

UNIVERSIDAD NACIONAL DE TRUJILLO

Facultad de Ciencias Físicas y Matemáticas

Escuela Académico 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: Dr. Jorge Luis Gutierrez Gutierrez

Trujillo - La Libertad

2018

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

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

Tesis presentada a la Escuela Académico Profesional de Informática en la Facultad de Ciencias Físicas y Matemáticas de la Universidad Nacional de Trujillo, como requisito parcial para la obtención del grado de Bachiller en ciencia de la computación (Título profesional de Ing. Informático)

AUTOR: JOSUÉ GASTÓN TÁVARA IDROGO

ASESOR: JORGE LUIS GUTIERREZ GUTIERREZ

Trujillo - La Libertad

2018

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, 23 de octubre del 2018

Dedico esta tesis a :

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

Mis hermanas por el apoyo continuo y paciencia.

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 . . .

A mi asesor Prof. Dr. Jorge Luis Gutierrez Gutierrez que siempre se mostro 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 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.

Esta investigación pretende contribuir en la industria automotriz, específicamente en los campos de construcción de vehículos autónomos y de los sistemas avanzados de asistencia al conductor, iniciando con la adquisición de imágenes, luego para el pre-procesamiento, se implementará algoritmos de realce de contraste, reducción de ruido, rotación y proyecciones de escalamiento con la finalidad de aumentar el conjunto de datos y poder ejecutar el aprendizaje profundo a través de arquitecturas de redes neuronales convolucionales. Se van a realizar diferentes diseños de arquitecturas convolucionales y se escogerá el que obtenga los mejores resultados.

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

Abstract

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

This research aims to contribute in the automotive industry, specifically in the fields of autonomous vehicle construction and advanced driver assistance systems, starting with the acquisition of images, then for the previous processing, the algorithm of contrast reality, reduction is implemented of noise, rotation and scaling projections in order to increase the data set and be able to execute deep learning through convolutional neural network architectures. Different designs of convolutional architectures will be made and the one that obtains the best results will be chosen.

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

Lista de símbolos

Constantes:

- (1) r, \bar{r} Indice que denota regiones.
- (2) n Indice de bienes finales deseados por los consumidores.
- (3) ...

Variables:

- (5) x^r Vector columna que denota la actividad de producción.
- (6) u^r ...

Índice de figuras

1.1.	Señalización de velocidad	2
1.2.	Análisis de responsabilidad de accidentes	3
1.3.	Influencia de velocidad en accidentes	4
1.4.	Algunas muestras de la base de datos GTSRB	8
1.5.	Diagrama de bloques del modelo preentendido	10
1.6.	Diseño del Modelo del Ciclo de Vida del Desarrollo	14
2.1.	Ilustración de un modelo de aprendizaje profundo	20
2.2.	Diagrama de Venn donde cada sección incluye un ejemplo de una tecnología de IA(ilustra la relación entre estas diferentes disciplinas de la IA)	23
2.3.	Entrada y salida de una CNN	24
2.4.	Analisis de una CNN	25
2.5.	Terminología de capas complejas(izquierda) y de capas simples(derecha)	26
2.6.	Convolucion entre el filtro y parte de la imagen	27

2.7.	Posicionamiento del kernel/filtro por pixel	27
2.8.	Proceso matemático convolucional	28
2.9.	Cálculo Convolutional	28
2.10.	Conectividad dispersa	29
2.11.	Resultado de Convolución(conjunto de mapas de activación. generado a partir de 3 fil- tros para 3 carecteristicas: diagonal derecha, cruzamiento central y diagonal izquierda).	
	Simbolo de convolución: \otimes	31
2.12.	Imagen con stride igual a 2, para el filtro tanto en largura como anchura	32
2.13.	Ejemplo de zero-padding con tamaño 2	33
2.14.	Ejemplo de filtro crado apartir de los 3 aspectos mencionados	34
2.15.	Funciones de activación	35
2.16.	Procedimiento de la función ReLU	36
2.17.	En la izquierda la red neuronal común y a la derecha la red neuronal diluida producida por la aplicación de dropout	37
2.18.	Procesos que pueden ocurrir durante entrenamiento. Dropout evita el sobreajuste . . .	38
2.19.	Formas de agrupamiento(pooling)	39
2.20.	Operación MAX-pooling en capa de Agrupación	39
2.21.	Resultado de Agrupación	40
2.22.	Max-pooling con filtro 2x2 y paso 2	41

2.23. Modelo de red neuronal convolucional profunda	42
2.24. Establecimiento de la Tasa de Aprendizaje	44
2.25. Establecimiento de la Tasa de Aprendizaje	47
2.26. Problemas con la Tasa de Aprendizaje	47
2.27. Ejemplo de Tasa de decaimiento de aprendizaje por época	48
 4.1. Speed limit (20km/h)	59
4.2. Speed limit (50km/h)	59
4.3. Bumpy road	60
4.4. Children crossing	60
4.5. Wild animals crossing	60
4.6. Turn left ahead	60
4.7. Distribución de ejemplos por señal para el entrenamiento(Total 39209)	60
4.8. Pare	61
4.9. Resalto	61
4.10. Zona Escolar	61
4.11. Peatones	61
4.12. Paradero	61
4.13. No Estacionar	61
4.14. Disminuir Velocidad	62

4.15. Distribución de ejemplos por señal para el entrenamiento(Total 614)	62
4.16. Distribución de ejemplos por señal para la evaluación - Alemania	63
4.17. Distribución de ejemplos por señal para la evaluación - Perú	63
4.18. El aumento de datos infla artificialmente los conjuntos de datos usando transformaciones que preservan las categorías de los objetos	64
4.19. Imágenes volteadas horizontalmente	65
4.20. Imágenes volteadas verticalmente	65
4.21. Imágenes volteadas primero horizontal y luego verticalmente	66
4.22. Imágenes que volteadas horizontal o verticalmente, cambian su categoría	66
4.23. Distribución de categoría, luego de aplicar el Flip(en sus distintos tipos)- Señales de Alemania	67
4.24. Ejemplo de cinco proyecciones por cada imagen - Dataset Alemania	68
4.25. Ejemplo de cinco rotaciones por cada imagen - Dataset Alemania	69
4.26. Ejemplo de cinco aplicaciones de zoom(in/out) por cada imagen - Dataset Alemania .	69
4.27. Ilustración de un modelo de aprendizaje profundo	70
4.28. Dataset balanceado(cada categoría posee igual cantidad de imágenes)	71
4.29. Dataset no balanceado - Señales de Tránsito de Perú	72
4.30. Imágenes a las cuales se le aplicaron la técnica CLAHE	73
4.31. Imágenes procesadas en escala de grises	74

4.32. Arquitectura de la CNN	76
4.33. Modelo del diseño de la Red propuesta	78
4.34. Modelo A Tasa de Acierto por iteración - Dataset de imágenes de Alemania	82
4.35. Modelo A Tasa de pérdida por iteración	82
4.36. Modelo B Tasa de Acierto por iteración - Dataset de imágenes de Alemania	83
4.37. Modelo B Tasa de pérdida por iteración	83
4.38. Modelo C Tasa de Acierto por iteración - Dataset de imágenes de Alemania	84
4.39. Modelo C Tasa de pérdida por iteración	84
4.40. Modelo D Tasa de Acierto por iteración - Dataset de imágenes de Alemania	84
4.41. Modelo D Tasa de pérdida por iteración	85
4.42. Modelo E Tasa de Acierto por iteración - Dataset de imágenes de Alemania	85
4.43. Modelo E Tasa de pérdida por iteración	85
4.44. Modelo A Tasa de Acierto por iteración - Dataset de imágenes de Alemania	86
4.45. Modelo A Tasa de pérdida por iteración	86
4.46. Modelo B Tasa de Acierto por iteración - Dataset de imágenes de Alemania	87
4.47. Modelo B Tasa de pérdida por iteración	87
4.48. Modelo C Tasa de Acierto por iteración - Dataset de imágenes de Alemania	88
4.49. Modelo C Tasa de pérdida por iteración	88
4.50. Modelo D Tasa de Acierto por iteración - Dataset de imágenes de Alemania	89

4.51. Modelo D Tasa de pérdida por iteración	89
4.52. Modelo E Tasa de Acierto por iteración - Dataset de imágenes de Alemania	89
4.53. Modelo E Tasa de pérdida por iteración	90
4.54. Matriz de Confusión del Modelo A - Dataset de imágenes de Alemania	93
4.55. Matriz de Confusión del Modelo A - Dataset de imágenes de Perú	94

Índice de tablas

3.1.	Indicadores para la investigación	54
4.1.	Distribución Entrenamiento y Validación para Dataset Balanceado - Señales de Tránsito de Alemania	71
4.2.	Distribución Entrenamiento y Validación para Dataset no Balanceado - Señales de Tránsito de Perú	72
4.3.	Hiperparámetros del Modelo	75
4.4.	Minibatch e iteraciones en el Dataset de Señales de Tránsito de Alemania Balanceado .	75
4.5.	Minibatch e iteraciones en el Dataset de Señales de Tránsito de Peru No Balanceado .	76
4.6.	Diseño compuesto de 2 capas convolucionales y 2 capas totalmente conectadas	78
4.7.	Diseño compuesto de 3 capas convolucionales y 2 capas totalmente conectadas	79
4.8.	Diseño compuesto de 3 capas convolucionales y 2 capas totalmente conectadas. Variando el número de kernels en la 2da capa convolucional	79

4.9. Diseño compuesto de 3 capas convolucionales y 2 capas totalmente conectadas. Variando el número de kernels en la 3ra capa convolucional	80
4.10. Diseño compuesto de 4 capas convolucionales y 2 capas totalmente conectadas	81

Índice general

Dedicatoria	IV
Agradecimientos	V
Resumen	VI
Abstract	VII
Lista de símbolos	VIII
Índice de Figuras	XIV
Índice de Tablas	XVI
1. Introducción	1
1.1. Antecedentes de la Investigación	4
1.1.1. INTERNACIONALES:	4
1.1.2. NACIONALES:	7
1.1.3. ESTADO DEL ARTE:	7
1.2. Formulación del problema	9
1.3. Hipótesis	9
1.4. Importancia de la investigación	9

1.4.1. Justificación Académica	9
1.4.2. Justificación Social	10
1.5. Objetivos	11
1.5.1. Objetivo general	11
1.5.2. Objetivos específicos	12
1.6. Contribución de la investigación	12
1.7. Metodología de la investigación	13
1.7.1. Metodología para el diseño del Modelo	14
1.8. Estructura de la tesis	14
2. Marco teórico	15
2.1. Aprendizaje Profundo	15
2.2. Red Convolucional	23
2.2.1. Capa Convolucional	26
2.2.1.1. Etapa de Convolución	26
2.2.1.2. Capa ReLU(Rectified Linear Units)	35
2.2.1.3. Técnica Dropout	37
2.2.1.4. Capa de Agrupación(Pooling)	38
2.2.2. Capa totalmente conectada (Fully-connected layer)	41
2.3. Entrenamiento y Validación	43
2.3.1. Descenso de Gradiente	43
2.3.1.1. Gradiente Descendiente por Lotes(Batch Gradient Descent) .	45
2.3.1.2. Gradiente Descendiente Estocástico(Stochastic Gradient Descent-SGD)	45
2.3.1.3. Mini-batch Gradient Descent	46

2.3.2. Tasa de Aprendizaje (Learning Rate)	46
2.3.3. Optimizador ADAM	49
2.3.4. Validación Cruzada	49
2.3.5. Función Softmax	51
2.3.6. Método de Regularización L2	52
3. Materiales y Técnicas	53
3.1. Tipo de investigación	53
3.2. Variables de la Investigación	53
3.2.1. Variable Dependiente	53
3.2.2. Variable Independiente	54
3.3. Indicadores	54
3.4. Recolección de Datos para la Construcción del Modelo	54
3.4.1. Técnica de Recolección	54
3.4.2. Población	55
3.4.3. Muestra	55
3.4.3.1. Arquitectura AlexNet	55
3.4.3.2. Arquitectura Inception	56
3.5. Recolección de Datos para el Entrenamiento y Evaluación del Modelo	56
3.5.1. Técnica de Recolección	56
3.5.2. Población	57
3.5.3. Muestra	57
3.5.3.1. Señales de Tránsito de Alemania	57
3.5.3.2. Señales de Tránsito de Perú	58

4. Desarrollo de la Investigación	59
4.1. Análisis del conjunto de Imágenes	59
4.1.1. Datos Iniciales para el entrenamiento	59
4.1.1.1. Señales de Tránsito de Alemania	59
4.1.1.2. Señales de Tránsito de Perú	61
4.1.2. Datos para el evaluación	62
4.1.2.1. Señales de Tránsito de Alemania	62
4.1.2.2. Señales de Tránsito de Perú	63
4.1.3. Proceso de Aumento de Datos(Data Augmentation)	64
4.1.3.1. Flipping	64
4.1.3.2. Projection(Proyección)	67
4.1.3.3. Rotation(Rotación)	69
4.1.3.4. Zoom	69
4.1.3.5. Equalizacion del histograma	70
4.1.4. Dataset final para el Entrenamiento	71
4.1.4.1. Señales de Tránsito de Alemania - Dataset Balanceado . . .	71
4.1.4.2. Señales de Tránsito de Perú - Dataset No Balanceado . . .	72
4.1.5. Pre-procesamiento de Imágenes(Normalization)	73
4.2. Arquitectura del Modelo	75
4.2.1. Diseño de la Red	77
4.2.1.1. Diseño A	78
4.2.1.2. Diseño B	79
4.2.1.3. Diseño C	79
4.2.1.4. Diseño D	80

4.2.1.5. Diseño E	81
4.3. Entrenamiento y Validación	81
4.3.1. Señales de Tránsito de Alemania	82
4.3.1.1. Análisis del Entrenamiento y Validación del Diseño A	82
4.3.1.2. Análisis del Entrenamiento y Validación del Diseño B	83
4.3.1.3. Análisis del Entrenamiento y Validación del Diseño C	84
4.3.1.4. Análisis del Entrenamiento y Validación del Diseño D	84
4.3.1.5. Análisis del Entrenamiento y Validación del Diseño E	85
4.3.2. Señales de Tránsito de Perú	86
4.3.2.1. Análisis del Entrenamiento y Validación del Diseño A	86
4.3.2.2. Análisis del Entrenamiento y Validación del Diseño B	87
4.3.2.3. Análisis del Entrenamiento y Validación del Diseño C	88
4.3.2.4. Análisis del Entrenamiento y Validación del Diseño D	89
4.3.2.5. Análisis del Entrenamiento y Validación del Diseño E	89
4.4. Resultados de Evaluación	90
4.4.1. Señales de Tránsito de Alemania	92
4.4.1.1. Análisis del Diseño A	93
4.4.2. Señales de Tránsito de Perú	94
4.4.2.1. Análisis del Diseño A	94
5. Resultados de la tesis	23
5.1. Teóricos	23
5.2. Computacionales	23
6. Consideraciones finales	24

6.1. Conclusiones	24
6.2. Trabajos futuros	25
A. Primer apendice	31
B. Segundo apendice	32
C. Tercer apendice	33

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 05 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 de los vehículos, (OMS, 2017).



Figura 1.1: Señalización de velocidad

Fuente: (OMS, 2017)

Por otra parte, de los vehículos que se venden 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 local, 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 America 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,(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 multicategórica que comúnmente presenta desigualdades en las frecuencias de aparición de las 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 el clasificador que 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. Antecedentes de la Investigación

En este capítulo se presentan 7 trabajos previos con relación a al tema abordado en la investigación y que sirven de base para este.

1.1.1. INTERNACIONALES:

(Vicen-Bueno et al., 2007) en su paper "Traffic Sign Classification by Image Preprocessing and Neural Networks", propusieron una investigación que se centra en clasificar 9 tipos de señales de tráfico de circulación(color azul) de España utilizando una combinación de diferentes

técnicas de preprocessamiento 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 preprocessamiento 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 paper "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 tasas de error mejores que las anteriores técnicas de 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 preprocesamiento 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, a parte 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 %.

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, de los cuales el mejor resultado logró un acierto de 95.29 % cerca al estado del arte(99.46 %), en un total de 12630 señales evaluadas. Lo importante del antecedente es que servirá como elemento de comparación para la investigación propuesta.

1.1.3. ESTADO DEL ARTE:

Para clasificación de señales de tránsito se han realizado diferentes estudios de la base de datos GTSRB (German Traffic Sign Benchmark), que cuenta con más de 50 mil imágenes a color distribuidas en 43 clases.



Figura 1.4: Algunas muestras de la base de datos GTSRB

Fuente: (Stallkamp et al., 2012)

Estos estudios usan diferentes métodos como Clasificación basado 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 una red neuronal convolucional (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. Las imágenes se complementan con varios conjuntos de características precalculadas para permitir, de ser necesario, la aplicación de algoritmos de aprendizaje automático sin conocimientos básicos en el procesamiento de imágenes.

(Cireşan et al., 2012), utilizó una arquitectura de 9 capas que consistía en una capa de entrada, 3 capas convolucionales usando 100, 150 y 250 filtros cuyos tamaños por cada capa fueron de 42x42, 18x18 y 6x6 neuronas respectivamente, 3 capas pooling con estructuras parecidas a las convolucionales y 1 capa totalmente conectada compuesta por 300 neuronas de entrada para las 43 neuronas de salida que representaban a cada clase.

Los resultados sobre la base de datos GTSRB fueron realizados a finales del año 2011 produciendo un acierto del 99.46 % para el conjunto de imágenes evaluadas, (Stallkamp et al., 2012);

sin embargo, desde el año 2012 se han propuesto nuevas variantes de redes neuronales convolucionales, ya sea usando una funciones de activación diferentes, usando más capas en la red, mas filtros o introduciendo nuevos métodos de optimización.

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. Importancia de la investigación

1.4.1. 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 como lo es el reconocimiento de señales de tránsito vehicular desde la perspectiva de un aprendizaje profundo eficiente. Sin embargo realizar esto no es tarea simple, ya que se requiere de un modelo de reconocimiento que funcione para imágenes que se encuen-

tran 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.

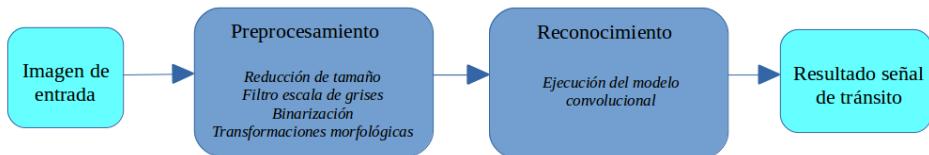


Figura 1.5: Diagrama de bloques del modelo preentendido
Fuente propia

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, cálculo matemático 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 reconocer de señales de tránsito vehicular.

1.4.2. Justificación Social

Teniendo conocimiento de lo descrito en la realidad problemática, 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 constantes infracciones y muertes se puedan evitar y en consecuencia reducir estos índices progresivamente. Por ejemplo, el recono-

cimiento 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. Adicionalmente, podría ser un componente útil en cámaras de video vigilancia convirtiéndolas en más inteligentes con la capacidad de monitorear que las señales de tránsito sean respetadas.

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

1.5. Objetivos

1.5.1. 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.

1.5.2. Objetivos específicos

- a) Obtener un conjunto de imágenes(dataset) donde se muestren diferentes señales de tránsito vehicular.
- b) Dividir un conjunto de imágenes 2 grupos, uno para el entrenamiento y otro para evaluación y analizar el dataset de entrenamiento a través de métodos de procesamiento de imágenes. De ser necesario, con estos mismos, aumentar la cantidad de imágenes.
- c) Implementar diferentes arquitecturas de redes convolucionales profundas en las cuales el dataset de entrenamiento será procesado.
- d) Experimentar el uso de diversas funciones de activación, funciones de costo, ajuste de hiperparámetros y métodos de optimización para dichas arquitecturas.
- e) Evaluar individualmente el rendimiento que se obtiene de las arquitecturas implementadas en el dataset de entrenamiento y evaluación.
- f) Elaborar un modelo computacional basado en las evaluaciones realizadas.

1.6. Contribución de la investigación

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 para organismos internacionales y 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 de la sociedad.

1.7. Metodología 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 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 com tasa de sensibilidad y de error, se pretende usar una Matriz de confusión, en la cual cada fila de la matriz representa las instancias en una clase predecida, mientras que cada columna representa las instancias en una clase real (o viceversa). A través de esta matriz se pueden conocer valores estadísticos del espacio ROC(Receiver Operating Characteristic), métrica usada

para evaluar la calidad del modelo clasificador(reconocedor).

1.7.1. 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 Figura muestra las diferentes fases de la metodología.

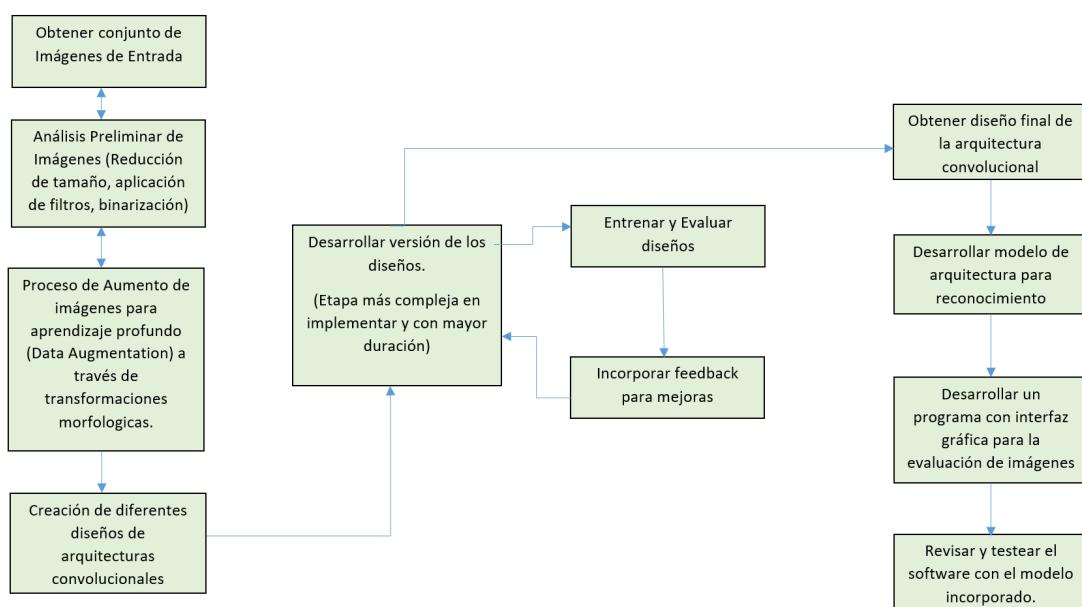


Figura 1.6: Diseño del Modelo del Ciclo de Vida del Desarrollo

Fuente propia

1.8. Estructura de la tesis

El presente trabajo está dividido en seis capítulos. El primer capítulo presenta los aspectos generales del tema tratado: la formulación del problema, importancia de la investigación, los objetivos, la contribución, además de la metodología de la investigación y la estructura de la tesis.

En el capítulo dos se presenta el referencial teórico, soporte del tema, contemplando los con-

ceptos de inteligencia artificial, aprendizaje automático basado en redes renuronales detallando y diferenciando sus características en el área de aprendizaje profundo. Además se analiza las características y procesos de una red convolucional a detalle.

El tercer capítulo trata sobre los materiales y las técnicas de recolección de datos

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

En el quinto capítulo se presentan los resultados obtenidos en la investigación.

En el capítulo seis 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, en el séptimo y octavo capítulo se describen las referencias bibliográficas usadas para la investigación en esta tesis y los anexos donde se presentan los programas elaborados y en apéndice un pequeño glosario de ciertos términos usados en esta investigación.

Capítulo 2

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 optimización combinatoria, complejidad computacional, metaheurísticas, ciencia de la información y logística, conocimientos sin los cuales sería difícil de modelar y solucionar matemática y computacionalmente cualquier tipo de problema de optimización.

2.1. 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.

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 permitir 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 direccionado 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" porque 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 X 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.

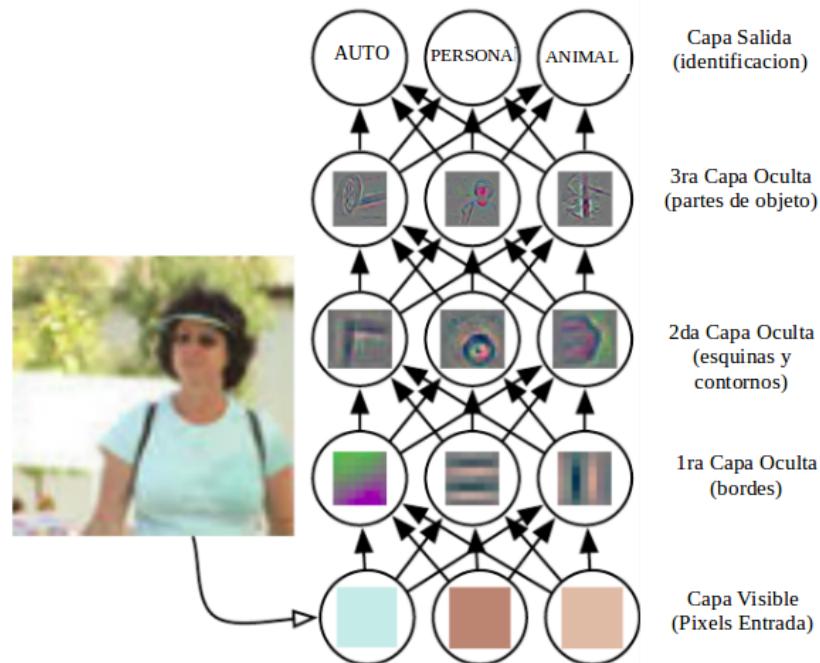


Figura 2.1: Ilustración de un modelo de aprendizaje profundo

Fuente propia

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 la arquitectura. 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 una arquitectura, 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. 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. 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 con conceptos más simples y representaciones más abstractas calculadas en base a representaciones menos abstractas. La siguiente 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.

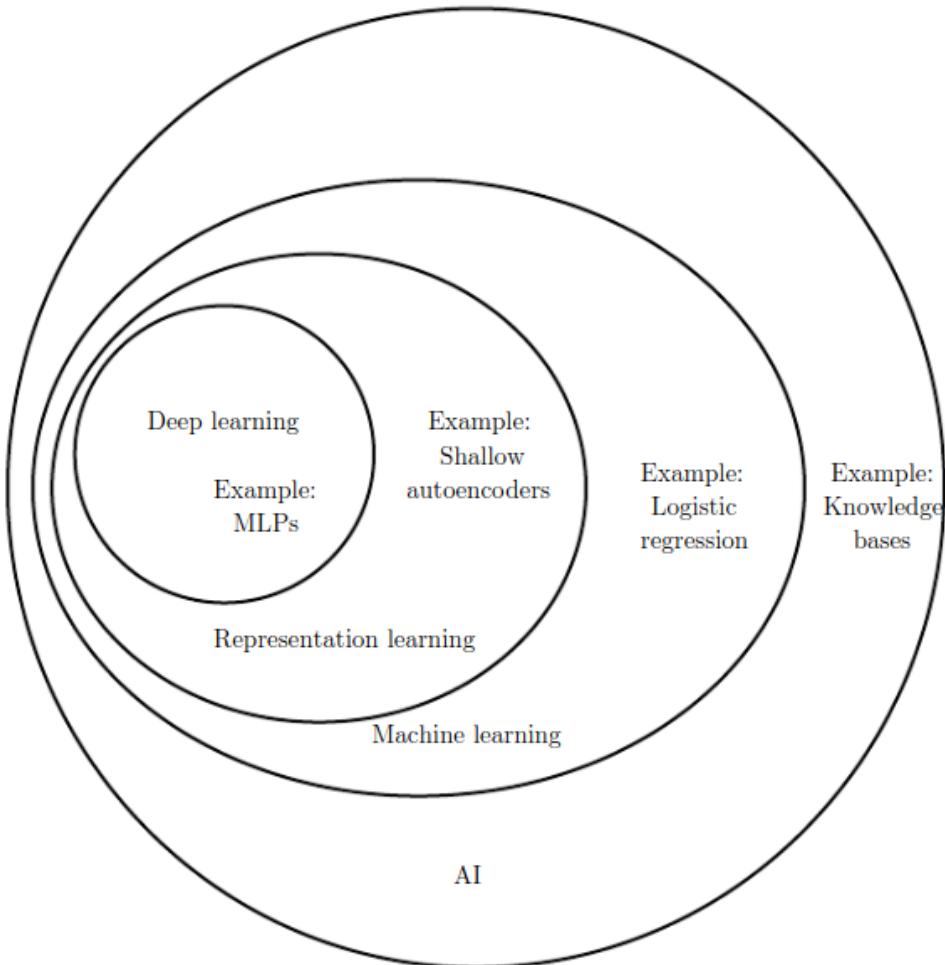


Figura 2.2: Diagrama de Venn donde cada sección incluye un ejemplo de una tecnología de IA(ilustra la relación entre estas diferentes disciplinas de la IA)

Fuente: (Goodfellow et al., 2016)

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

Las arquitecturas 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 de la arquitectura 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. Cada neurona recibe algunas entradas, realiza operaciones matemáticas. Toda la red expresa una única función de puntuación diferiable: 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).

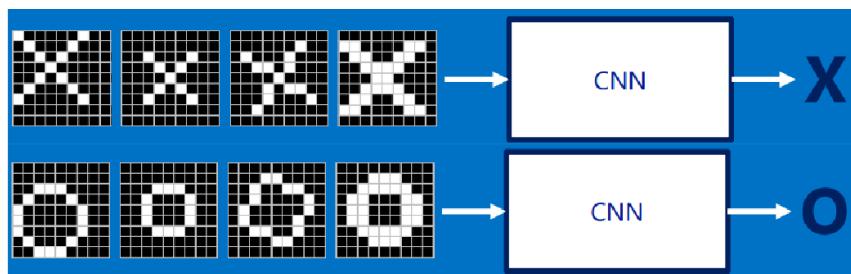


Figura 2.3: Entrada y salida de una CNN

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 hiper-literal, es decir, que coincida pixel a pixel para que se trate de imágenes iguales.

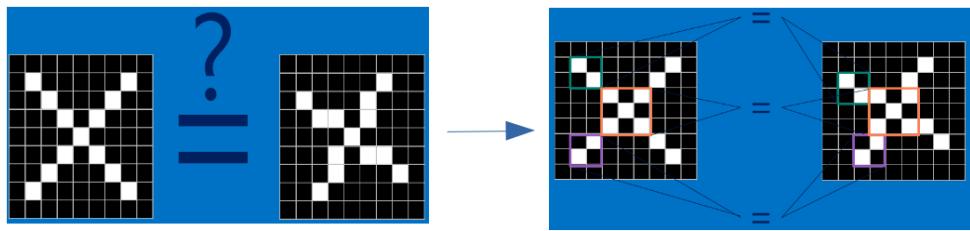


Figura 2.4: Análisis de una CNN

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 una función diferenciable, 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.

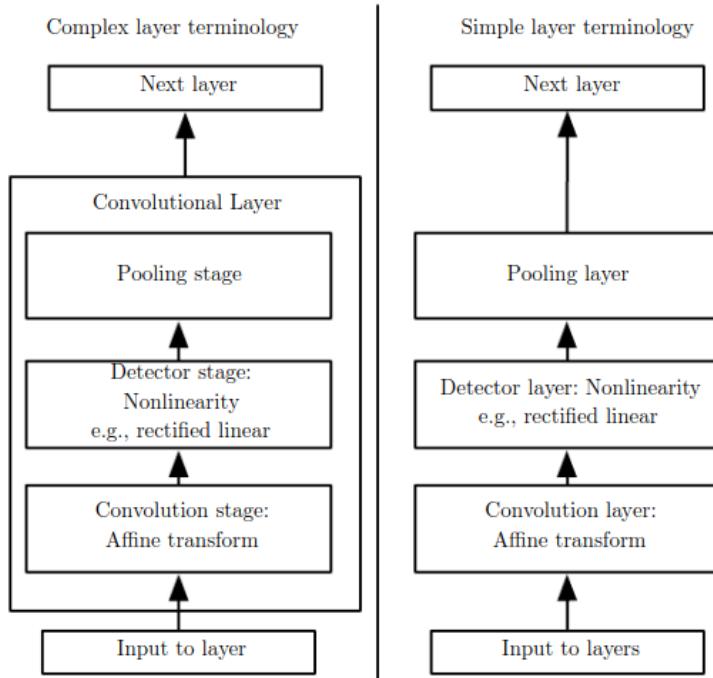


Figura 2.5: Terminología de capas complejas(izquierda) y de capas simples(derecha)

Fuente: (Goodfellow et al., 2016)

2.2.1. Capa Convolucional

Los parámetros de la capa convolucional consisten basicamente 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.

2.2.1.1. 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, convolvió) 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.

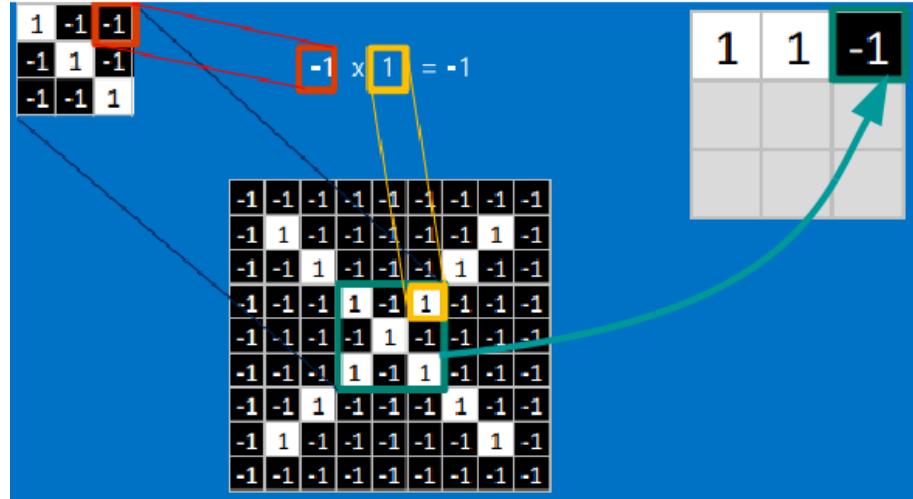


Figura 2.6: Convolucion entre el filtro y parte de la imagen

Fuente: (Rohrer, 2016)

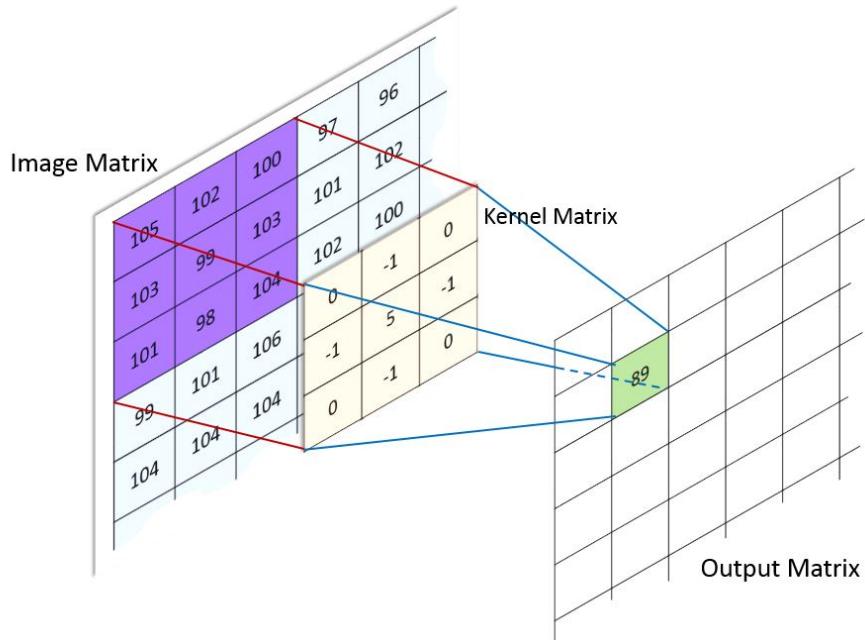


Figura 2.7: Posicionamiento del kernel/filtro por pixel

Fuente: Fuente 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.

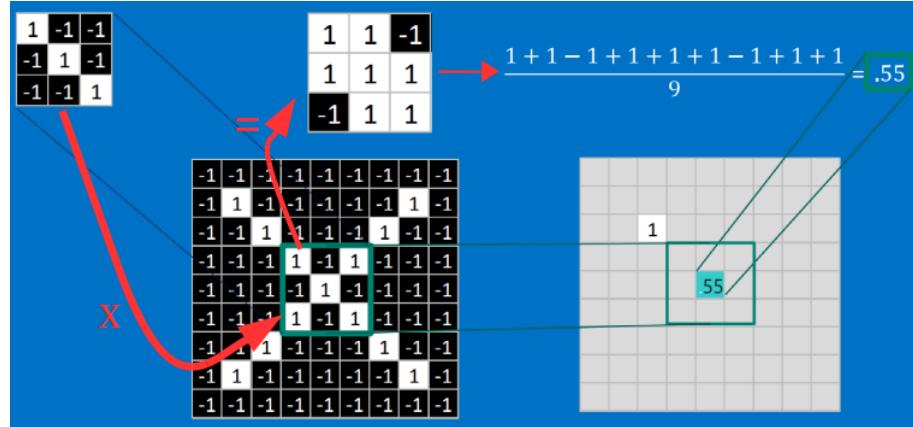


Figura 2.8: Proceso matemático convolucional

Fuente:(Rohrer, 2016)

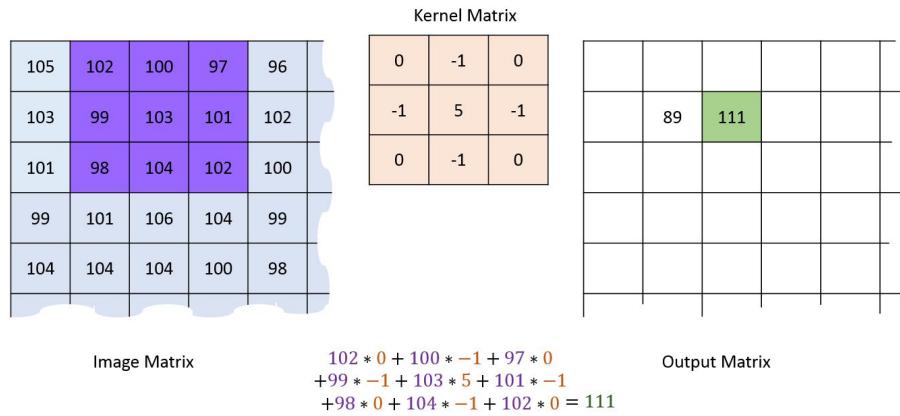


Figura 2.9: Cálculo Convolucional

Fuente: Fuente 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 repre-**

sentaciones 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). Esto se logra haciendo que el kernel sea 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. Esto significa que necesitamos almacenar menos parámetros, lo que reduce los requisitos de memoria del modelo y mejora su eficiencia estadística. También significa que el cálculo de la salida requiere menos operaciones.

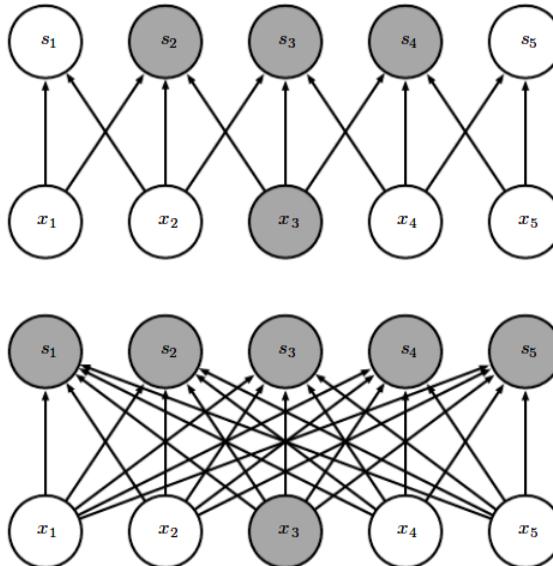


Figura 2.10: Conectividad dispersa

Fuente:(Rohrer, 2016)

Estas mejoras en la eficiencia suelen ser bastante grandes. Si hay entradas y salidas, la multiplicación de la matriz requiere $m \times n$ parámetros, y los algoritmos utilizados en la práctica

por ejemplo tienen complejidad de tiempo de ejecución $O(m \times n)$. Si limitamos el número de conexiones que cada salida puede tener a k , entonces el enfoque dispersamente conectado requiere solo $k \times n$ parámetros y $O(k \times n)$ en tiempo de ejecución. Para muchas aplicaciones prácticas, es posible 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 X, 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 .

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. Entonces, 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.

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 compartir parámetros, se puede decir que una red hastió pesas, porque el valor del peso aplicado a una entrada está vinculado al valor de una ponderación aplicada en donde. En una red neuronal

convolucional, cada miembro del kernel se utiliza en cada posición de la entrada (excepto tal vez algunos de los píxeles del límite, 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 un conjunto separado de parámetros para cada ubicación, aprendemos solo un conjunto.

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.

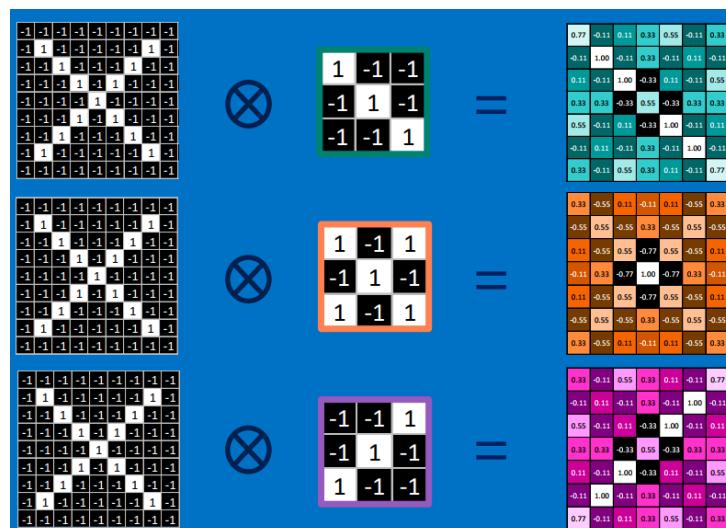


Figura 2.11: Resultado de Convolución(conjunto de mapas de activación. generado a partir de 3 filtros para 3 características: diagonal derecha, cruzamiento central y diagonal izquierda). Simbolo de convolución: \otimes

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 bidi-mensional 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 14.

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.

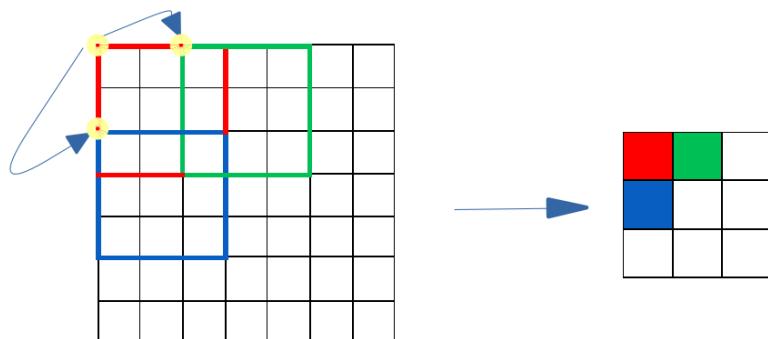


Figura 2.12: Imagen con stride igual a 2, para el filtro tanto en largura como anchura

Fuente propia

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

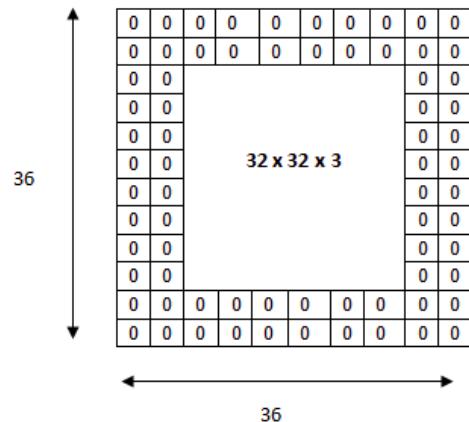


Figura 2.13: Ejemplo de zero-padding con tamaño 2

Fuente 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:

$$\text{conv}_j^n = \sum_{k=1}^K x_k^n \times w_{kj}^n + b_n$$

En el que:

- x, w, b son valores de entrada, pesos y biases(sesgos), 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

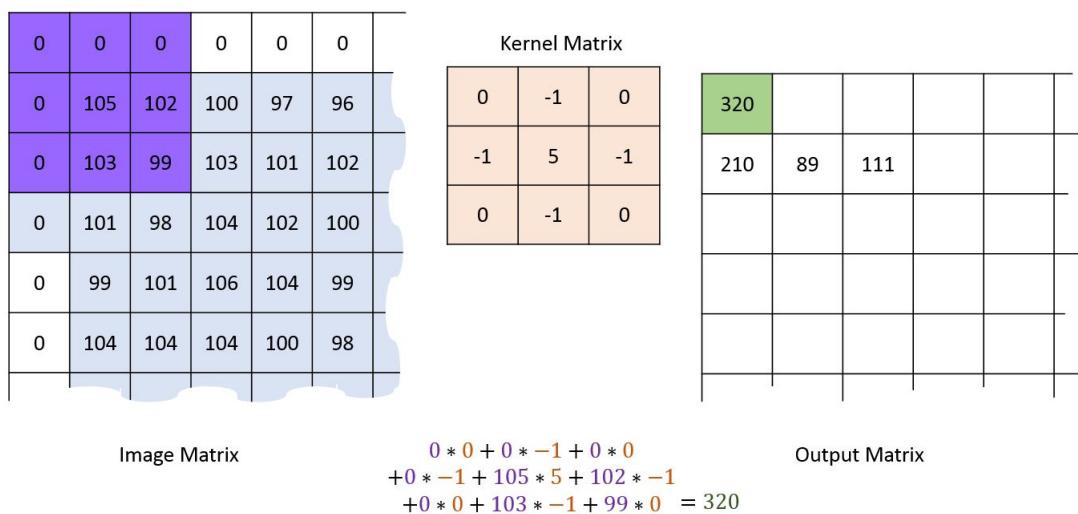


Figura 2.14: Ejemplo de filtro crado apartir de los 3 aspectos mencionados

Fuente propia

2.2.1.2. 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. Al apilar 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)

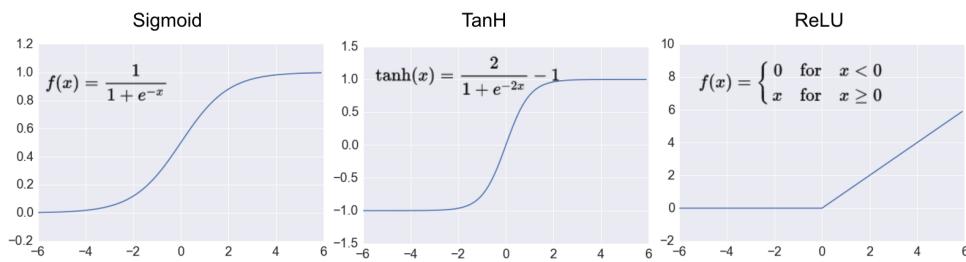


Figura 2.15: Funciones de activación

Fuente 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 conv. 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).

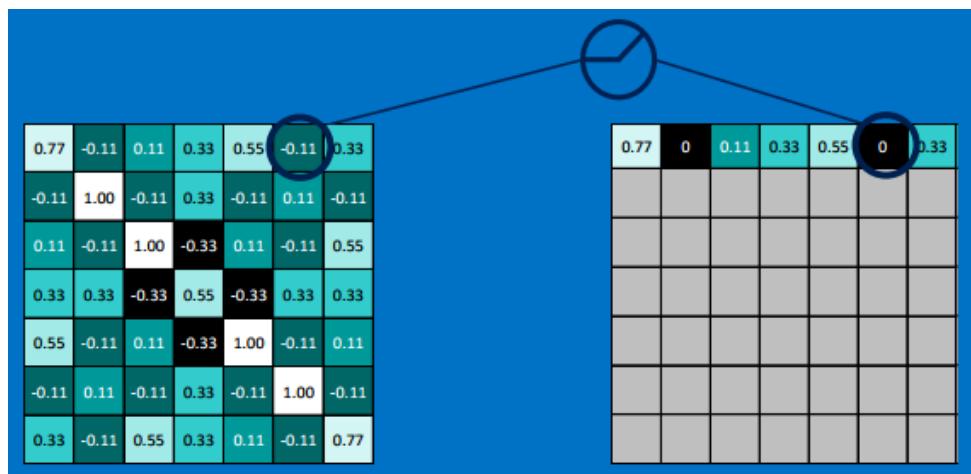


Figura 2.16: Procedimiento de la función ReLU

Fuente: (Rohrer, 2016)

Es por ello que entre la capa(etapa) de convolucion y la capa(etapa) de pooling puede encontrarse la capa ReLU que esta 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 tambien 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.

2.2.1.3. 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 de reducir el sobreajuste que puede darse durante el entrenamiento de una red neuronal. Consiste en establecer a cero la salida de cada neurona oculta con una probabilidad 0.5(comúnmente). Las neuronas que se .^abandonan"de esta manera no contribuyen al pase directo y no participan en las siguientes etapas de entrenamiento.

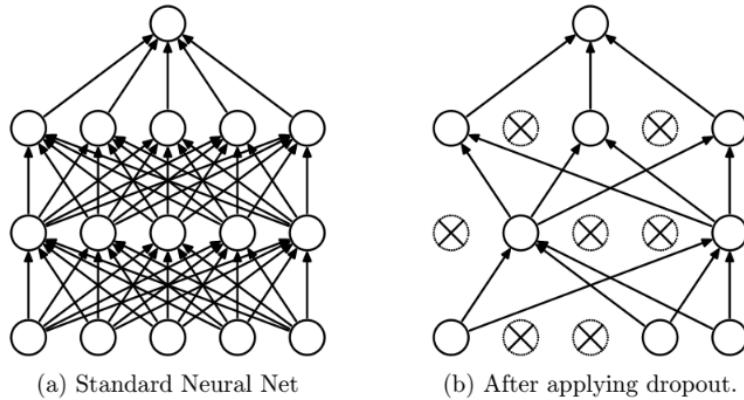


Figura 2.17: En la izquierda la red neuronal común y a la derecha la red neuronal diluida producida por la aplicación de dropout

(Romero, 2015a)

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

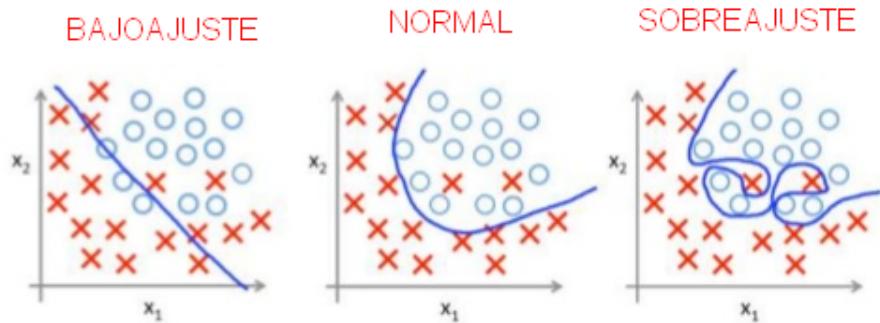


Figura 2.18: Procesos que pueden ocurrir durante entrenamiento. Dropout evita el sobreajuste

Fuente propia

2.2.1.4. Capa de Agrupación(Pooling)

Una variedad de cálculos que reducen la dimensionalidad de un mapa de características se conocen como agrupación. La agrupación, que se aplica a cada canal por separado, permite que la red sea robusta e invariante 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 en un menor número de valores. Puede ser configurado en función del tamaño de su campo receptivo (por ejemplo, 2×2) y en función a la operación de agrupamiento (por ejemplo, máximo-max o promedio-average), como se muestra en Fig. 20. Normalmente, la agrupación se produce en bloques que no se solapan (es decir, el paso o stride es igual al tamaño de la agrupación). 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

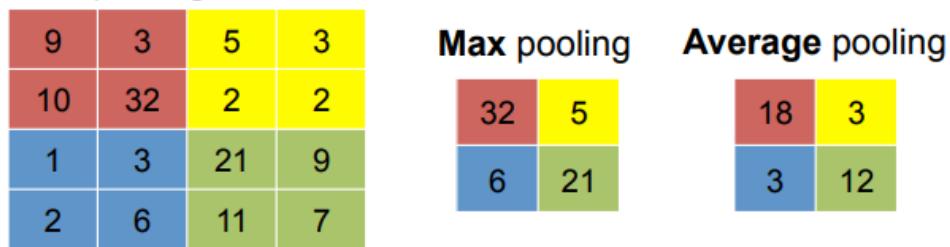


Figura 2.19: Formas de agrupamiento(pooling)

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 el cálculo en la red. El segundo es que controlará el sobreajuste.

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) através de una imagen y tomar el valor máximo de la ventana en cada paso.

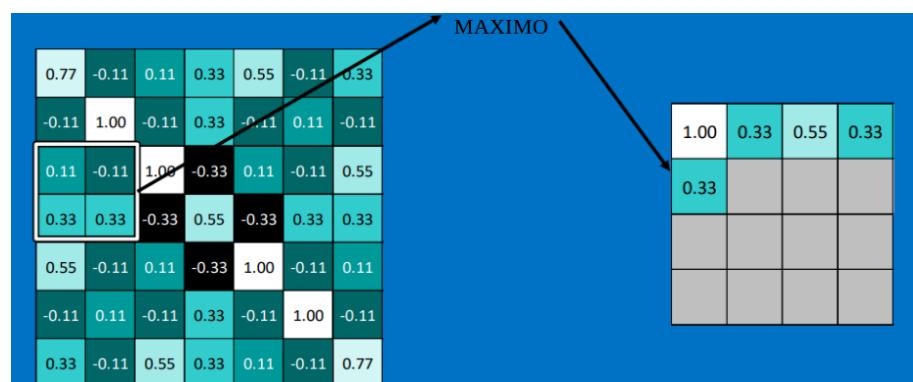


Figura 2.20: Operación MAX-pooling en capa de Agrupación

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 un versión agrupada de las imágenes de entrada.



Figura 2.21: Resultado de Agrupación

Fuente: (Rohrer, 2016)

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

Y produce un dato de salida con los siguiente 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,

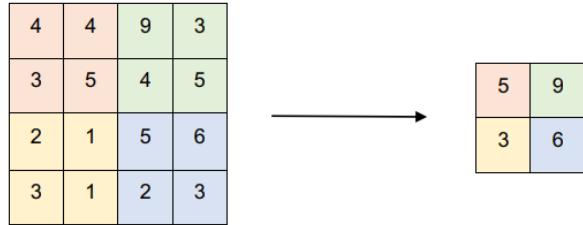


Figura 2.22: Max-pooling con filtro 2x2 y paso 2

Fuente propia

2.2.2. Capa totalmente conectada (Fully-connected layer)

Esta capa por lo general aparece al final de la arquitectura 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.

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

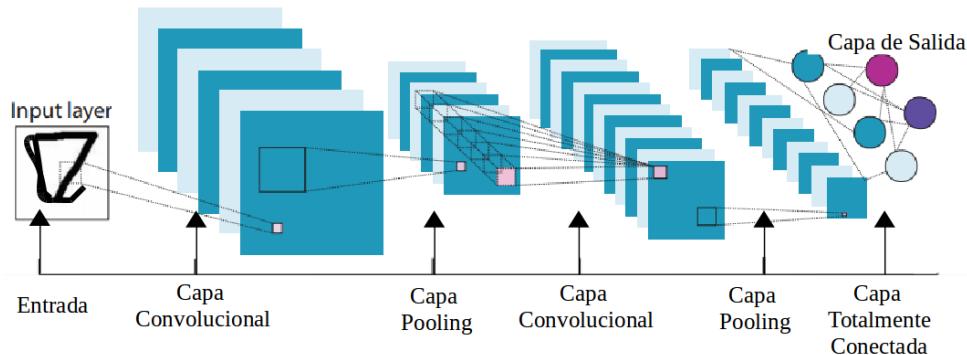


Figura 2.23: Modelo de red neuronal convolucional profunda

Fuente propia

- Se pasa una imagen de entrada a la primera capa convolucional. La salida complicada 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.
- Cada filtro dará una característica diferente para ayudar a la predicción de clase correcta. 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 predicción.
- A medida que profundizamos en la red, se extraen características más específicas en comparación con una red superficial 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 la capa de salida 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 varios iteraciones para obtener mejores resultados.

2.3. Entrenamiento y Validación

La red procesa los registros en los datos de entrenamiento uno a la vez, usando los pesos 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(refianan)continuamente. Para realizar este proceso, existe una método de optimización muy popular denominado **Descenso de gradiente**(Gradient Descent).

2.3.1. 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.

der. 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)

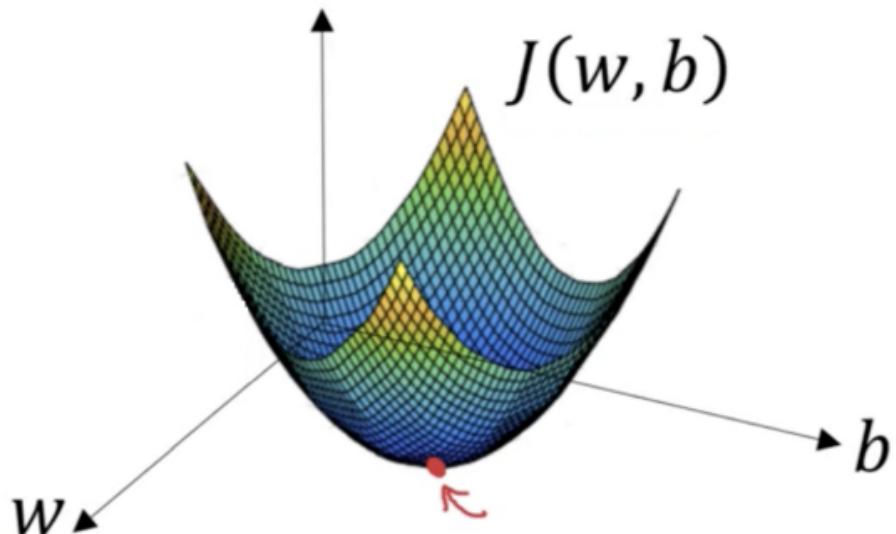


Figura 2.24: Establecimiento de la Tasa de Aprendizaje

(Donges N., 2018)

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

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

La derivada del error con respecto al peso

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

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

Representación de la distancia por la dirección del cambio. η es 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.

Se tienen diversos tipos, estos son:

2.3.1.1. 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.

2.3.1.2. 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.

2.3.1.3. Mini-batch Gradient Descent

Es el método de 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.

2.3.2. 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, por ejemplo, el tamaño del paso en el que el gradiente cae en la dirección máxima de la pendiente, (Jeremy J., 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.

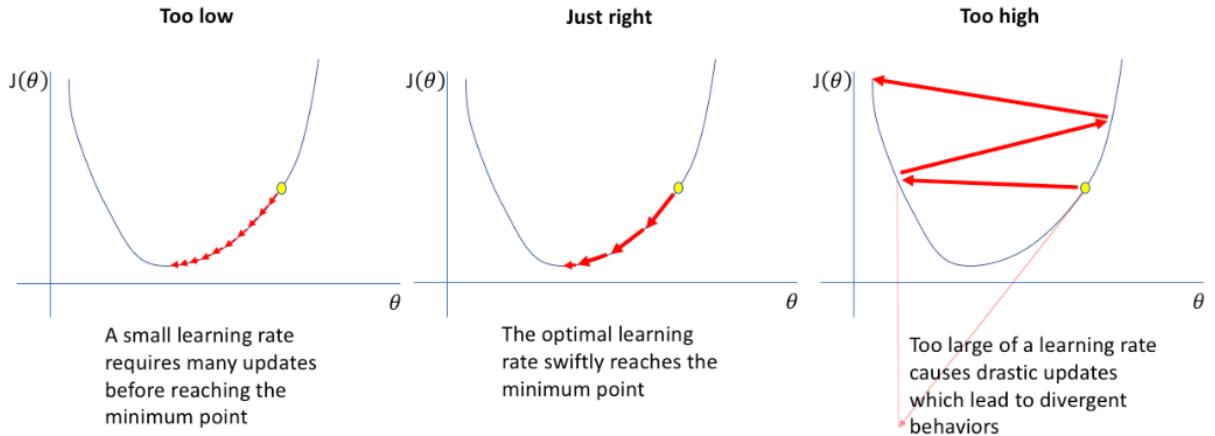


Figura 2.25: Establecimiento de la Tasa de Aprendizaje

(Jeremy J., 2018)

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 apartir del cual la pérdida ya no se reduce y en lugar de eso comienza a incrementarse).

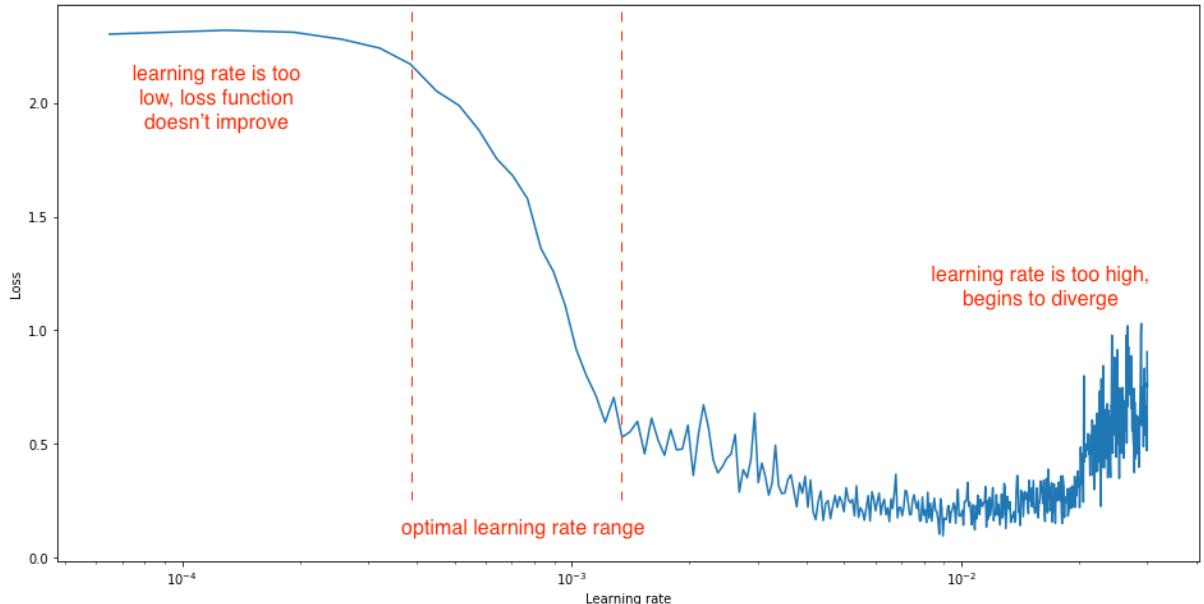


Figura 2.26: Problemas con la Tasa de Aprendizaje

(Jeremy J., 2018)

Reduciendo lentamente la tasa de aprendizaje a lo largo del tiempo ayuda a acelerar el apren-

dizaje. Esto es lo que se conoce como tasa de decaimiento de aprendizaje(**learning rate decay**).

entrenamiento/learning_rate_decay

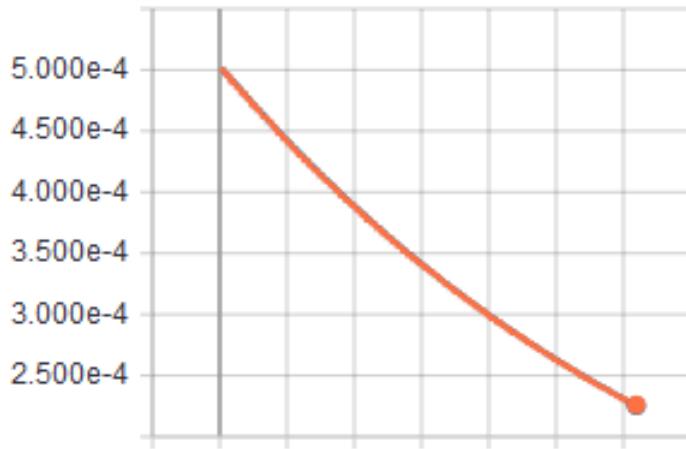


Figura 2.27: Ejemplo de Tasa de decaimiento de aprendizaje por época

Fuente 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; algunos de ellos son: Adagrad, Adadelta, RMSprop, Adam.

En esta investigación se usaran los métodos de Descenso de Gradiente Mini-batch y optimizador Adam, conjuntamente con una tasa de decaimiento de aprendizaje.

2.3.3. 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. Realiza actualizaciones más grandes para parámetros infrecuentes y actualizaciones más pequeñas para los frecuentes. Debido a esto, es adecuado para datos dispersos (NLP o reconocimiento de imágenes). Otra ventaja es que básicamente simplifica la necesidad de ajustar la tasa de aprendizaje. 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 .

2.3.4. 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 75 % 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)

Se utiliza el subconjunto de validación para evaluar el desempeño de diferentes arquitecturas

del modelo (diferentes topologías), para luego escoger una de ellas. Este subconjunto(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 indique 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 multiclas 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 la función de coste/pérdida dada por la validacion 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 regresion. 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, convirtiendo el aprendizaje más rápido,(Romero, 2015a).

La pérdida de entropía cruzada aumenta 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).

2.3.5. Función Softmax

Conocida tambien como función exponencial normalizada, es la funcion 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

obternela. 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 el resultado 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 $x \mapsto \mathbf{x}^\top \mathbf{w}_1, \dots, \mathbf{x} \mapsto \mathbf{x}^\top \mathbf{w}_K$, 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.

2.3.6. 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.

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(lambda) a la función de pérdida, la cual penaliza parametrizaciones con pesos más elevados,(Romero, 2015b), 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.

Si $L(\theta, D)$ es la función de pérdida (en esta investigación - entropía cruzada), 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 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 $L(\theta, D)$ (cross entropy), sino que optimizamos $L(\theta, D) + \lambda R(\theta)$

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

Capítulo 3

Materiales y Técnicas

3.1. 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.

3.2. Variables de la Investigación

3.2.1. Variable Dependiente

Reconocimiento automático de señales de tránsito vehicular

3.2.2. Variable Independiente

Modelo basado en el aprendizaje profundo de redes neuronales convolucionales

3.3. Indicadores

Los indicadores nos permiten realizar mediciones y a su vez determinan la validez de la hipótesis planteada en la presente investigación. Se han considerado tres indicadores:

Tabla 3.1: Indicadores para la investigación

<u>Indicador</u>	<u>Descripción</u>	<u>Instrumento</u>
Tasa de Sensibilidad (Efectividad)	$S = \frac{\text{Verdaderos Positivos}}{\text{Verdaderos Positivos} + \text{Falsos Negativos}} \times 100$	Modelo de Reconocimiento
Tasa de Error (Especificidad)	$S = \frac{\text{Falsos Negativos}}{\text{Verdaderos Positivos} + \text{Falsos Negativos}} \times 100$	Modelo de Reconocimiento
El espacio ROC (Curvas ROC)	Relación entre efectividad y especificidad	Modelo de Reconocimiento

3.4. Recolección de Datos para la Construcción del Modelo

3.4.1. Técnica de Recolección

Revisión de la literatura(análisis de documentos)

3.4.2. 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.

3.4.3. Muestra

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

3.4.3.1. 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 lineal conocida como RELU. Similar a LenNet-5(Y. LeCun, 1998) pero más grande, teniendo aproximadamente 60 millones de parámetros.

3.4.3.2. 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 por un cuidadoso que permite aumentar 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).

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

3.5.1. Técnica de Recolección

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

3.5.2. 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 tomaran en cuenta imágenes donde se muestren señales de tránsito vehicular del tipo vertical en sus 3 subcategorías.

3.5.3. 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.

3.5.3.1. Señales de Tránsito de Alemania

(Stallkamp et al., 2011) 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.

3.5.3.2. 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)

Capítulo 4

Desarrollo de la Investigación

4.1. Análisis del conjunto de Imágenes

4.1.1. Datos Iniciales para el entrenamiento

4.1.1.1. 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 4.1: Speed limit (20km/h)



Figura 4.2: Speed limit (50km/h)



Figura 4.3: Bumpy road



Figura 4.4: Children crossing



Figura 4.5: Wild animals crossing



Figura 4.6: Turn left ahead

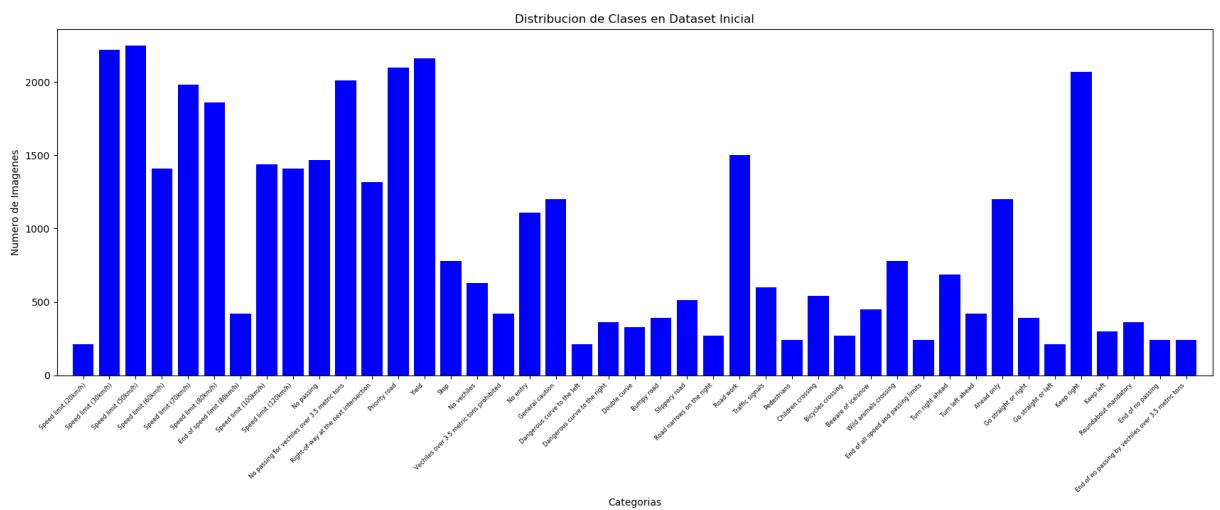


Figura 4.7: Distribución de ejemplos por señal para el entrenamiento(Total 39209)

Fuente propia

4.1.1.2. 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 pixeles.



Figura 4.8: Pare



Figura 4.9: Resalto



Figura 4.10: Zona Escolar



Figura 4.11: Peatones



Figura 4.12: Paradero



Figura 4.13: No Estacionar



Figura 4.14: Disminuir Velocidad

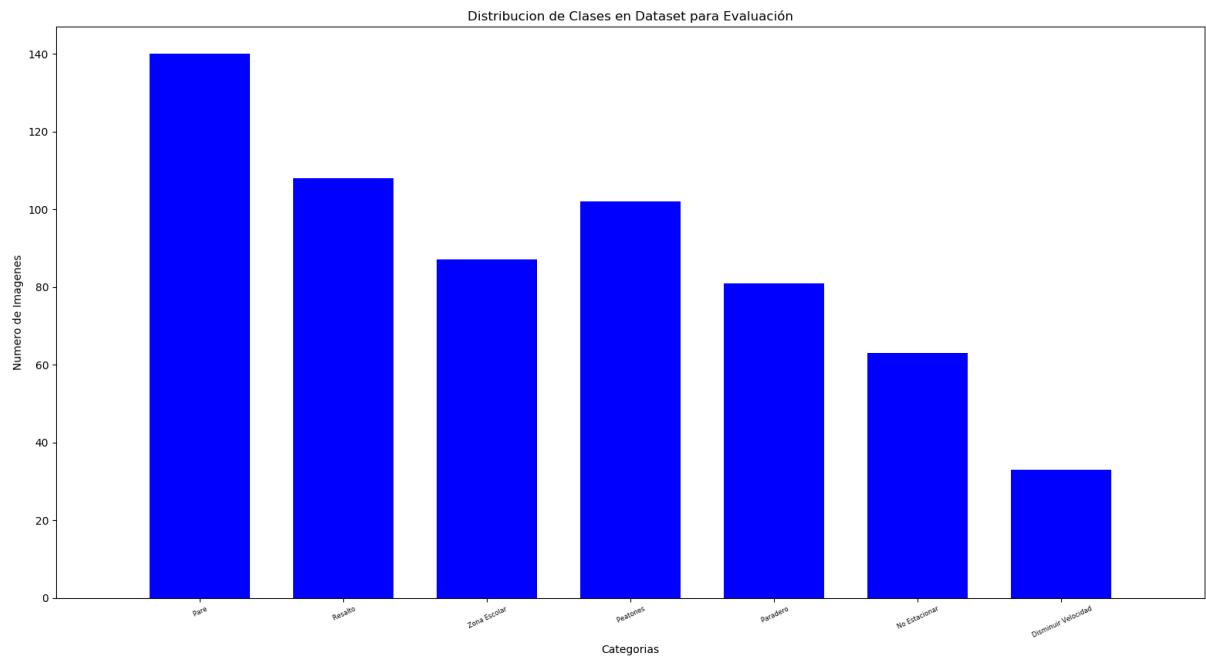


Figura 4.15: Distribución de ejemplos por señal para el entrenamiento(Total 614)

Fuente propia

4.1.2. Datos para el evaluación

4.1.2.1. 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.

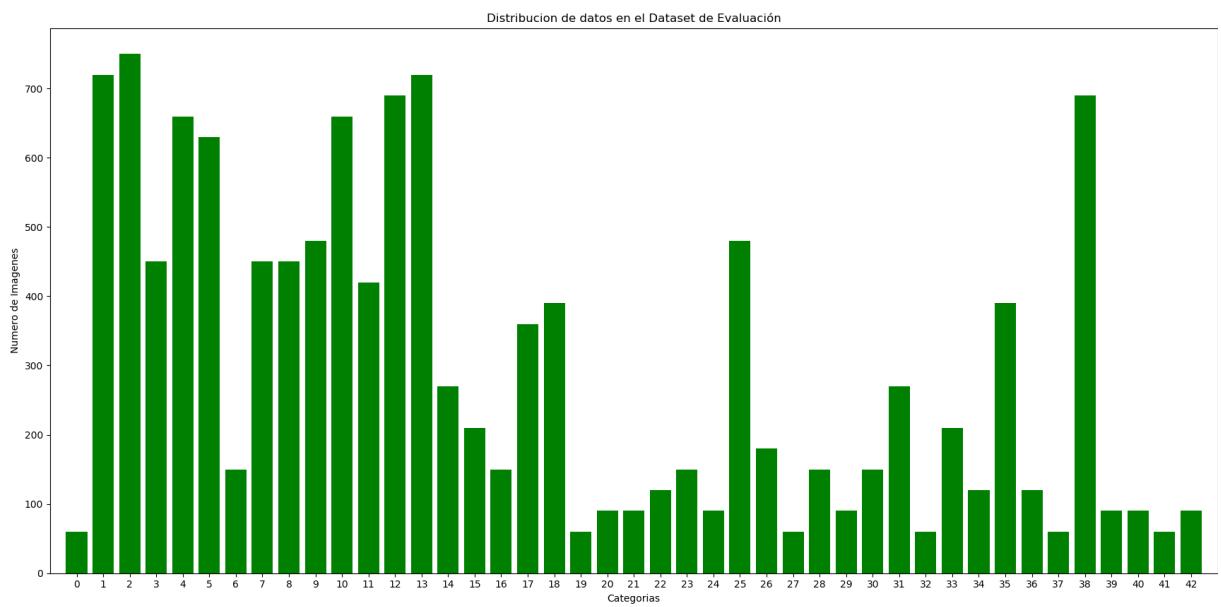


Figura 4.16: Distribución de ejemplos por señal para la evaluación - Alemania

Fuente propia

4.1.2.2. Señales de Tránsito de Perú

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.

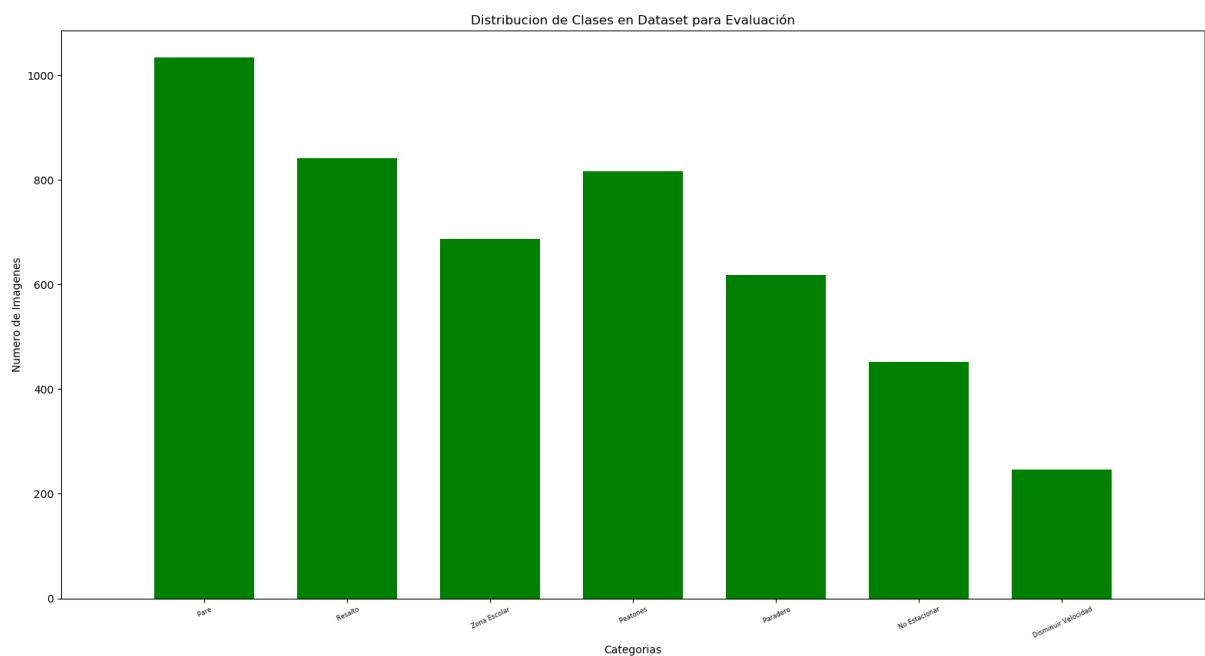


Figura 4.17: Distribución de ejemplos por señal para la evaluación - Perú

Fuente propia

4.1.3. 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).

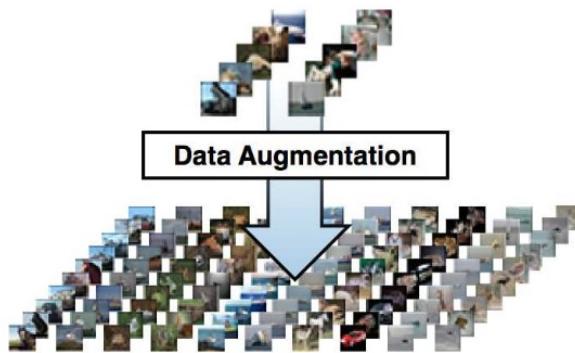


Figura 4.18: El aumento de datos infla artificialmente los conjuntos de datos usando transformaciones que preservan las categorías de los objetos

(Taylor and Nitschke, 2017)

Son utilizadas las siguientes técnicas:

4.1.3.1. 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:



Flipping Vertical:



Figura 4.19: Imágenes volteadas horizontalmente

Fuente propia

Figura 4.20: Imágenes volteadas verticalmente

Fuente propia

Flipping Horizontal y Vertical:



Figura 4.21: Imágenes volteadas primero horizontal y luego verticalmente

Fuente propia

Incluso, hay signos que luego de voltearse, 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.

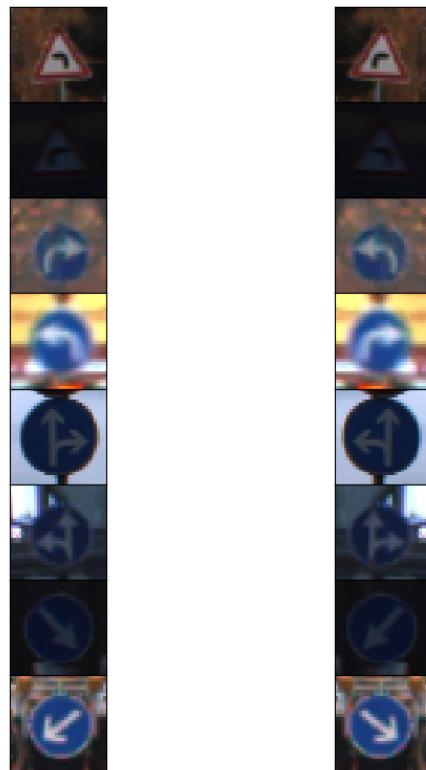


Figura 4.22: Imágenes que volteadas horizontal o verticalmente, cambian su categoría

Fuente propia

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

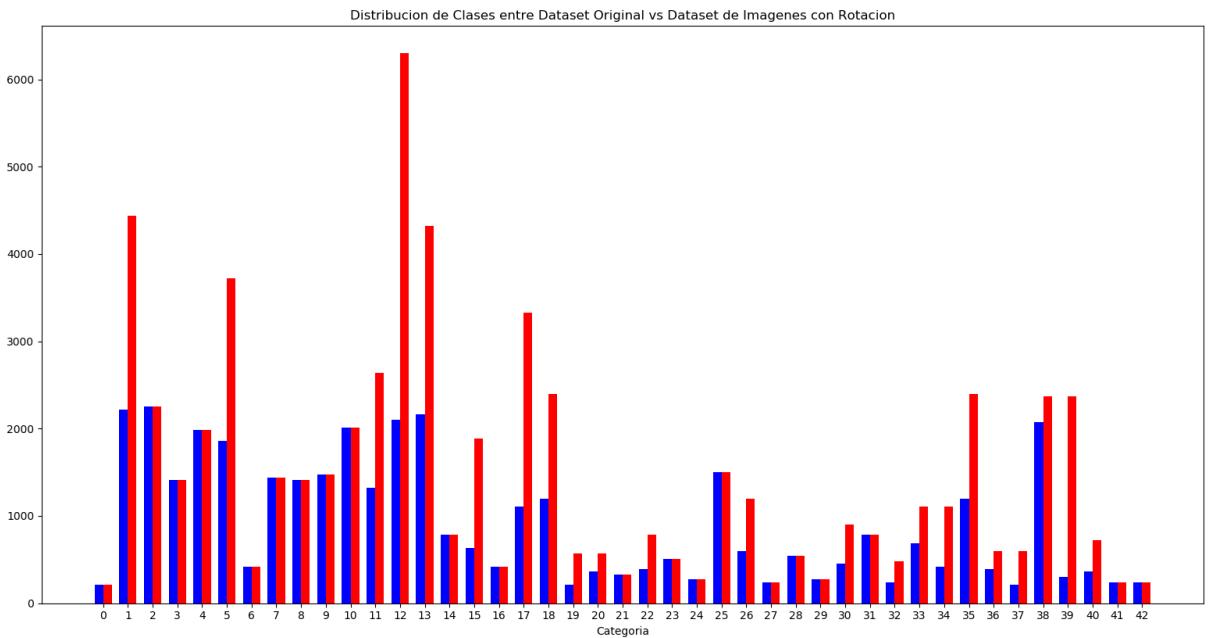


Figura 4.23: Distribución de categoría, luego de aplicar el Flip(en sus distintos tipos)- Señales de Alemania

Fuente propia

Las siguientes técnicas fueron utilizadas para el dataset de señales de Tránsito de Alemania y de Perú.

4.1.3.2. 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. Esto obliga 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 parece también ocuparse de escalar al azar a medida que colocamos al azar las esquinas de la imagen en un rango [-delta, +delta].

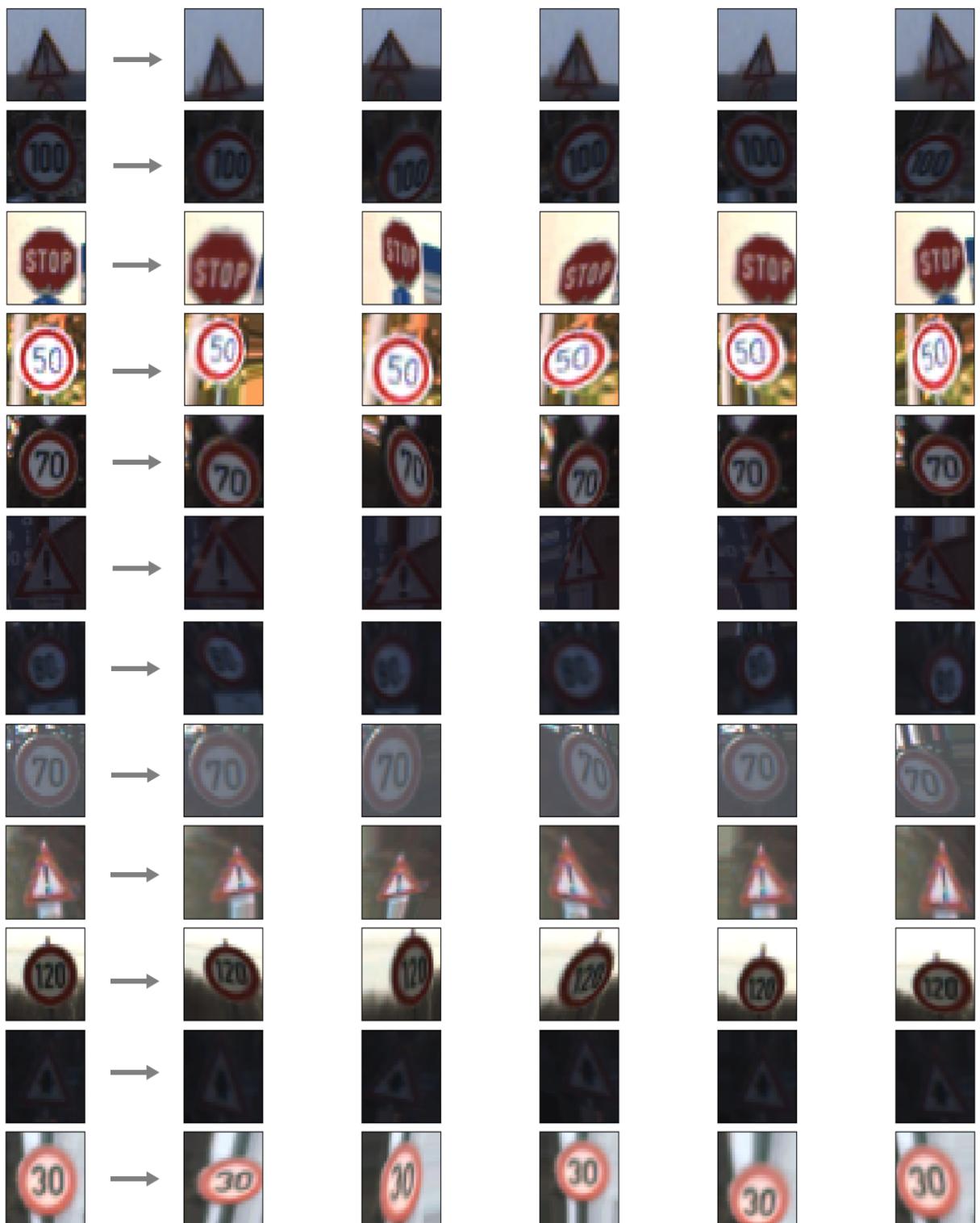


Figura 4.24: Ejemplo de cinco proyecciones por cada imagen - Dataset Alemania

Fuente propia

4.1.3.3. 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.

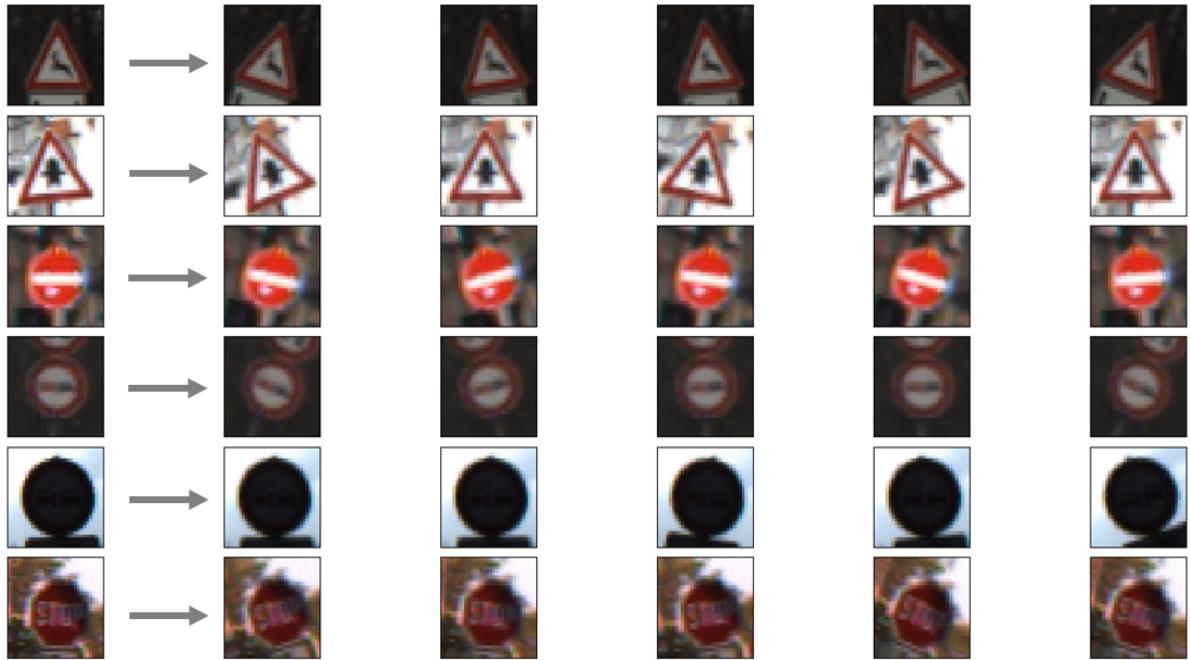


Figura 4.25: Ejemplo de cinco rotaciones por cada imagen - Dataset Alemania

Fuente propia

4.1.3.4. Zoom

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



Figura 4.26: Ejemplo de cinco aplicaciones de zoom(in/out) por cada imagen - Dataset Alemania

Fuente propia

4.1.3.5. Equalizacion 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.

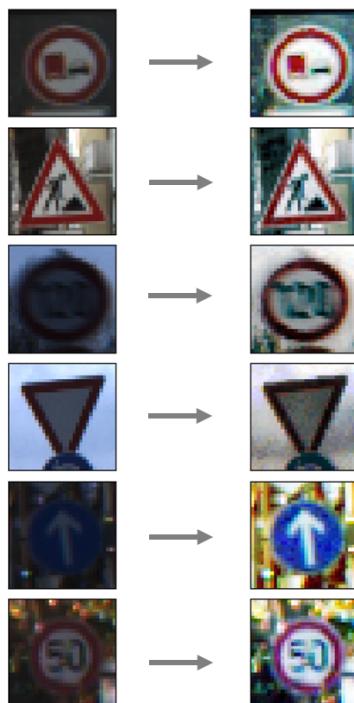


Figura 4.27: Ilustración de un modelo de aprendizaje profundo

Fuente propia

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

4.1.4. Dataset final para el Entrenamiento

4.1.4.1. Señales de Tránsito de Alemania - Dataset Balanceado

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

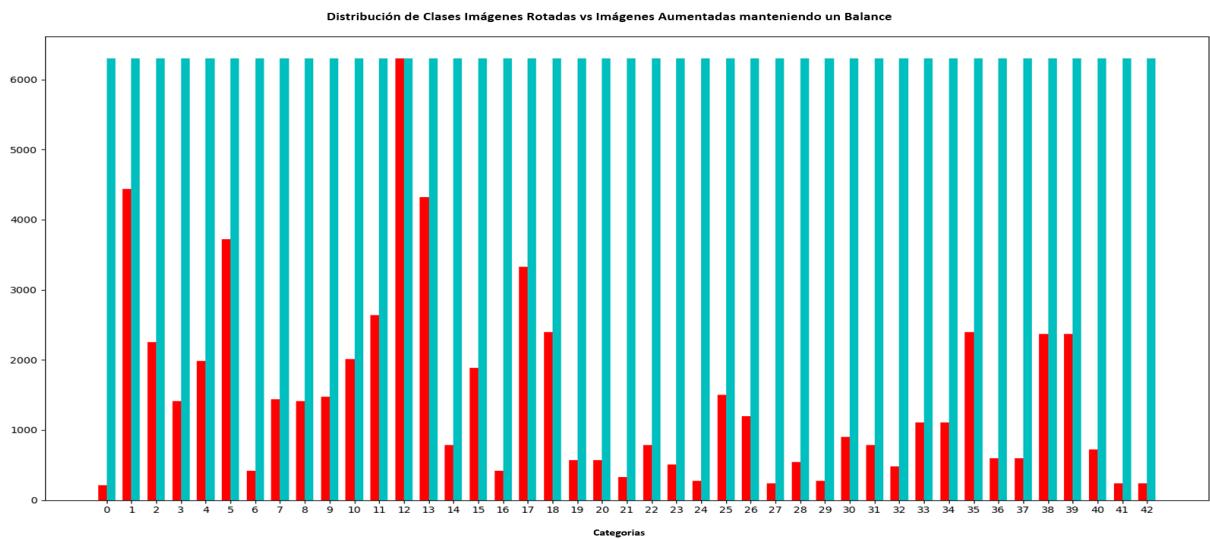


Figura 4.28: Dataset balanceado(cada categoría posee igual cantidad de imágenes)

Fuente 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 4.1: Distribución Entrenamiento y Validación para Dataset Balanceado - Señales de Tránsito de Alemania

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

4.1.4.2. 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**.

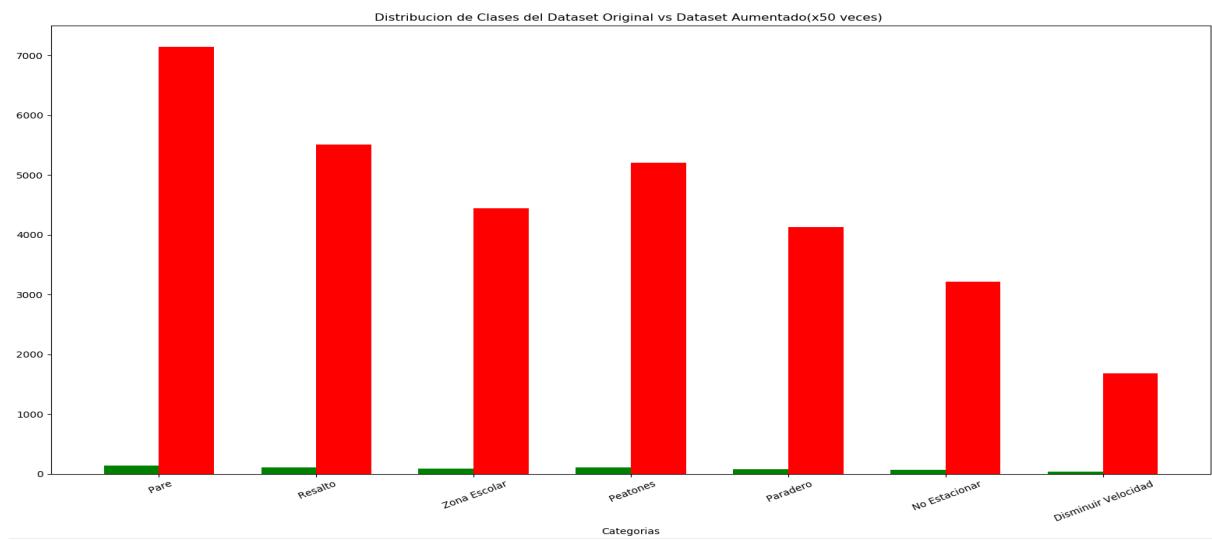


Figura 4.29: Dataset no balanceado - Señales de Tránsito de Perú

Fuente propia

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

Tabla 4.2: Distribución Entrenamiento y Validación para 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 %)

4.1.5. 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 4.30: Imágenes a las cuales se le aplicaron la técnica CLAHE

Fuente propia

Por otra parte, para convertir un color de un espacio de color basado en un modelo de co-

lor 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 (linealización) 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 rec601 luma (Y') se calcula como: $Y' = 0,299R' + 0,587G' + 0,114B'$

Pierre Sermanet y Yann LeCun mencionaron en su artículo (Y. LeCun, 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.

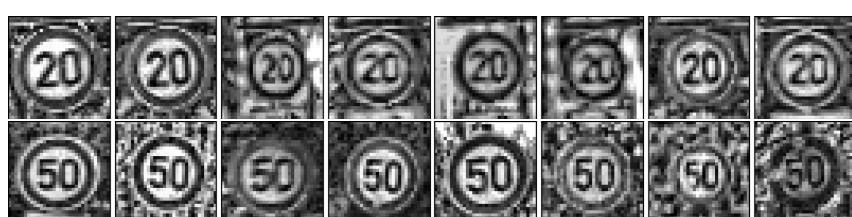


Figura 4.31: Imágenes procesadas en escala de grises

Fuente propia

4.2. 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 sobretodo despues de algunas pruebas realizadas, los siguientes hiperparámetros fueron seleccionados de manera específica:

Tabla 4.3: Hiperparámetros del Modelo

HIPERPARÁMETROS	TIPO	HIPERPARÁMETROS	TIPO
Inicialización de Pesos	Xavier	Tasa de Aprendizaje	0.0005
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	<i>L2 Lasso(lambda = 0.0001)</i>	Épocas	100

Recordemos que en cada época se procesa el dataset completo de imágenes, por lo que cada época es igual al mini-batch multiplicado por un número de iteraciones, sin embargo debido a que se tiene 3 distintos datasets para el entrenamiento (**Sección 4.1.4 - 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.

$$Epoch = Minibatch \times \text{iteraciones}$$

Tabla 4.4: Minibatch 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 4.5: Minibatch e iteraciones en el Dataset de Señales de Tránsito de Peru No Balanceado

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

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).

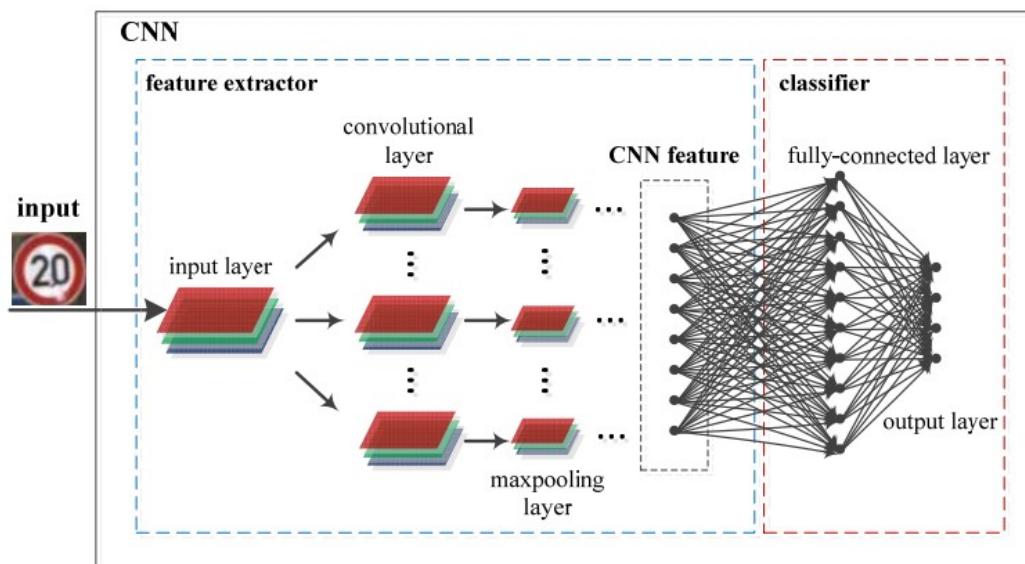


Fig. 1. CNN feature extractor architecture.

Figura 4.32: Arquitectura de la CNN

Extreme Learning Classifier

La arquitectura de la red está inspirada en el 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. 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 submuestren proporcionalmente antes de entrar en el clasificador. Así se garantiza que todas las funciones de escala múltiple experimenten la misma cantidad de máximo aprovechamiento.

4.2.1. Diseño de la Red

Teniendo en cuenta lo descrito en la **sección 2.2 (Figura 2.5)**, para describir las capas convolucionales se utilizará la terminología compleja(cada capa tiene múltiples etapas).

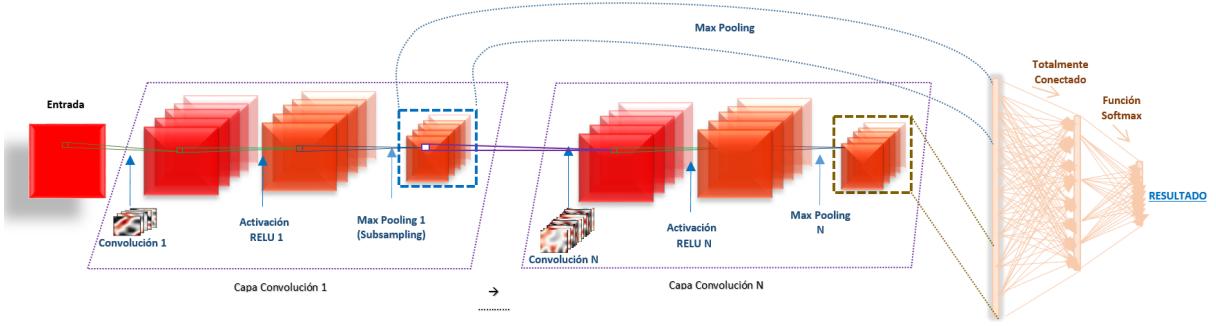


Figura 4.33: Modelo del diseño de la Red propuesta

Fuente 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. Posteriormente, el diseño que obtuvo mejor resultado de entrenamiento y validación fue utilizado para entrenar el dataset de señales de tránsito del Perú. Estos 5 modelos son descritos a continuación.

4.2.1.1. Diseño A

Tabla 4.6: Diseño compuesto de 2 capas convolucionales y 2 capas totalmente conectadas

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	32 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)	1024 neuronas	–	43 neuronas	–

4.2.1.2. Diseño B

Tabla 4.7: Diseño compuesto de 3 capas convolucionales y 2 capas totalmente conectadas

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	64 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	–

4.2.1.3. Diseño C

Tabla 4.8: Diseño compuesto de 3 capas convolucionales y 2 capas totalmente conectadas.
Variando el número de kernels en la 2da capa convolucional

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	64 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	–

4.2.1.4. Diseño D

Tabla 4.9: Diseño compuesto de 3 capas convolucionales y 2 capas totalmente conectadas.
Variando el número de kernels en la 3ra capa convolucional

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	64 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	–

4.2.1.5. Diseño E

Tabla 4.10: Diseño compuesto de 4 capas convolucionales y 2 capas totalmente conectadas

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 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 = 2) 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	–

4.3. Entrenamiento y Validación

Para estimar el error de clasificación, como fue mencionado en la **Sección 2.3.4 (Validación Cruzada)** 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 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.

4.3.1. Señales de Tránsito de Alemania

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

4.3.1.1. Análisis del Entrenamiento y Validación del Diseño A

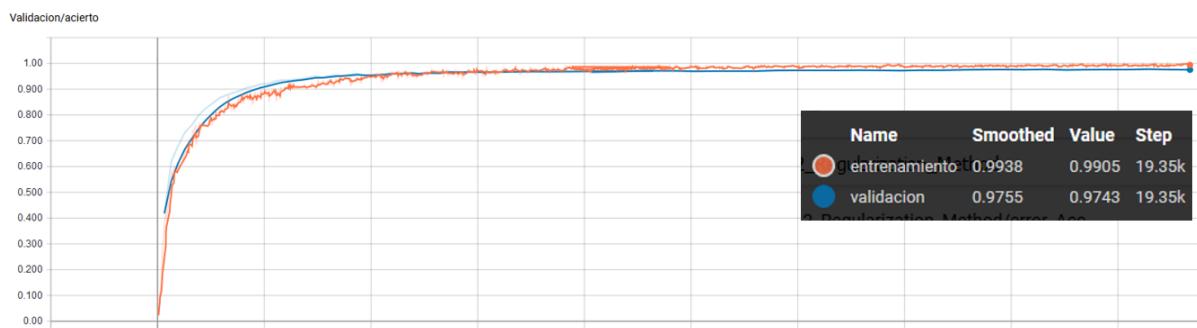


Figura 4.34: Modelo A Tasa de Acierto por iteración - Dataset de imágenes de Alemania
Fuente propia

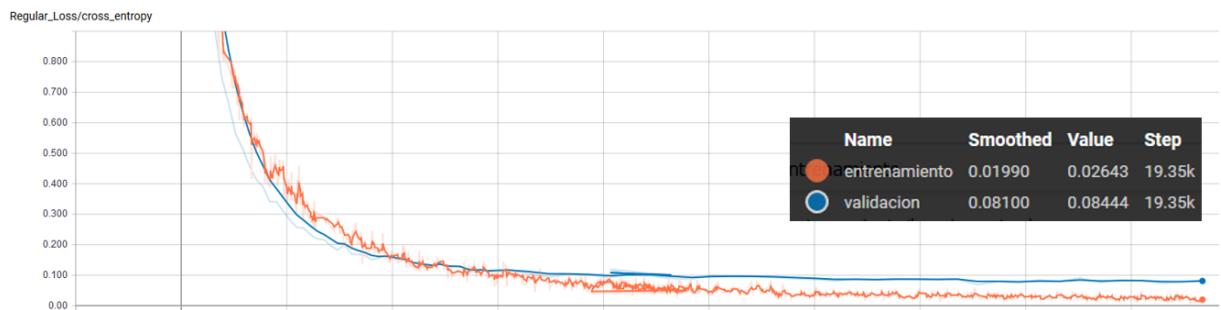


Figura 4.35: Modelo A Tasa de pérdida por iteración
Fuente propia

Luego de 50 épocas iteradas(19350 iteraciones), se observa que rápidamente el modelo se sobreajusta al dataset de entrenamiento. Es decir, rápidamente el valor de pérdida es muy cercana a cero, lo que a su vez hace que se separe de manera abrupta de los resultados de validación. Iteraciones posteriores harán que el modelo no permita obtener resultados correctos durante la prueba de evaluación.

4.3.1.2. Análisis del Entrenamiento y Validación del Diseño B



Figura 4.36: Modelo B Tasa de Acierto por iteración - Dataset de imágenes de Alemania

Fuente propia

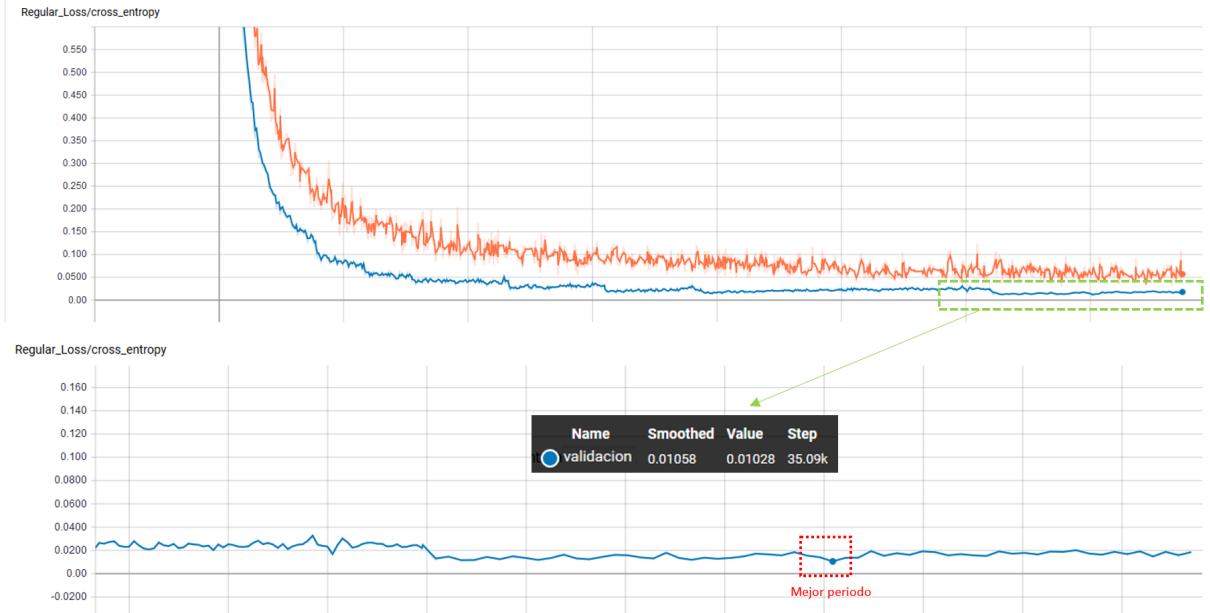


Figura 4.37: Modelo B Tasa de pérdida por iteración

Fuente propia

Durante 100 épocas iteradas(38700 iteraciones), aproximadamente en **34830 iteraciones(80 épocas)** se obtiene la mejor tasa de acierto de validación(99.26 %). Iteraciones posteriores hacen que el modelo se sobreajuste al conjunto de entrenamiento y no permita evaluar correctamente imágenes previamente analizadas.

4.3.1.3. Análisis del Entrenamiento y Validación del Diseño C

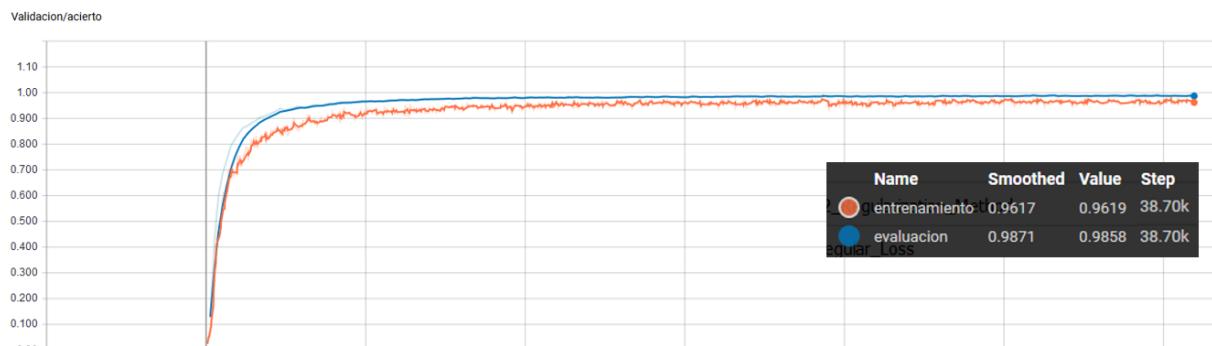


Figura 4.38: Modelo C Tasa de Acierto por iteración - Dataset de imágenes de Alemania
Fuente propia

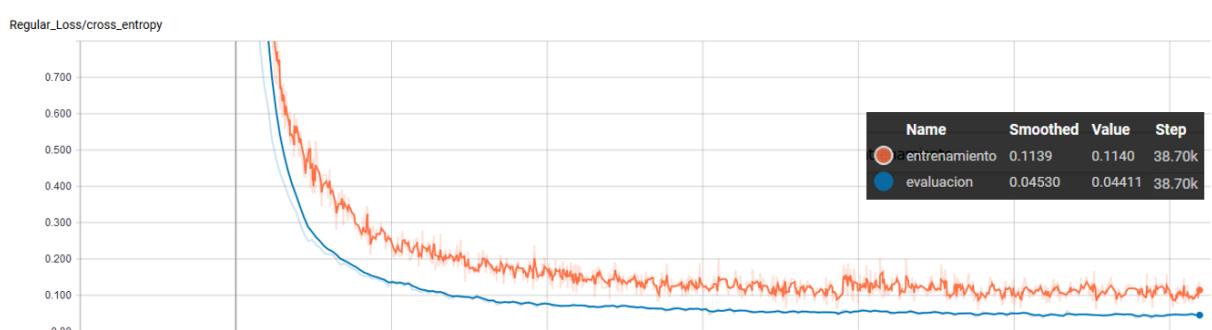


Figura 4.39: Modelo C Tasa de pérdida por iteración
Fuente propia

Durante 100 épocas iteradas(38700 iteraciones), al finalizar estas, se obtiene la mejor tasa de acierto de validación. Iteraciones posteriores harán probablemente que el modelo se sobreajuste al conjunto de entrenamiento y no permita evaluar correctamente imágenes previamente nos analizadas.

4.3.1.4. Análisis del Entrenamiento y Validación del Diseño D

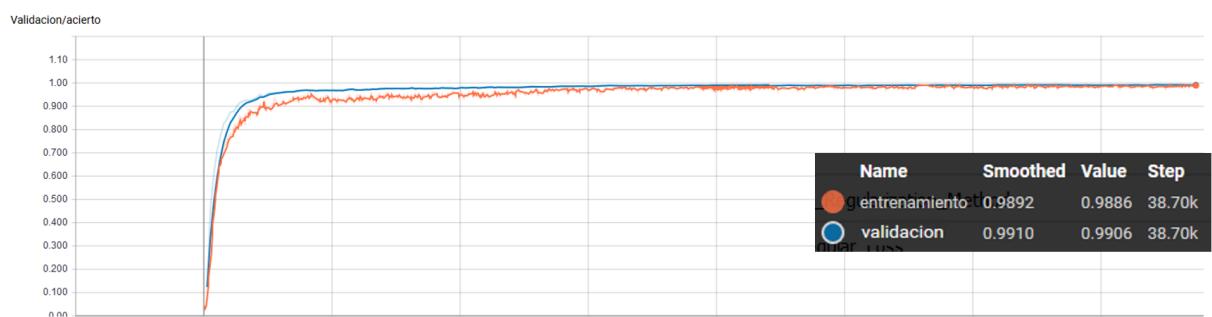


Figura 4.40: Modelo D Tasa de Acierto por iteración - Dataset de imágenes de Alemania
Fuente propia

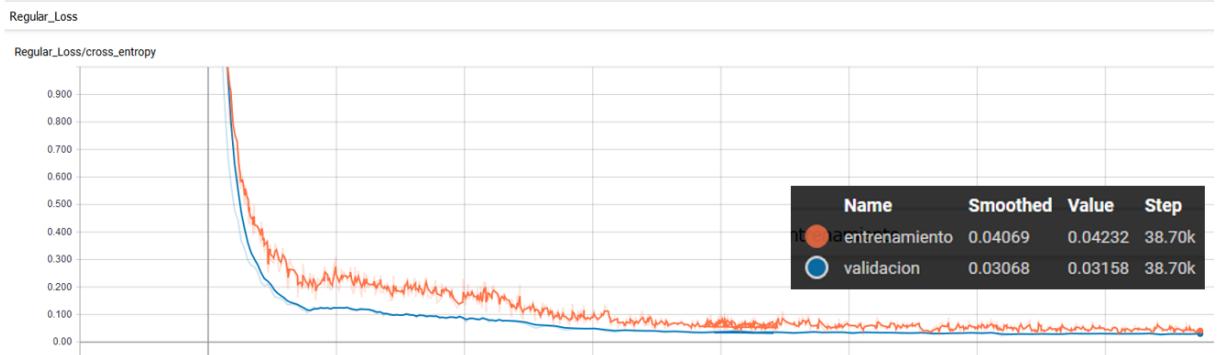


Figura 4.41: Modelo D Tasa de pérdida por iteración

Fuente propia

Al finalizar 100 épocas (38700 iteraciones) se obtiene una de las mejores tasas de acierto de validación. Iteraciones posteriores harán probablemente que el modelo se sobreajuste al conjunto de entrenamiento y no permita evaluar correctamente imágenes previamente analizadas.

4.3.1.5. Análisis del Entrenamiento y Validación del Diseño E

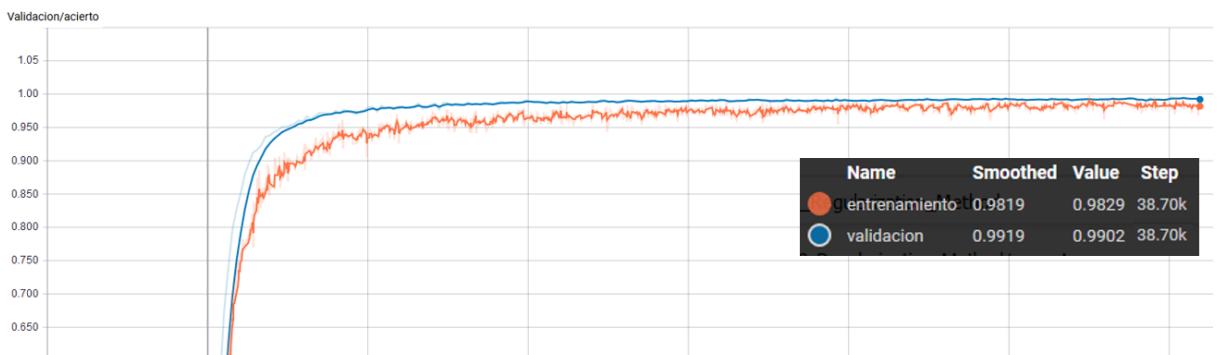


Figura 4.42: Modelo E Tasa de Acierto por iteración - Dataset de imágenes de Alemania

Fuente propia

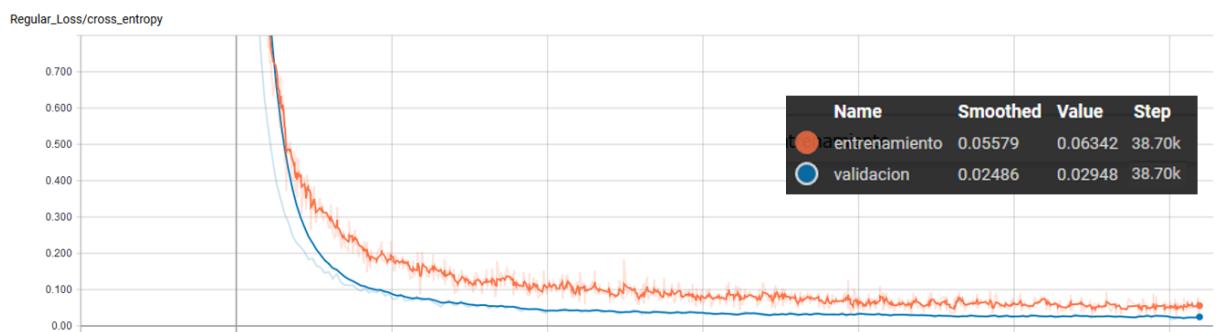


Figura 4.43: Modelo E Tasa de pérdida por iteración

Fuente propia

Al finalizar 100 épocas (38700 iteraciones) se obtiene una de las mejores tasas de acierto de validación. Iteraciones posteriores harán probablemente que el modelo se sobreajuste al conjunto de entrenamiento y no permita evaluar correctamente imágenes previamente analizadas.

4.3.2. Señales de Tránsito de Perú

Luego de haber entrenado las diferentes arquitecturas durante 100 épocas(7700 iteraciones), obtenemos los siguientes resultados:

4.3.2.1. Análisis del Entrenamiento y Validación del Diseño A

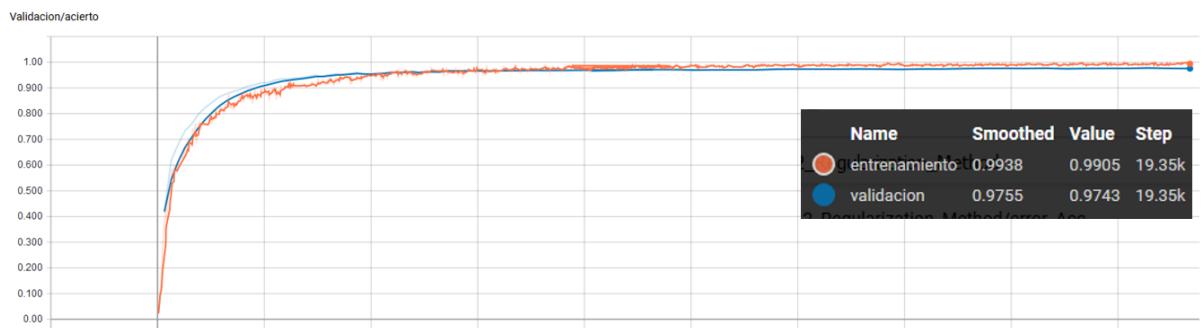


Figura 4.44: Modelo A Tasa de Acierto por iteración - Dataset de imágenes de Alemania

Fuente propia

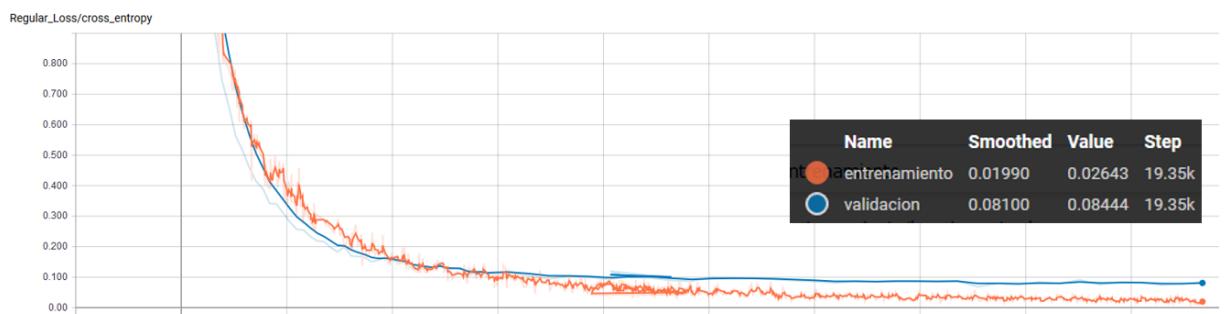


Figura 4.45: Modelo A Tasa de Pérdida por iteración

Fuente propia

Luego de 50 épocas iteradas(19350 iteraciones), se observa que rápidamente el modelo se sobreajusta 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.

Iteraciones posteriores harán que el modelo no permita obtener resultados correctos durante la prueba de evaluación.

4.3.2.2. Análisis del Entrenamiento y Validación del Diseño B



Figura 4.46: Modelo B Tasa de Acierto por iteración - Dataset de imágenes de Alemania

Fuente propia

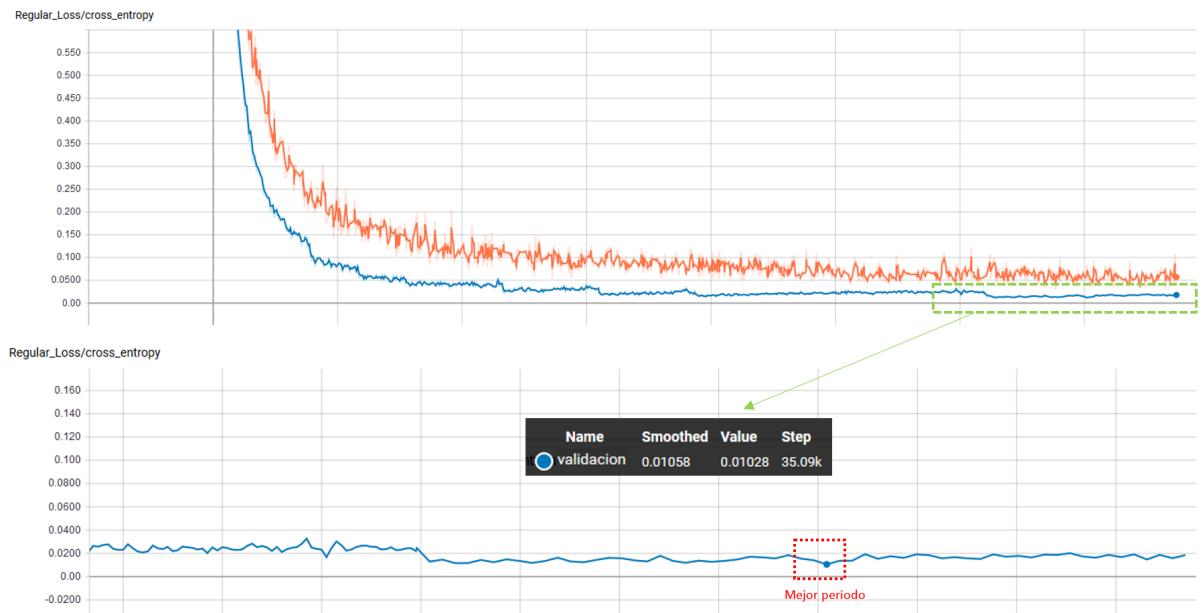


Figura 4.47: Modelo B Tasa de pérdida por iteración

Fuente propia

Durante 100 épocas iteradas(38700 iteraciones), aproximadamente en **34830 iteraciones(80 épocas)** se obtiene la mejor tasa de acierto de validación(99.26 %). Iteraciones posteriores hacen que el modelo se sobreajuste al conjunto de entrenamiento y no permita evaluar correctamente

imágenes previamente analizadas.

4.3.2.3. Análisis del Entrenamiento y Validación del Diseño C

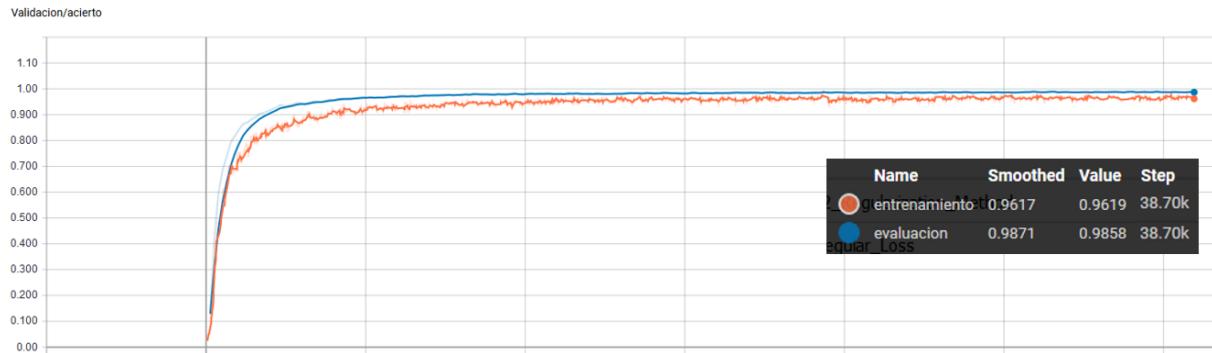


Figura 4.48: Modelo C Tasa de Acierto por iteración - Dataset de imágenes de Alemania

Fuente propia

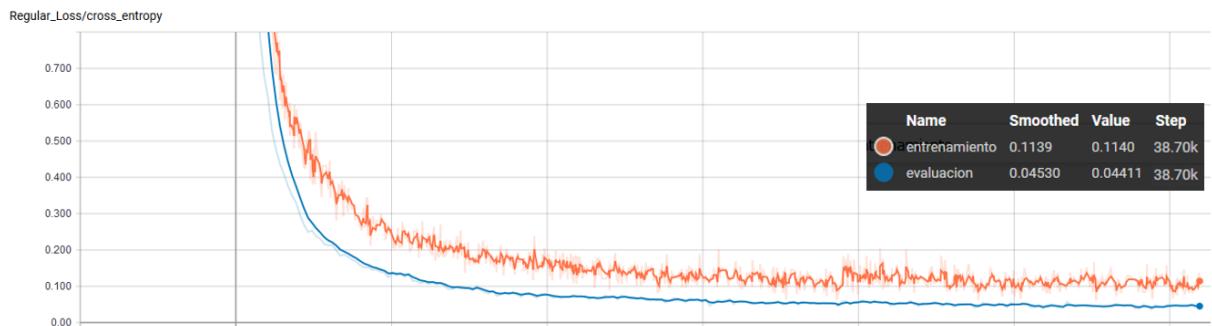


Figura 4.49: Modelo C Tasa de pérdida por iteración

Fuente propia

Durante 100 épocas iteradas (38700 iteraciones), al finalizar estas, se obtiene la mejor tasa de acierto de validación. Iteraciones posteriores harán probablemente que el modelo se sobreajuste al conjunto de entrenamiento y no permita evaluar correctamente imágenes previamente analizadas.

4.3.2.4. Análisis del Entrenamiento y Validación del Diseño D

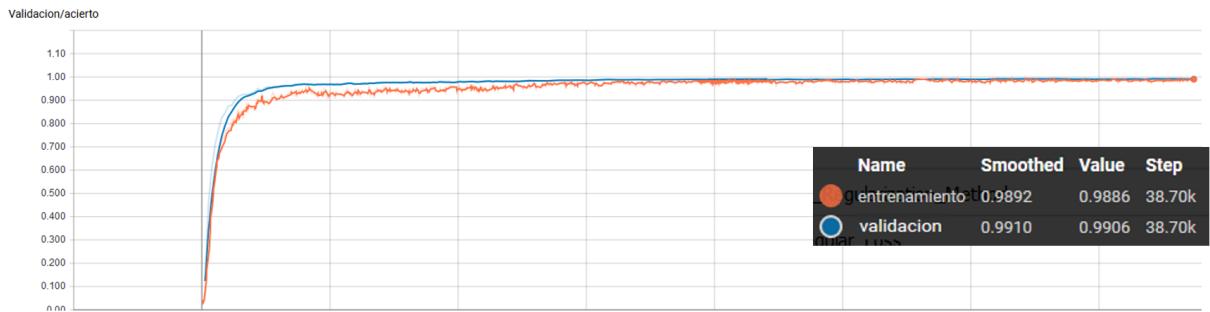


Figura 4.50: Modelo D Tasa de Acierto por iteración - Dataset de imágenes de Alemania

Fuente propia

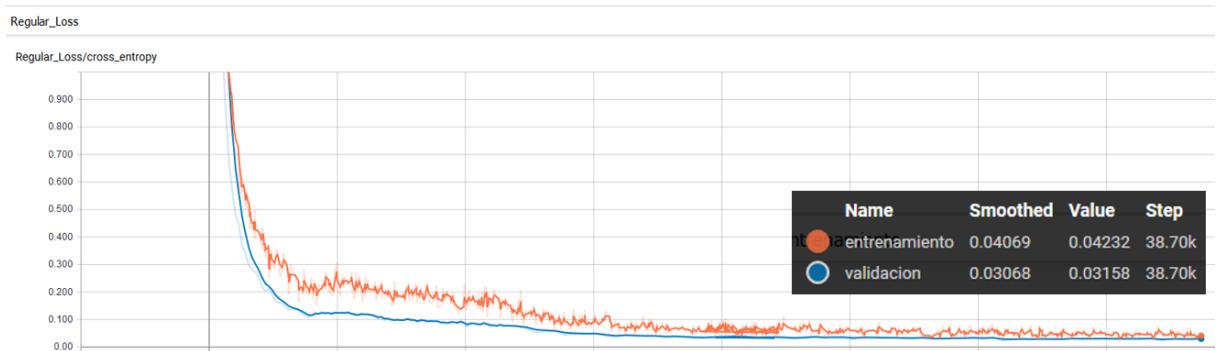


Figura 4.51: Modelo D Tasa de pérdida por iteración

Fuente propia

Al finalizar 100 épocas (38700 iteraciones) se obtiene una de las mejores tasas de acierto de validación. Iteraciones posteriores harán probablemente que el modelo se sobreajuste al conjunto de entrenamiento y no permita evaluar correctamente imágenes previamente analizadas.

4.3.2.5. Análisis del Entrenamiento y Validación del Diseño E

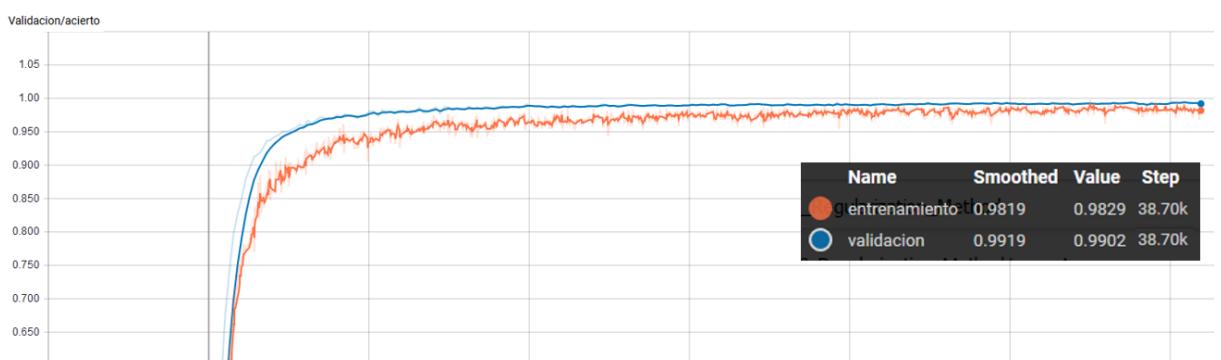


Figura 4.52: Modelo E Tasa de Acierto por iteración - Dataset de imágenes de Alemania

Fuente propia

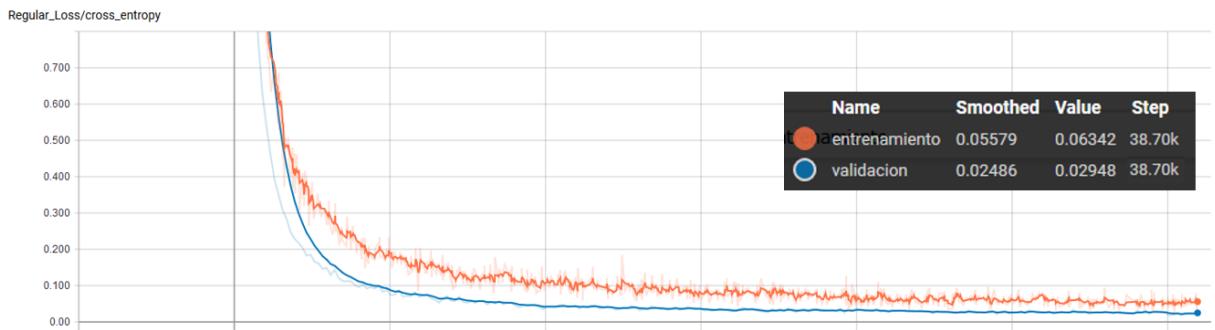


Figura 4.53: Modelo E Tasa de pérdida por iteración

Fuente propia

Al finalizar 100 épocas (38700 iteraciones) se obtiene una de las mejores tasas de acierto de validación. Iteraciones posteriores harán probablemente que el modelo se sobreajuste al conjunto de entrenamiento y no permita evaluar correctamente imágenes previamente nos analizadas.

4.4. Resultados de Evaluación

La información de la prueba de clasificación se dispone en 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; en la Tabla 3, se muestra una matriz de confusión para dos clases. 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 donde se obtienen los 3 indicadores de desempeño (mencionado en la Sección 3.3 Indicadores).

1. Tasa de Sensibilidad(Efectividad): $S = \frac{\text{Verdaderos Positivos}}{\text{Verdaderos Positivos} + \text{Falsos Negativos}} \times 100$
2. Tasa de Error(Especificidad): $S = \frac{\text{Falsos Negativos}}{\text{Verdaderos Positivos} + \text{Falsos Negativos}} \times 100$
3. El espacio ROC (Curvas ROC): Relación entre efectividad y especificidad

El desempeño de la clasificación se puede verificar con los indicadores de Proporción de verdaderos positivos(Tasa de Sensibilidad / Recall), con el cual se puede medir la efectividad del clasificador, y la proporción de verdaderos negativos (Tasa de Error) que nos muestra la especificidad del clasificador.

Por otra parte , el espacio ROC (Receiver Operating Characteristic) define un sistema de coordenadas usadas para visualizar el desempeño del clasificador. Las curvas ROC presentan el compromiso entre efectividad y especificidad del clasificador, un aumento en la sensibilidad está acompañado por un decremento en la especificidad. Es decir, las curvas ROC muestran la relación entre las muestras clasificadas adecuadamente (PVP, Proporción de Verdaderos Positivos) y las muestras que no pertenecen a la clase pero se clasificaron como si lo fueran (PFP Proporción de Falsos Positivos). En el espacio ROC la PFP se dibuja como variable independiente y la PVP como variable dependiente. Cada clasificador es representado por el punto (PFP, PVP). Para dibujar la curva ROC se construye la línea convexa formada por los puntos (PFP, PVP) de los clasificadores que se estén evaluando ($PFP=1-PVN$), 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. El mejor sistema de entrenamiento es el que produce un conjunto de clasificadores que maximice el área bajo la curva,(Sandoval, 2009).

4.4.1. Señales de Tránsito de Alemania

Obtenemos los siguiente resultados:

4.4.1.1. Análisis del Diseño A

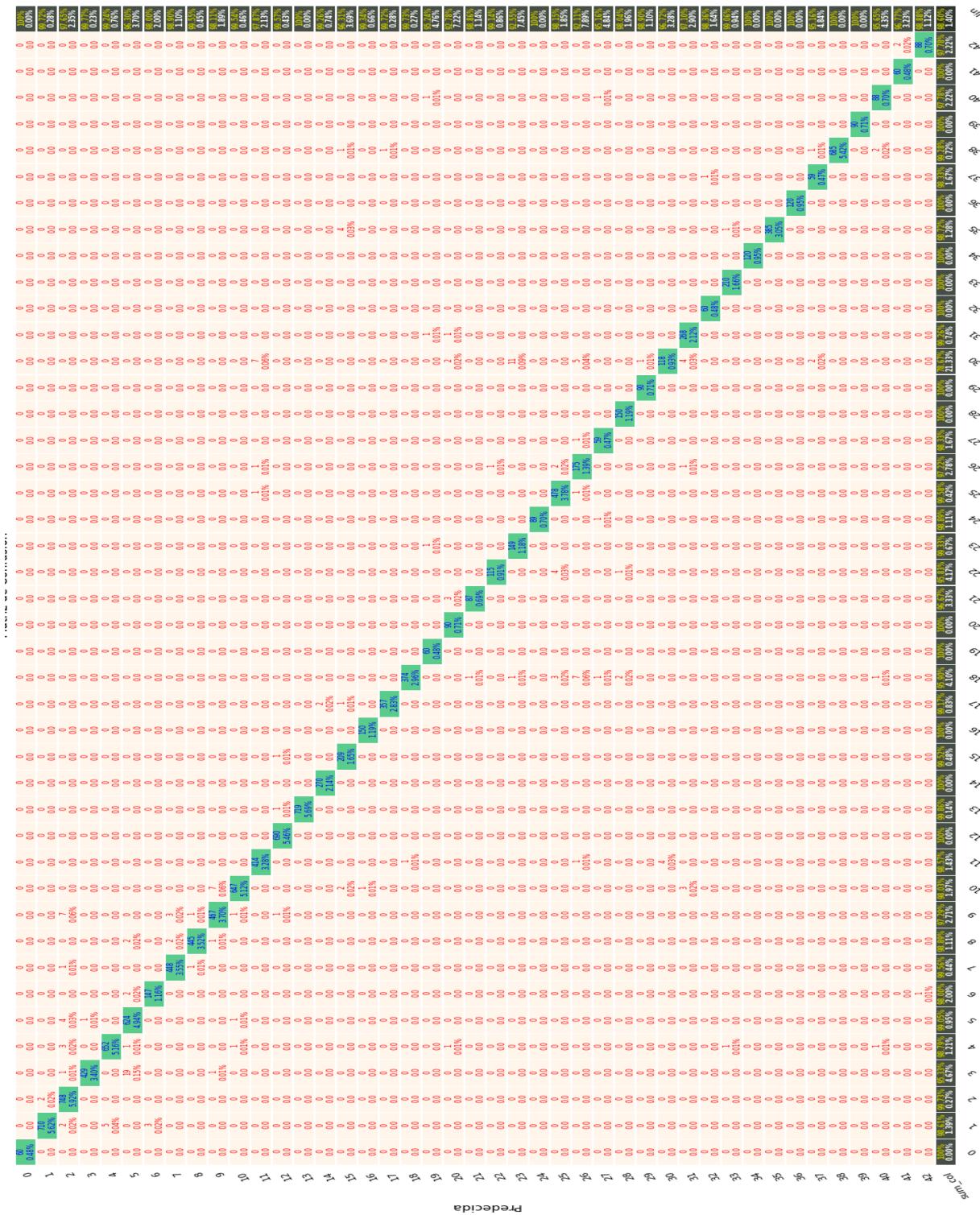


Figura 4.54: Matriz de Confusión del Modelo A - Dataset de imágenes de Alemania
Fuente propia

4.4.2. Señales de Tránsito de Perú

Obtenemos los siguiente resultados:

4.4.2.1. Análisis del Diseño A

		Matriz de Confusión								
		0	1	2	3	4	5	6	7	8
Predicida	0	1020 21.71%	0 0.0	3 0.06%	1 0.02%	0 0.0	1 0.02%	1 0.02%		1026 99.42% 0.58%
	1	4 0.09%	837 17.82%	10 0.21%	2 0.04%	3 0.06%	1 0.02%	9 0.19%		866 96.65% 3.35%
	2	3 0.06%	2 0.04%	668 14.22%	2 0.04%	2 0.04%	3 0.06%	2 0.04%		682 97.95% 2.05%
	3	1 0.02%	2 0.04%	6 0.13%	807 17.18%	1 0.02%	5 0.11%	3 0.06%		825 97.82% 2.18%
	4	3 0.06%	0 0.0	0 0.0	3 0.06%	610 12.98%	2 0.04%	2 0.04%		620 98.39% 1.61%
	5	0 0.0	0 0.0	0 0.0	2 0.04%	3 0.06%	437 9.30%	0 0.0		442 98.87% 1.13%
	6	3 0.06%	1 0.02%	0 0.0	0 0.0	0 0.0	3 0.06%	230 4.90%		237 97.05% 2.95%
	sum_col	1034 98.65% 1.35%	842 99.41% 0.59%	687 97.23% 2.77%	817 98.78% 1.22%	619 98.55% 1.45%	452 96.68% 3.32%	247 93.12% 6.88%		4698 98.11% 1.89%
		Deseada							sum_dn	

Figura 4.55: Matriz de Confusión del Modelo A - Dataset de imágenes de Perú

Fuente propia

Capítulo 5

Resultados de la tesis

Al culminar con la investigación se llegaron a resultados interesantes del punto de vista tanto teórico como computacional. ...

5.1. Teóricos

5.2. Computacionales

Capítulo 6

Consideraciones finales

6.1. Conclusiones

Conclusion Our MCDNN won the German traffic sign recognition benchmark with a recognition rate of 99.46(98.84algorithm (98.31ing method, increases the recognition rate from an average of 98.5299.46DNN recognition rates, but combining them into a MCDNN increases robustness to various types of noise and leads to more recognized traffic signs. We plan to embed our method in a more general system that first localizes traffic signs in realistic scenes and then classifies them

Ejemplo

La investigación bibliográfica revela que realmente existe una preocupación de los gobiernos federales con el destino final de los residuos sólidos, con el objetivo de preservar la salud de la población y el medio ambiente urbano y rural. Por ejemplo, se observa la creación, en el caso del Brasil, de la Ley 12.305/10. Sin embargo existe una laguna entre las metas propuestas en la ley con las metas reales de los gobiernos locales. Eso se debe a la falta de una buena estructura

organizacional, gerencial y operacional de los gobiernos locales capaz de atender las demandas locales y las necesidades de la población.

La falta de cuadros especializados, tanto en los gobiernos centrales como locales, para realizar la planificación y modelamiento de una red logística reversa puede ser compensada con la contribución de los investigadores que actúan en ese campo del conocimiento. Es muy difícil la formación de un equipo que tenga todo el conocimiento en las áreas de ciencia de la computación, de geo procesamiento, de modelamiento matemático y de logística reversa, entre otras. Esa es una de las principales justificativas que los gobiernos, federales y locales, argumentan a la falta de planificación de una red logística reversa que funcione eficaz y eficientemente.

Por lo tanto, como quedó demostrado a lo largo de este trabajo, es posible realizar el modelamiento matemático para este tipo de problema con baja inversión, así como aplicarlo en varias regiones sin necesidad de grandes cambios en el modelamiento propuesto. El modelo propuesto calcula los flujos en la red logística reversa, permitiendo dimensionar la cantidad y capacidad de las unidades productivas y de los vehículos.

...

6.2. Trabajos futuros

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. page 48.
- 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>.
- 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 medir 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>.

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. *Przegląd Elektrotechniczny*, R. 90, nr 11:169–172.

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

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.

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. pages 1097–1105.
- 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 Escoria, 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, Z., . P. F. (2009). Procesamiento de imágenes para la clasificación de café cereza.
- Sermanet, P. and LeCun, Y. (2011). Traffic sign recognition with multi-scale convolutional networks. pages 2809–2813.
- 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. *Neural Networks*, (0).
- SUTRAN (2014). Texto unico ordenado del reglamento nacional de transito - codigo de transito. page 188.
- 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., García-González, A., Torijano-Gordo, E., Gil-Pita, R., and Rosa-Zurera, M. (2007). *Traffic Sign Classification by Image Preprocessing and Neural Networks*, pages 741–748. 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/>.

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

Apéndice A

Primer apendice

hola como

Apéndice B

Segundo apendice

si te escucho

Apéndice C

Tercer apendice

sdsdsd