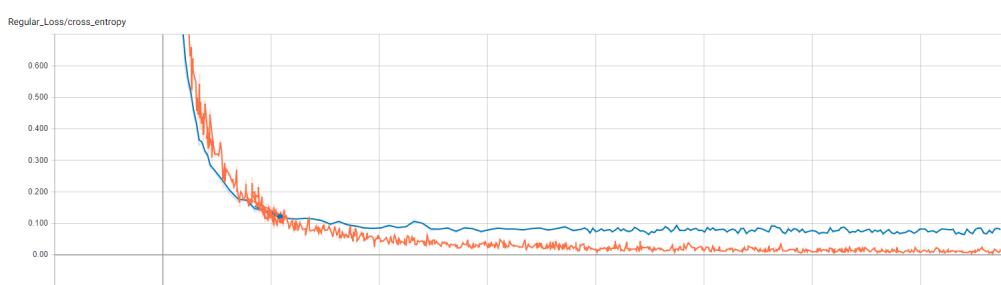
3.3.2. Señales de Tránsito de Perú

Las cinco arquitecturas fueron entrenadas durante 100 épocas (7700 iteraciones) y se obtuvo los siguientes resultados:

3.3.2.1. Análisis del Entrenamiento y Validación del Diseño A



Fuente: Elaboración propia



Fuente: Elaboración propia

Luego de 3850 iteraciones (50 épocas), el modelo se sobre-ajusta al dataset de entrenamiento, rápidamente el valor de pérdida es muy cercano a cero, y hace que se separe de manera abrupta de los resultados en la validación. De esta manera, el modelo rápidamente memoriza las señales con las cuales se está entrenando lo que no permitirá evaluar con exactitud señales nunca antes vistas.

3.3.2.2. Análisis del Entrenamiento y Validación del Diseño B

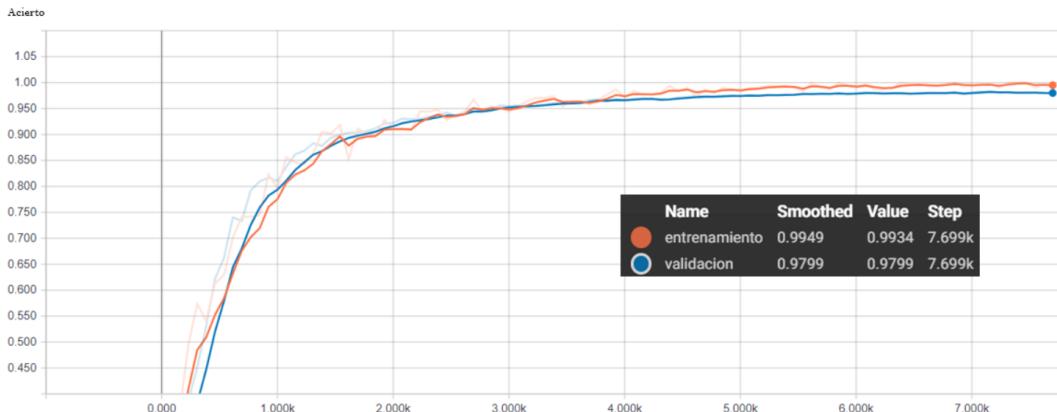


Figura 3.50: Modelo B Tasa de Acierto por iteración - Dataset de imágenes de Perú

Fuente: Elaboración propia

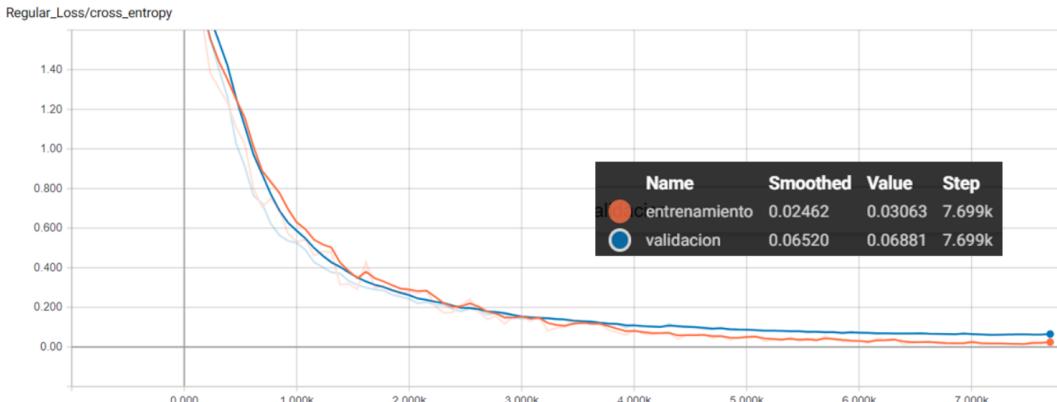


Figura 3.51: Modelo B Tasa de pérdida por iteración

Fuente: Elaboración propia

En este modelo, a partir de 4000 iteraciones (50 épocas) se percibe que el modelo presenta dificultades para seguir extrayendo características que permitan un mejor entrenamiento, por lo que la tasa de Error no es disminuida en iteraciones posteriores, haciendo que el modelo se sobreajuste al conjunto de entrenamiento y no permita evaluar correctamente imágenes previamente no analizadas.

3.3.2.3. Análisis del Entrenamiento y Validación del Diseño C

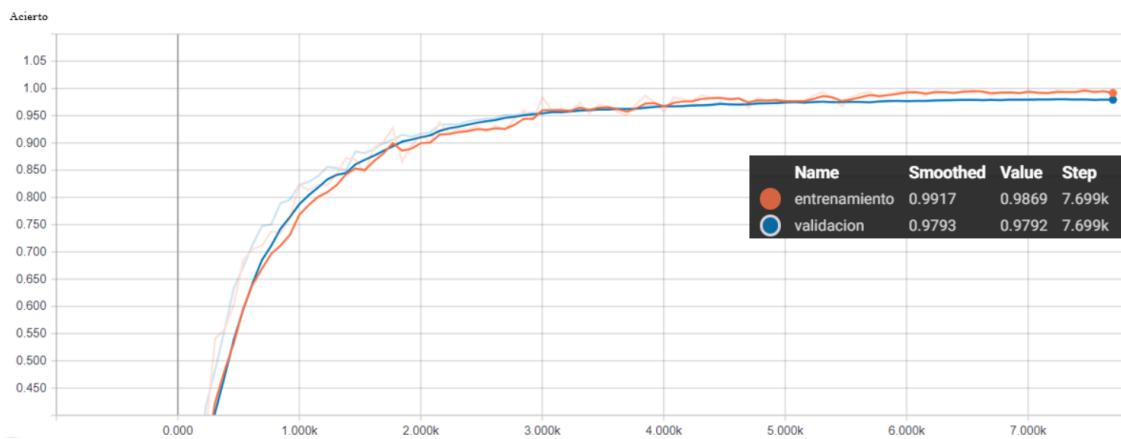


Figura 3.52: Modelo C Tasa de Acierto por iteración - Dataset de imágenes de Perú

Fuente: Elaboración propia

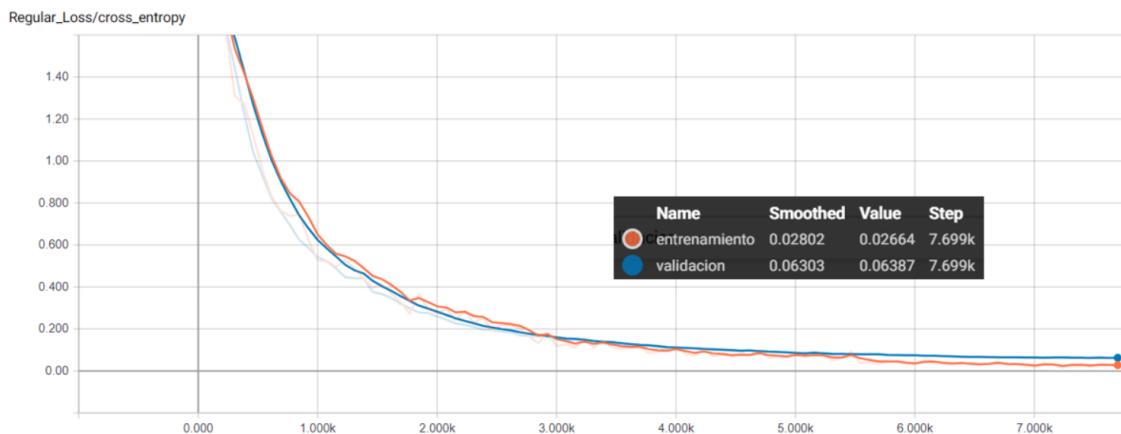


Figura 3.53: Modelo C Tasa de pérdida por iteración

Fuente: Elaboración propia

Al finalizar las 100 épocas iteradas, se obtienen buenas tasas de acierto y error de validación.

Sin embargo, a partir de 70 épocas entrenadas (5500 iteraciones), se observa que las tasas de acierto y error presentadas en el entrenamiento comienzan a diferir de las tasas de validación, por lo que se observa una tendencia a no mejorar estos resultados en iteraciones posteriores.

3.3.2.4. Análisis del Entrenamiento y Validación del Diseño D

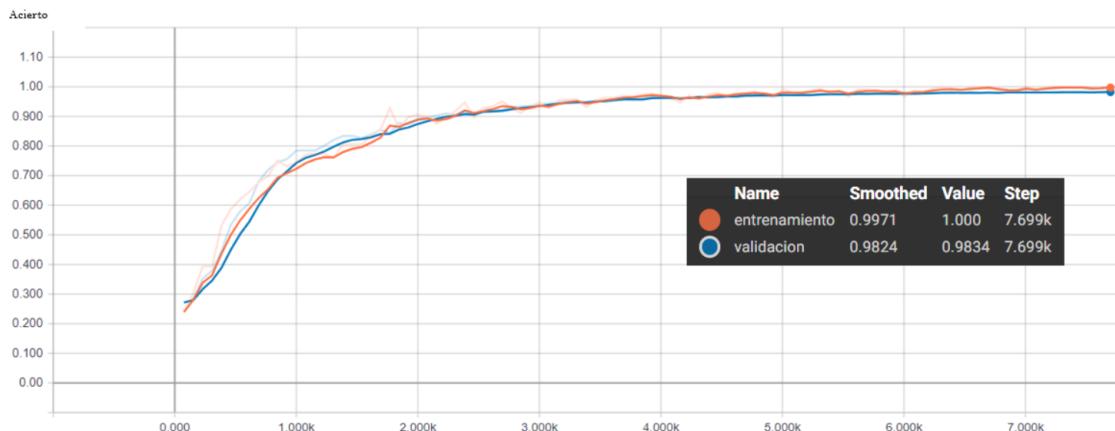


Figura 3.54: Modelo D Tasa de Acierto por iteración - Dataset de imágenes de Perú

Fuente: Elaboración propia

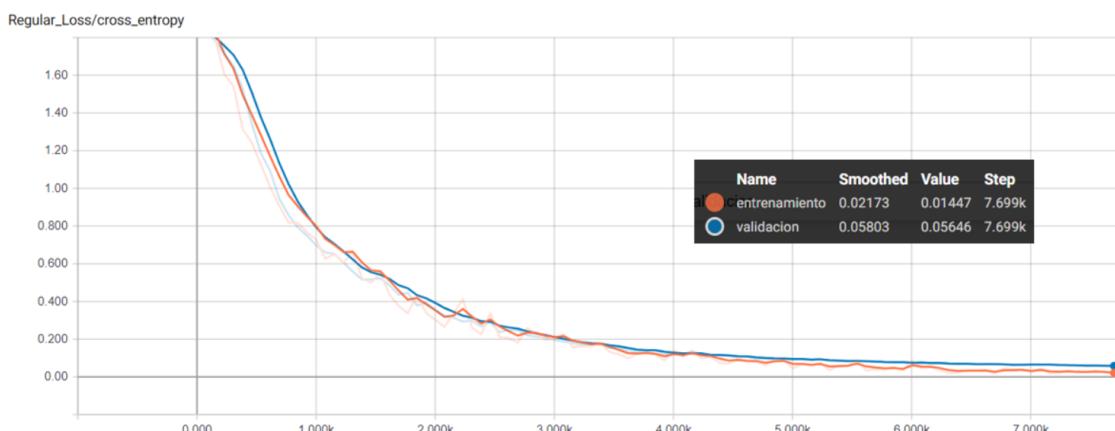


Figura 3.55: Modelo D Tasa de pérdida por iteración

Fuente: Elaboración propia

Este modelo presenta características muy similares al anterior e incluso se observa que al finalizar las 100 épocas, la tasa de acierto del entrenamiento es prácticamente perfecta generando una pequeña mejora en los resultados de acierto y error en el conjunto de validación respecto a los modelos anteriores.

3.3.2.5. Análisis del Entrenamiento y Validación del Diseño E

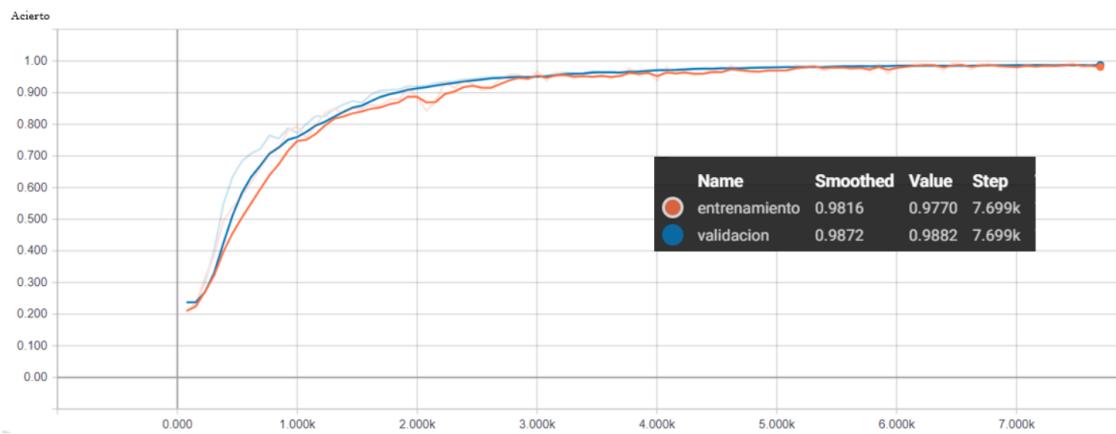


Figura 3.56: Modelo E Tasa de Acierto por iteración - Dataset de imágenes de Perú

Fuente: Elaboración propia

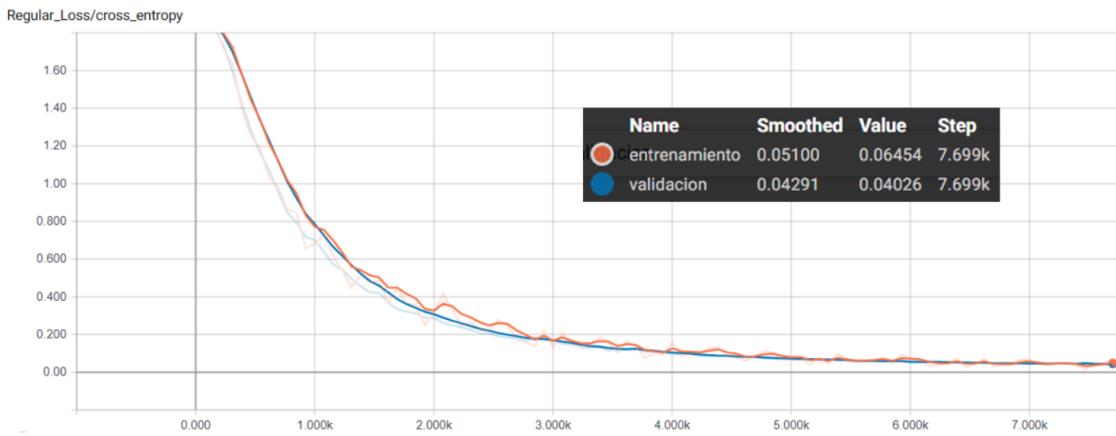


Figura 3.57: Modelo E Tasa de pérdida por iteración

Fuente: Elaboración propia

Al finalizar 100 épocas (7700 iteraciones) se obtiene las mejores tasas de acierto y error en validación. Iteraciones posteriores harán probablemente que el modelo se sobreajuste al conjunto de entrenamiento y no permita evaluar correctamente imágenes previamente no analizadas.

En resumen, los 5 modelos lograron similares resultados de acierto y error durante el entrenamiento y al igual que durante el entrenamiento del dataset de Alemania, el modelo A es el que presenta mayor tasa de acierto y menor tasa de pérdida, sin embargo, como fue antes mencionado, esto no garantiza que sea el modelo que mejor generalice el reconocimiento de señales por lo que se tendrá que analizar los mismos resultados revisando los valores durante la validación de los modelos.

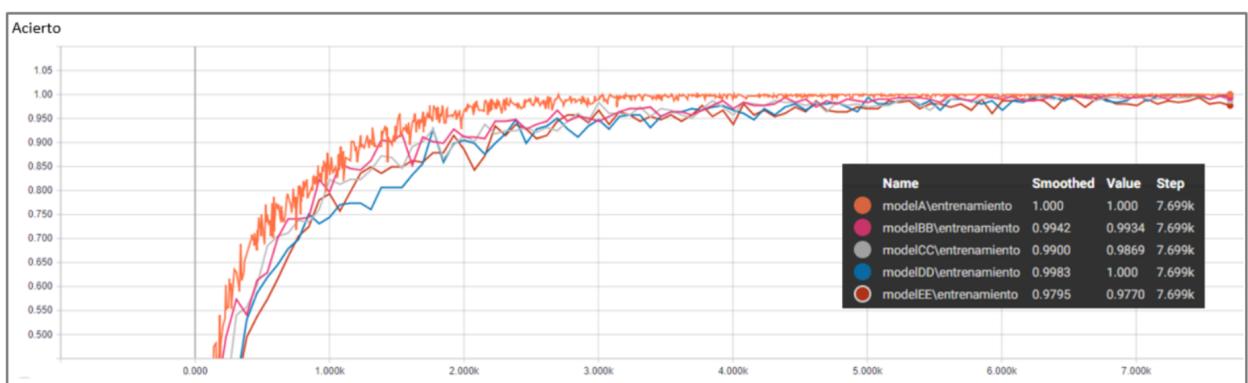


Figura 3.58: Análisis del Acierto durante el Entrenamiento de los modelos - Dataset Señales de Tránsito de Perú

Fuente: Elaboración propia

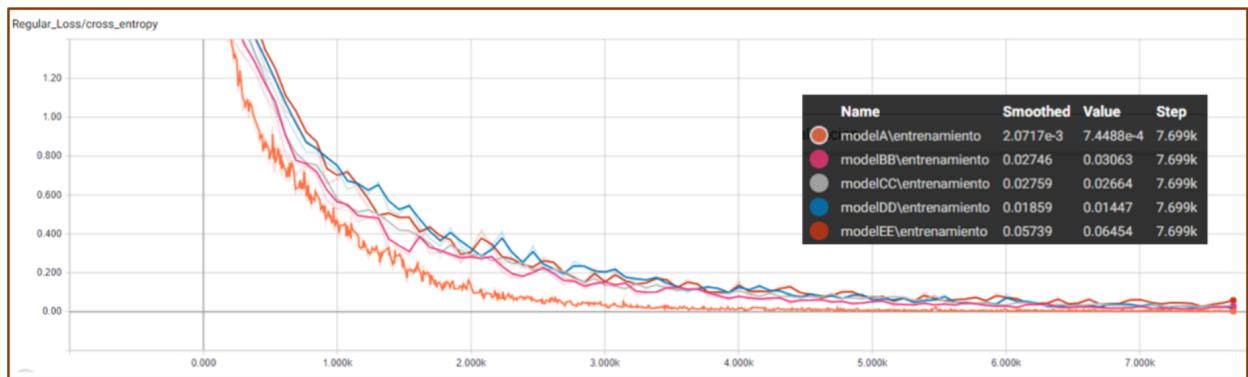


Figura 3.59: Análisis del Error durante el Entrenamiento de los modelos - Dataset Señales de Tránsito de Perú

Fuente: Elaboración propia

A través de la validación del modelo (usando dataset de imágenes no analizadas durante el entrenamiento), se observa que el Modelo E obtuvo las mejores tasas de Acierto y Error.

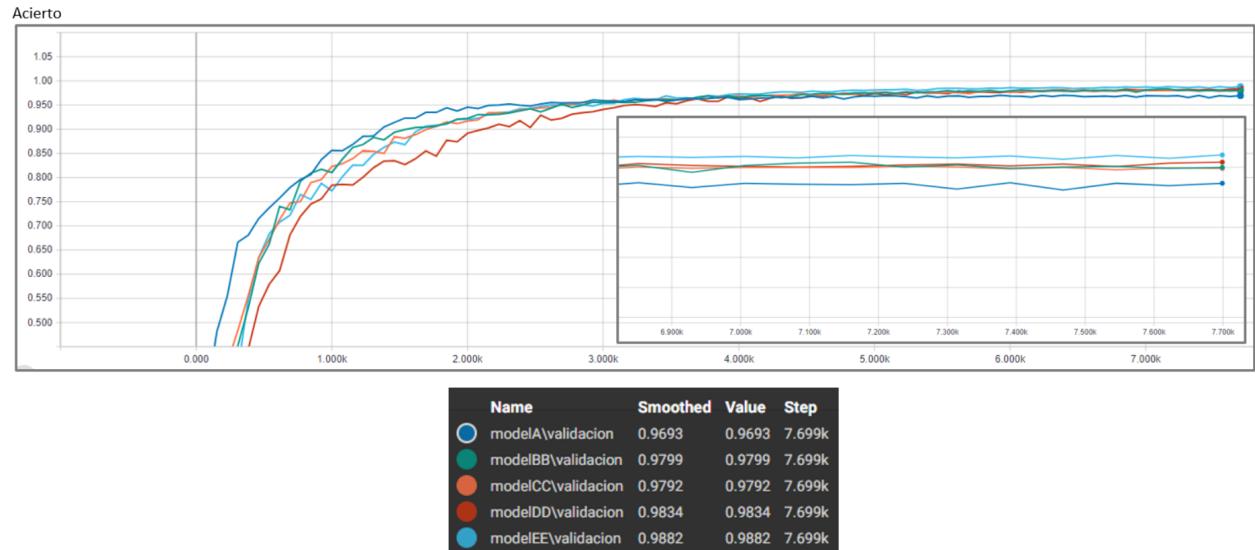


Figura 3.60: Análisis del Acierto en la Validación de los modelos - Dataset Señales de Tránsito de Perú

Fuente: Elaboración propia

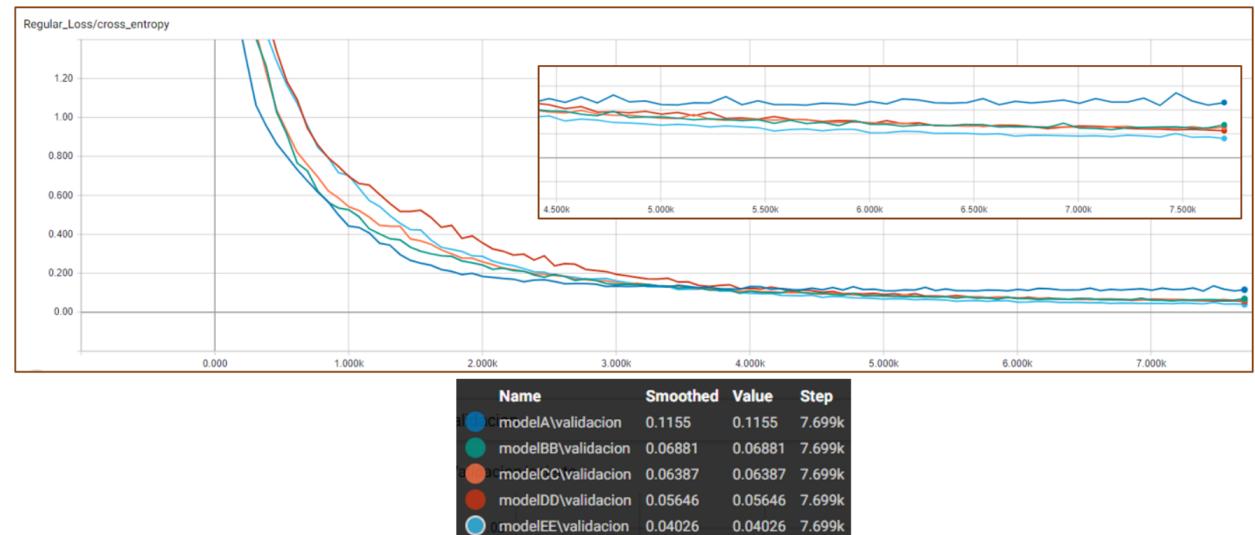


Figura 3.61: Análisis del Error en la Validación de los modelos - Dataset Señales de Tránsito de Perú

Fuente: Elaboración propia

Capítulo 4

Resultados de la tesis

Al culminar con la investigación, para la prueba de resultados de clasificación se dispone de una matriz de confusión. La matriz de confusión es una matriz cuadrada cuyo orden es el número de clases. En las columnas se presentan las clases reales mientras que en las filas se presentan las clases asignadas por el clasificador, por ende, la suma vertical muestra la distribución real de las clases, mientras que la suma horizontal muestra la distribución de las clases producida por el clasificador.

El análisis de los clasificadores se hace con base en la matriz de confusión de la cual se obtienen los 4 de los 6 indicadores de desempeño mencionados en la **Sección 2.5.3 Indicadores**.

- Prop. de Verdaderos Positivos(Efectividad): $PVP = \frac{\text{Verdaderos Positivos}}{\text{Verdaderos Positivos} + \text{Falsos Negativos}}$
- Prop. de Verdaderos Negativos(Especificidad): $PVN = \frac{\text{Verdaderos Negativos}}{\text{Verdaderos Negativos} + \text{Falsos Positivos}}$

- Valor Predictivo Positivo (Precisión): $PPV = \frac{Verdaderos\ Positivos}{Verdaderos\ Positivos+Falsos\ Positivos}$
- Acierto (Exactitud): $ACC = \frac{Verdaderos\ Positivos+Verdaderos\ Negativos}{Total\ de\ Imagenes}$

Donde:

- *Verdaderos Positivos*: Imágenes correctamente identificadas.
- *Falsos Positivos*: Imágenes incorrectamente identificadas.
- *Verdaderos Negativo*: Imágenes correctamente rechazadas.
- *Falsos Negativos*: Imágenes incorrectamente rechazadas.

Utilizando los anteriores indicadores, podemos obtener dos más:

- Curvas ROC: Relación entre Efectividad y Especificidad
- Curvas PR: Relación entre Precisión y Efectividad(Recall)

Las curvas ROC (Característica Operativa del Receptor, del inglés - *Receiver Operating Characteristic*) define un sistema de coordenadas usadas para visualizar el desempeño del clasificador.

Las curvas ROC presentan el compromiso entre efectividad (sensibilidad) y especificidad del clasificador, un aumento en la sensibilidad está acompañado por un decremento en la especificidad. Esto significa que la esquina superior izquierda de la gráfica es el punto ideal, debido a que muestran la relación entre las muestras clasificadas adecuadamente (*Proporción de Verdaderos Positivos - PVP*) y las muestras que no pertenecen a la clase pero se clasificaron como si lo fueran (*Proporción*

de Falsos Positivos - PFP). Esta última, calculada a través de la Especificidad (PVN), siendo:

$$PFP = 1 - PVN$$

Las curvas ROC se utilizan normalmente en la clasificación binaria para estudiar la salida de un clasificador. Para dibujarlas se construye la línea convexa formada por los puntos (PFP, PVP) de los clasificadores que se estén evaluando, junto con los puntos de los clasificadores triviales (0,0) y (1,1). La curva más cercana a los bordes izquierdo y superior en el espacio ROC, es la prueba más acertada porque significa que hay mayor acierto. La curva que más se acerque a la diagonal de 45 grados en el espacio ROC, es la prueba menos acertada. Para extender la curva ROC y obtener un área ROC a la clasificación de múltiples clases, como es el caso de esta investigación, solo es necesario binarizar la salida. Se puede dibujar una curva ROC por clase, pero también se puede dibujar una curva ROC considerando cada elemento de la matriz de confusión por clase, como una predicción binaria (micro-promedio). Otra medida de evaluación para la clasificación de múltiples clases es el macro promedio, que otorga igual peso a la clasificación de cada etiqueta. El mejor sistema de entrenamiento es el que produce un conjunto de clasificadores que maximice el área bajo la curva (AUC - *Area Under the Curve*), (Sandoval and Prieto, 2009).

De manera similar, las Curvas PR (*Precision -Recall*) representan la relación entre la Efectividad y Precisión del clasificador. El objetivo es tener un modelo que se posicione en la esquina superior derecha, que básicamente consiste en obtener solo los positivos verdaderos sin falsos po-

sitivos ni falsos negativos: un clasificador perfecto. En la situación en la que se tiene clases con cantidades no balanceadas, como es el caso del Dataset- Perú, a menudo es más útil utilizar el Área bajo la Curva PR como indicador del clasificador, (Davis and Goadrich, 2006).

4.1. Señales de Tránsito de Alemania

Luego de evaluar cada uno de los 5 modelos durante 100 épocas, el diseño del modelo E es el que presentó los mejores resultados.

4.1.1. Tabla de Resultados

Tabla 4.1: Indicadores de los 5 modelos entrenados en el Dataset - Alemania

Indicadores	Modelo A(%)	Modelo B(%)	Modelo C(%)	Modelo D(%)	Modelo E(%)
PVP	96.14	98.35	98.08	98.37	98.61
PVN	99.93	99.96	99.96	99.96	99.97
PPV	95.69	97.52	97.57	97.78	98.01
AUC-PR	94.31	96.88	96.60	96.93	97.30
AUC-ROC	98.04	99.15	99.02	99.17	99.29
ACC	97.08	98.41	98.27	98.43	98.62

4.1.2. Matriz de Confusión

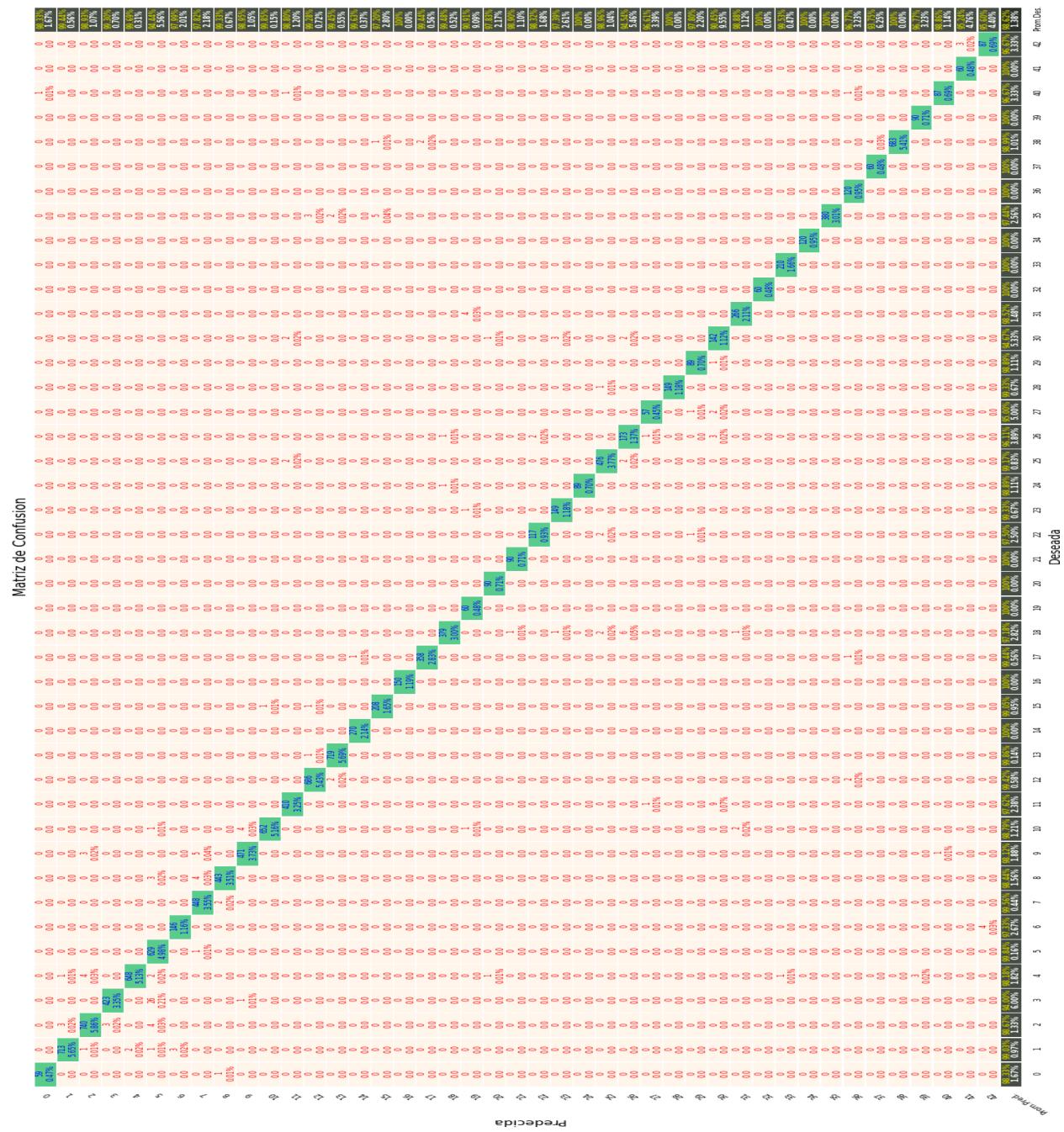


Figura 4.1: Matriz de Confusión del Modelo E - Dataset de imágenes de Alemania

Fuente: Elaboración propia

4.1.3. Curvas ROC - AUC

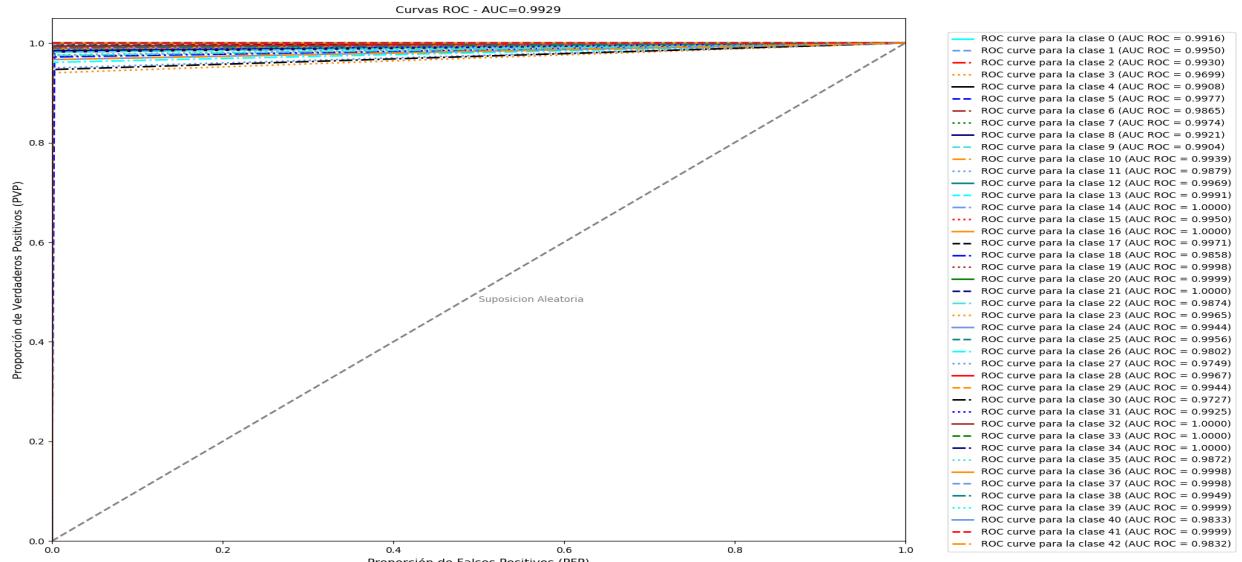


Figura 4.2: Área debajo de la Curva ROC del Modelo E - Dataset de imágenes de Alemania

Fuente: Elaboración propia

4.1.4. Curvas PR - AUC

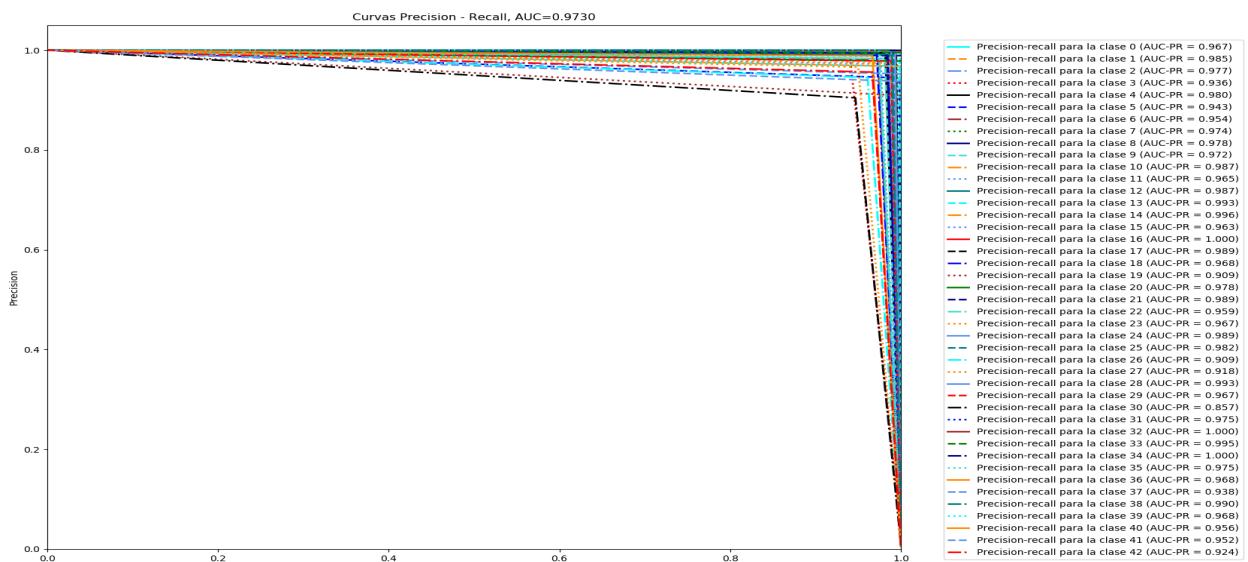


Figura 4.3: Área debajo de la Curva PR del Modelo E - Dataset de imágenes de Alemania

Fuente: Elaboración propia

4.2. Señales de Tránsito de Perú

Luego de evaluar cada uno de los 5 modelos, el diseño del **modelo E** también es el que presentó los mejores resultados.

4.2.1. Tabla de Resultados

Tabla 4.2: Indicadores de los 5 modelos entrenados en el Dataset - Perú

Indicadores	Modelo A(%)	Modelo B(%)	Modelo C(%)	Modelo D(%)	Modelo E(%)
PVP	95.78	97.61	98.21	98.09	98.81
PVN	99.44	99.69	99.75	99.69	99.83
PPV	96.75	98.29	98.51	98.04	98.94
AUC-PR	94.06	96.75	97.33	96.71	98.19
AUC-ROC	97.62	98.66	98.99	98.90	99.33
ACC	96.74	98.23	98.55	98.21	99.02

4.2.2. Matriz de Confusión

		Matriz de Confusión						
		Predicción			Deseada			Res. Dese.
Deseada	Predicción	0	1	2	3	4	5	6
0	0	1025 21.82%	0 0.02%	0 0.0	0 0.02%	0 0.0	0 0.02%	99.81% 0.19%
1	0	2 0.04%	832 17.71%	3 0.06%	0 0.0	1 0.02%	0 0.0	99.17% 0.83%
2	0	4 0.09%	683 14.54%	4 0.09%	0 0.02%	1 0.02%	0 0.04%	97.57% 2.43%
3	0	5 0.11%	0 0.02%	0 0.0	812 17.28%	0 0.0	2 0.04%	98.78% 1.22%
4	0	0 0.02%	0 0.0	0 0.02%	615 13.09%	2 0.04%	1 0.02%	99.35% 0.65%
5	0	0 0.02%	0 0.0	0 0.02%	445 9.47%	1 0.02%	0 0.02%	99.55% 0.45%
6	0	0 0.02%	0 0.0	0 0.02%	0 0.0	1 0.02%	240 5.11%	98.36% 1.64%
		99.13% 0.87%	98.81% 1.19%	99.42% 0.58%	99.39% 0.61%	99.35% 0.65%	98.45% 1.55%	97.17% 2.83%

Figura 4.4: Matriz de Confusión del Modelo E - Dataset de imágenes de Perú

Fuente: Elaboración propia

4.2.3. Curvas ROC - AUC

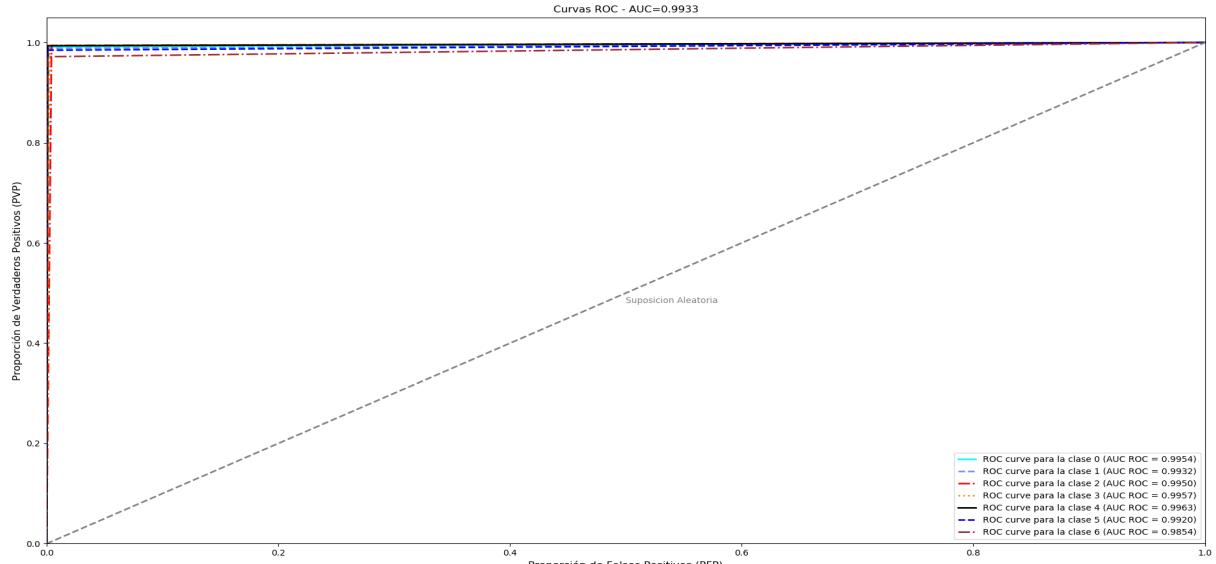


Figura 4.5: Área debajo de la Curva ROC del Modelo E - Dataset de imágenes de Perú

Fuente: Elaboración propia

4.2.4. Curvas PR - AUC

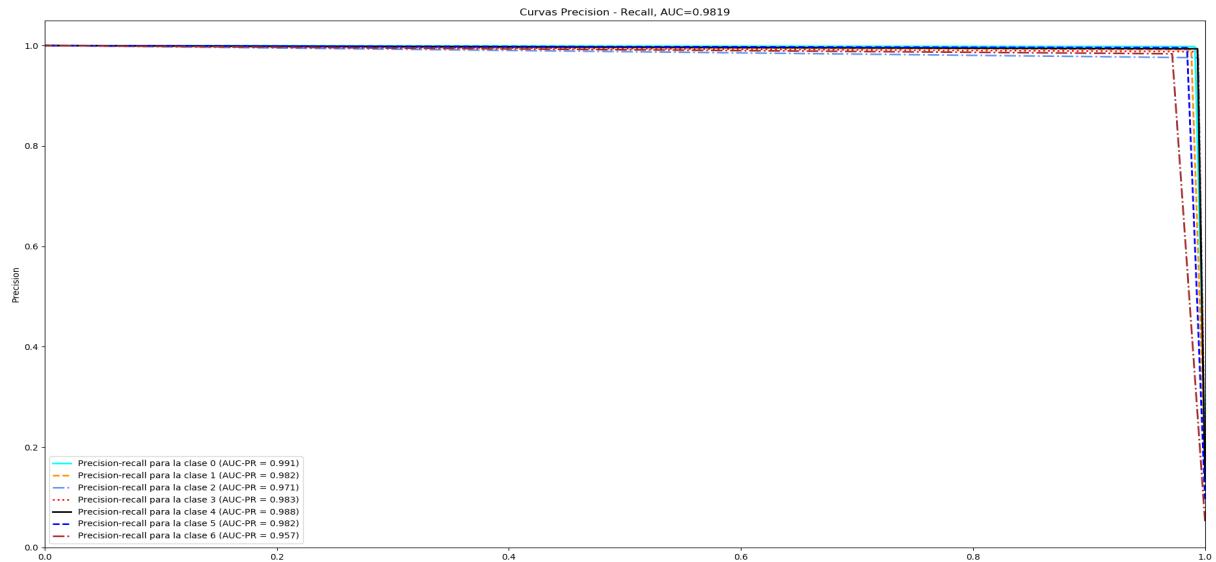


Figura 4.6: Área debajo de la Curva PR del Modelo E - Dataset de imágenes de Perú

Fuente: Elaboración propia

4.3. Reconocimiento de Señal

Para continuar con la evaluación del rendimiento en el reconocimiento de señales de tránsito, los modelos para el Dataset de señales de Alemania y Perú con mejores resultados fueron utilizados en una aplicación programada en lenguaje Python, la cual se encarga de analizar y reconocer una nueva imagen que contenga una señal de tránsito.

A continuación, se muestran los pasos necesarios para reconocer una imagen que contiene una señal de tránsito:

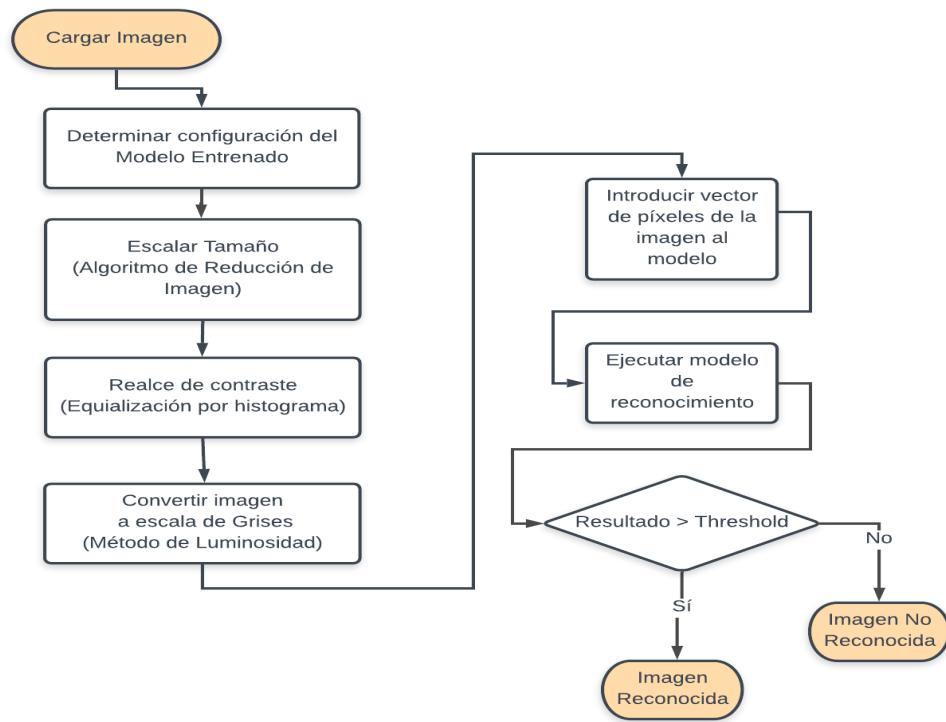


Figura 4.7: Flujograma para el reconocimiento de una señal de tránsito

Fuente: Elaboración propia

4.3.1. Interfaz de Aplicación



Figura 4.8: Interfaz reconociendo Señal de Tránsito “Wild Animals Crossing” - Alemania

Fuente: Elaboración propia

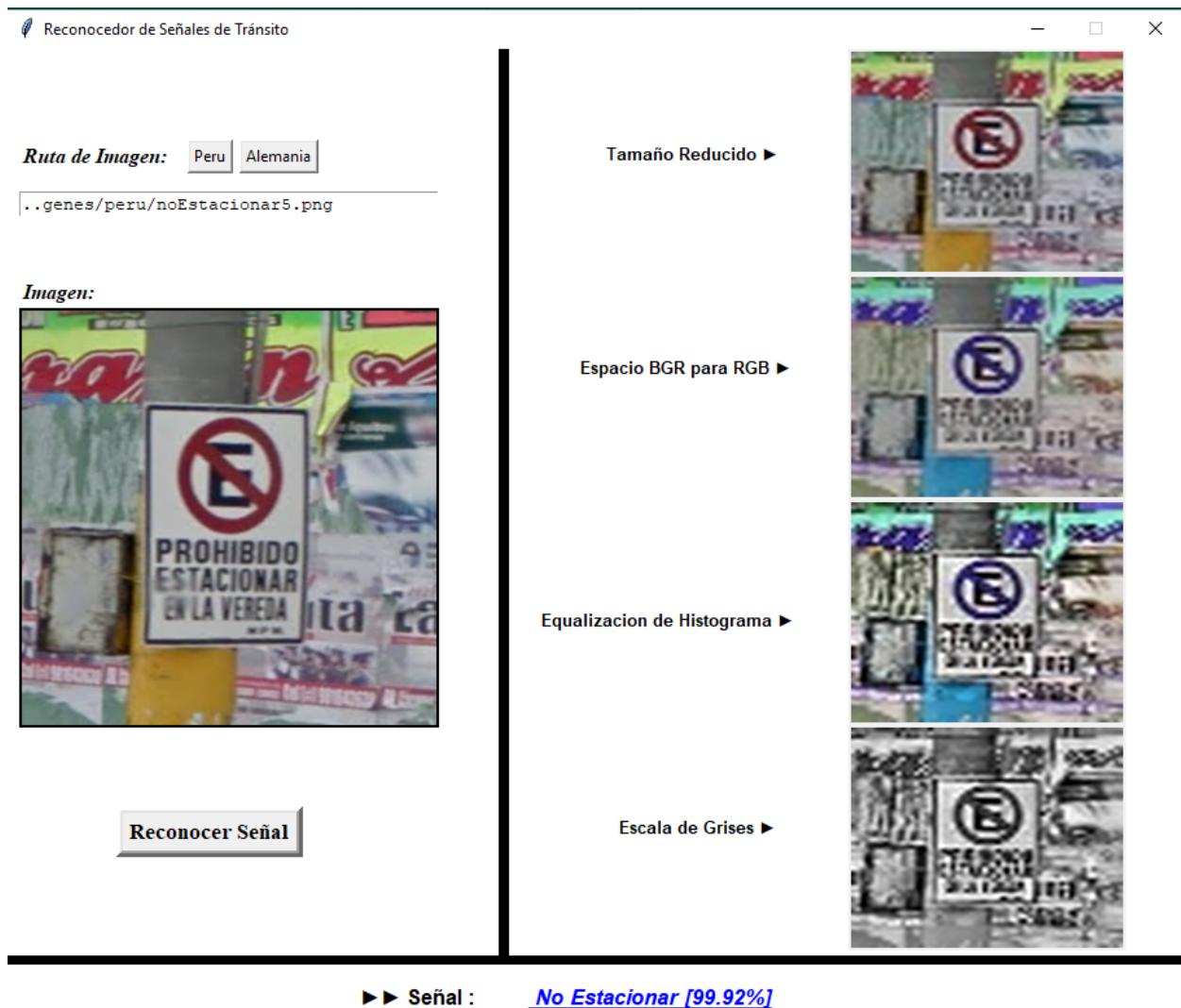


Figura 4.9: Interfaz reconociendo Señal de Tránsito “No Estacionar” - Perú

Fuente: Elaboración propia

Capítulo 5

Consideraciones finales

5.1. Conclusiones

Al finalizar la investigación podemos confirmar que se cumplieron los objetivos específicos planteados.

Se obtuvo dos datasets de imágenes de señales de tránsito, uno de Alemania distribuidas en 43 categorías y otro de Perú distribuidas en 7 categorías, compuestos inicialmente por 51839 imágenes y 614 imágenes respectivamente.

Posteriormente, se analizaron los datasets y al dividirlos en 3 grupos (entrenamiento, validación y evaluación), se determinó la necesidad de aumentar ambos y esto fue realizado con la ayuda de técnicas de procesamiento de imágenes, resultando en dos datasets robustos de imágenes, el dataset

de Alemania compuesto por 283530 imágenes de las cuales 203175 fueron para entrenamiento, 67725 para validación y 12630 para evaluación y el dataset de Perú compuesto por 36012 imágenes de las cuales 23485 fueron para entrenamiento, 3131 para validación y 4698 para evaluación. Ambos datasets ofrecidos para futuras investigaciones.

Fueron diseñados cinco modelos de red para cada dataset, los cuales fueron entrenados, validados y evaluados individualmente.

Luego de experimentar el uso de diversas funciones de activación, funciones de costo, ajuste de parámetros y métodos de optimización, se logró determinar un conjunto de hiperparámetros (Sección 3.2) de los cuales solo la Tasa de Aprendizaje varía para cada dataset de imágenes.

Al finalizar el análisis de los 10 modelos (cinco por cada dataset), se puede observar que cuanto más profunda sea la red neuronal, se obtienen mejores resultados.

De este modo, el modelo final obtenido compuesto principalmente de 4 capas convolucionales, 2 capas totalmente conectadas y un total de 90185 neuronas (Sección 3.2.1.5 - Diseño E - Alemania), contribuye en el reconocimiento de señales de Tránsito de Alemania con una tasa de acierto del **98.62 %**, mucho mejor que el resultado de 95.29 % obtenido por (Ayuque Arenas, 2016) y mucho más próximo al mejor resultado de 99.46 % obtenido en las investigaciones hechas en base al dataset GTSRB, (Cireşan et al., 2012).

Para el dataset de señales de tránsito vehicular del Perú, el modelo E con el mismo diseño neuronal y similares configuraciones compuesto por 279623 neuronas (Sección 3.2.1.5 - Diseño E

- Perú) permite también obtener un **alto grado de acierto (99.02 %)** tras analizar 4698 imágenes.

En conclusión, el objetivo general que trata sobre implementar un modelo basado en el aprendizaje profundo de redes neuronales convolucionales para reconocer automáticamente señales de tránsito vehicular fue conseguido a través del Modelo E.

5.2. Trabajos futuros y recomendaciones

El modelo puede ser mejorado(ampliado), teniendo muchas más capas convolucionales y capas totalmente conectadas para poder experimentar si existe o no alguna mejora en los resultados. Además, se recomienda obtener muchas más imágenes para exceder el rendimiento humano, (Goodfellow et al., 2016)

El dataset de señales de tránsito del Perú también puede ser ampliado con la finalidad de abarcar más categorías, ya que se tiene confianza por lo mostrado con el dataset de Alemania que el modelo es robusto para soportar mayor cantidad de estas.

Se sugiere integrar el modelo obtenido en un sistema más general que primero localice las señales de tránsito en escenas que abarcan más de una señal de tránsito, para luego proceder a su multireconocimiento.

Bibliografía

- Ayuque Arenas, K. J. M. (2016). Diseño de un sistema de clasificación de señales de tránsito vehicular utilizando redes neuronales convolucionales. *Universidad San Ignacio de Loyola.*
- Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg.
- Cireşan, D., Meier, U., Masci, J., and Schmidhuber, J. (2012). Multi-column deep neural network for traffic sign classification. *Neural Networks from IJCNN 2011*, 32(C):333 – 338.
- Consejo Nacional de Seguridad Vial, . (2014). La seguridad vial a nivel mundial.
- Cook, J. D. (2009). Three algorithms for converting color to grayscale.
<https://www.johndcook.com/blog/2009/08/24/algorithms-convert-color-grayscale>.
- Cordova Rampant, A. (2017). Los peruanos mueren más por los accidentes de tránsito que por la inseguridad ciudadana. <http://rpp.pe/data/los-peruanos-mueren-mas-por-los-accidentes-de-transito-1071653>.
- Davis, J. and Goadrich, M. (2006). The relationship between precision-recall and roc curves.

- Dong, X. and Zhou, D.-X. (2008). Learning gradients by a gradient descent algorithm. *Journal of Mathematical Analysis and Applications*, 341(2):1018 – 1027.
- Donges, N. (2018). Gradient descent in a nutshell. [Online; accessed Agosto 08, 2018].
- Elkan, C. (2012). Evaluating classifiers.
- Gestión.pe, D. (2016a). Accidentes de tránsito: Dos propuestas para aumentar la seguridad vial y reducir muertes. <https://gestion.pe/economia/accidentes-transito-dos-propuestas-2175696>.
- Gestión.pe, D. (2016b). Seguridad vial en perú: Nueve indicadores a mejorar para evitar accidentes. <https://gestion.pe/economia/seguridad-vial-peru-21756832>.
- Golik, P., Doetsch, P., and Ney, H. (2013). Cross-entropy vs. squared error training: a theoretical and experimental comparison.
- Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.
<http://www.deeplearningbook.org>.
- Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., and Wang, G. (2015). Recent advances in convolutional neural networks. *CoRR*, abs/1512.07108.
- Hai Nguyen, T. (2014). Morphological classification for traffic sign recognition. *Faculty of Electrical and Electronic Engineering*, 4(2):36–44.
- Hannan, M. A., Wali, S. B., Pin, T. J., Hussain, A., and Samad, S. A. (2014). Traffic sign classification based on neural network for advance driver assistance system. *Przeglad Elektrotechniczny*, R. 90, nr 11:169–172.

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., and Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22Nd ACM International Conference on Multimedia*, MM '14, pages 675–678, New York, NY, USA. ACM.

Jordan, J. (2018). Setting the learning rate of your neural network. [Online; accessed Agosto 08, 2018].

Karpathy, A. (2016). Convolutional neural networks (cnns / convnets). *Cs231- Stanford Course*.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Krizhevsky, A., Sutskever, I., and Hinton, G. (2012). Imagenet classification with deep convolutional neural networks.

LeCun, Y., L.Botou, Y.Bengio, and P.Haffner (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86.

Moore, A. W. (2001). Cross-validation for detecting and preventing overfitting. *School of Computer Science Carnegie Mellon University*.

Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 807–814, USA. Omnipress.

OMS (2017). 10 datos sobre la seguridad vial en el mundo.

<http://www.who.int/features/factfiles/roadsafety/es/>.

Poynton, C. (2003). 23 - gamma. In Poynton, C., editor, *Digital Video and HDTV*, The Morgan Kaufmann Series in Computer Graphics, pages 257 – 280. Morgan Kaufmann, San Francisco.

Rocha, C. and Escorcia, G. (2010). Sistema de visión artificial para la detección y el reconocimiento de señales de tráfico basado en redes neuronales. *Eighth LACCEI Latin American and Caribbean Conference for Engineering and Technology (LACCEI)*.

Rohrer, B. (2016). How convolutional neural networks work. *Data Science and Robots Blog*.

Romero, R. F. (2015a). Scc0270/scc5809 - redes neurais - deep learning.

Romero, R. F. (2015b). Scc0270/scc5809 - redes neurais - multilayer perceptron.

Romero Benites, R. (2017). Ocho peruanos mueren cada día en accidentes de tránsito.
<http://rpp.pe/data/ocho-peruanos-mueren-cada-dia-en-accidentes-de-transito-1068532>.

Sandoval and Prieto (2009). Procesamiento de imágenes para la clasificación de café cereza.

Sermanet, P. and LeCun, Y. (2011). Traffic sign recognition with multi-scale convolutional networks.

Stallkamp, J., Schlipsing, M., Salmen, J., and Igel, C. (2011). The german traffic sign recognition benchmark: A multi-class classification competition. In *IEEE International Joint Conference on Neural Networks*, pages 1453–1460.

Stallkamp, J., Schlipsing, M., Salmen, J., and Igel, C. (2012). Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition.

SUTRAN (2014). Texto unico ordenado del reglamento nacional de transito - codigo de transito.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S. E., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2014). Going deeper with convolutions. *CoRR*, abs/1409.4842.

Taylor, L. and Nitschke, G. (2017). Improving deep learning using generic data augmentation. *CoRR*, abs/1708.06020.

Vargas Romero, F. (2015). Implementación de un sistema inteligente para el reconocimiento de señales preventivas de seguridad vial. *Universidad Nacional de Trujillo*.

Vicen-Bueno, R., Torijano-Gordo, E., Gil-Pita, R., and Rosa-Zurera, M. (2007). *Traffic Sign Classification by Image Preprocessing and Neural Networks*. Springer Berlin Heidelberg, Berlin, Heidelberg.

Waze (2016). Las mejores y peores ciudades para conducir en américa latina, según waze. <http://cnnespanol.cnn.com/2016/09/15/las-mejores-y-peores-ciudades>.

Yadav, G., Maheshwari, S., and Agarwal, A. (2014). Contrast limited adaptive histogram equalization based enhancement for real time video system. In *2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 2392–2397.

Zeng, Y., Xu, X., Fang, Y., and Zhao, K. (2015). Traffic sign recognition using deep convolutional networks and extreme learning machine. volume 9242, pages 272–280.

Apéndice A

Pseudocódigo del Diseño de Red

Algoritmo 1: Función Principal

```
1 entradas ← Tensor(LoteImagenes, Anchura, Altura, Profundidad);
2 salidaDeseada ← Tensor(LoteImagenes, NumeroClases);
3 numFiltrosEntradaCapaN ← 1;
4 mientras  $N < \text{numCapasConvolucionales}$  hacer
5   capaConvN, pesosConvN ← capaConvolucion( datosImagen = entradas,
    numCanales = numFiltrosEntradaCapaN, tamFiltro = tamFiltroCapaN,
    numFiltros = numFiltrosSalidaCapaN,
    usarPooling = True
  );
6   capaConvDropOutN ← dropout(capaConvN, dropoutConvN);
7   entradas ← capaConvDropOutN;
8   numFiltrosEntradaCapaN ← numFiltrosSalidaCapaN;
9 fin
10 mientras  $N < (\text{numCapasConvolucionales} - 1)$  hacer
11   capaConvDropOutN ← ejecutarPooling( data = capaConvDropOutN,
    tamFiltro =  $2^{(\text{numCapasConvolucionales} - N)}$ 
  );
12   datosCapaN ← (datosCapaN + capaConvDropOutN[Anchura] *
    capaConvDropOutN[Altura] * capaConvDropOutN[Profundidad]);
13   capaSalida ← (capaSalida + capaConvDropOutN);
14 fin
15 capaFC1, pesosFC1 ← capaTotalmenteConectada( datosImagen = capaSalida,
  numCanales = datosCapaN,
  tamFiltro = numSalidas, ....[DadasAleatoriamente]
  usarRELU = True);
16 capaFC1dropOut ← dropout(capaFC1, dropoutFC1 = 0,5);
17 capaFC2, pesosFC2 ← capaTotalmenteConectada(  

  datosImagen = capaFC1dropOut,  

  numCanales = numSalidas,  

  tamFiltro = numClases,  

  usarRELU = False);
18 salidacalculada ← softmax(capaFC2);
19 error ← entropiaCruzada(salidacalculada, salidadeseada);
20 optimizador ← Adam(TasaAprendizaje).minimize(error, iterac);
```

Procedimiento capaConvolucion(*Img*, *numCanales*, *tamFiltro*, *numFiltros*, *usarPool*)

Entrada: *datosImagen*, *numCanales*, *tamFiltro*, *numFiltros*, *usarPooling*

Salida: *MatrizConvolucion*, *pesos*

```
1 forma  $\leftarrow$  [tamFiltro, tamFiltro, numCanales, numFiltros];
2 pesos  $\leftarrow$  inicializarPesos(forma);
3 biases  $\leftarrow$  inicializarBias(numFiltros);
4 convolucion  $\leftarrow$  convolucion2D( input = datosImagen,
    filter = pesos,
    formaPadding = [1, 1, 1, 1],
    padding = activado);
5 convolucion  $\leftarrow$  convolucion + biases;
6 convolucion  $\leftarrow$  funcionReLU(convolucion);
7 si usarPooling = True entonces
8   convolucion  $\leftarrow$  ejecutarPooling(convolucion, 2);
9 return (convolucion, pesos)
```

Procedimiento FuncionReLU(*datosDeConvolucion*)

Entrada: datos de Convolucion

Salida: Datos con funcion RELU aplicada

```
1 para cada item in datosDeConvolucion hacer
2   nuevosDatos[i]  $\leftarrow$  max(datosDeConvolucion[i], 0);
3 return nuevosDatos
```

Procedimiento entropiaCruzada(*vectorSalidaCalculada*,*vectorSalidaDeseada*)

Entrada: Vector de Salida Calculada y vector de Salida Deseada

Salida: Validación Cruzada de Vectores

```
1 crossEntropy  $\leftarrow$  -suma(vectorSalidaDeseada * nlog(vectorSalidaCalculada));
2 return crossentropy
```

Procedimiento softmax(*vectorDatos*)

Entrada: Vector de datos

Salida: Vector de datos entre 0 y 1

```
1 return  $e^{\text{vectorDatos}} / \sum_{k=1}^K e^{\text{vectorDatos}_k};$ 
```

Procedimiento capaTotalmenteConectada(entrada,numEntradas,numSalidas,usarRELU)

Entrada: entrada, numEntradas, numSalidas, usarRELU

Salida: Vector de datos Resultantes, pesos por Item del Vector

```
1 pesos ← inicializarPesos([numEntradas, numSalidas]);  
2 biases ← inicializarBias(numSalidas);  
3 capaFC ← (entrada * pesos) + biases;  
4 si usarRELU = True entonces  
5   capaFC ← funcionRELU(capaFC);  
6 return (capaFC, pesos)
```

Código de la implemantación completa ubicado en: <https://github.com/jtavara23/SignalsWindows>