UNIVERSIDAD AUTÓNOMA DE CHIHAUHUA

FACULTAD DE INGENIERÍA

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

CLASIFICACIÓN AUTOMÁTICA DE LOS TRAZOS DEL TEST DE BENDER BASADA EN MODELOS DE APRENDIZAJE PROFUNDO

POR:

ING. DIONISIO RUIZ VÁZQUEZ

TESIS PRESENTADA COMO REQUISITO PARA OBTENER EL GRADO DE MAESTRO EN INGENIERÍA EN COMPUTACIÓN

CHIHUAHUA, CHIH., MÉXICO **NOVIEMBRE DE 2018**

Clasificación automática de los trazos del Test de Bender basada en modelos de aprendizaje profundo. Tesis presentada por Ing. Dionisio Ruiz Vázquez como requisito parcial para obtener el grado de Maestría en Ingeniería en Computación, ha sido aprobada y aceptada por

M.I. Javier González Cantú
Director de la Facultad de Ingeniería

Dr. Alejandro Villalobos Aragón
Secretario de Investigación y Posgrado

M.S.I. Karina Rocío Requena Yáñez
Coordinadora Académica

Graciela Ramirez A

Dra. Graciela María de Jesús Ramírez Alonso
Director de Tesis

NOVIEMBRE 2018

Fecha

Comité:
Dra. Graciela María de Jesús Ramírez Alonso
Dr. Luis Carlos González Gurrola
Dr. Fernando Martínez Reyes
Dr. Raymundo Cornejo García

Dionisio Ruiz Vázquez

LATERAL DE JUAN ESCUTIA SUR #827

CHIHUAHUA, MÉXICO

NOVIEMBRE DE 2018

UNIVERSIDAD AUTÓNOMA DE
CHIHUAHUA

9 de noviembre de 2018

ING. DIONISIO RUIZ VÁZQUEZ

Presente

En atención a su solicitud relativa al trabajo de tesis para obtener el grado de Maestro en Ingeniería en Computación, nos es grato transcribirle el tema aprobado por esta Dirección, propuesto y dirigido por el director **Dra. Graciela María de Jesús Ramírez Alonso** para que lo desarrolle como tesis, con el título: **CLASIFICACIÓN AUTOMÁTICA DE LOS TRAZOS DEL TEST DE BENDER BASADA EN MODELOS DE APRENDIZAJE PROFUNDO.**

Índice

FACULTAD DE INGENIERÍA
Circuito No.1, Campus Universitario 2
Chihuahua, Chih., México. C.P. 31125
Tel. (614) 442-95-00
www.fing.uach.mx

UNIVERSIDAD AUTÓNOMA DE
CHIHUAHUA

4. Experimentación y resultados

4.1. Conjuntos de datos

4.2. Selección de arquitectura CNN

4.3. Pruebas con los conjuntos de datos BGT, MNIST y OIHACDB-40

5. Conclusiones y trabajo futuro

5.1. Trabajo futuro

Referencias

Solicitamos a Usted tomar nota de que el título del trabajo se imprima en lugar visible de los ejemplares de las tesis.

A T E N T A M E N T E
"Naturam subiecit aliis"

EL DIRECTOR

M.I. JAVIER GONZÁLEZ CANTÚ

EL SECRETARIO DE INVESTIGACIÓN Y POSGRADO

DR. ALEJANDRO VILLALOBOS ARAGÓN

FACULTAD DE INGENIERÍA U.A.CH.
CHIHUAHUA
Facultad de Ingeniería
DIRECCIÓN

FACULTAD DE INGENIERÍA
Circuito No.1, Campus Universitario 2
Chihuahua, Chih., México. C.P. 31125
Tel. (614) 442-95-00
www.fing.uach.mx

Resumen

En este trabajo se presenta una metodología para la clasificación automática de los nueve trazos del Test de Bender utilizando modelos computacionales. Esta prueba tiene por objetivo el evaluar la madurez viso-motriz por medio del análisis de los nueve trazos. En total se evaluaron nueve diferentes modelos computacionales basados en Redes Neuronales, particularmente Redes Neuronales Convolucionales. Con el fin de aumentar la base de datos y contar con muestras en donde se consideraran diferentes problemáticas como rotación, traslación, escalamiento y distorsión elástica se implementaron estrategias de "data augmentation". De igual manera se implementaron estrategias de transferencia de aprendizaje utilizando otras bases de datos con la finalidad de mejorar los resultados en la clasificación. Los resultados que se obtuvieron superan 90 % de exactitud en todas las iteraciones demostrando así que el uso de modelos computacionales basados en Redes Neuronales Convolucionales son capaces de clasificar automáticamente los nueve trazos de Bender. El trabajo tiene como finalidad crear las bases para una herramienta que puedan usar los expertos en el área y reducir la carga de trabajo al revisar el Test de Bender.

Abstract

In this work is presented a methodology to automatically classify the nine strokes on the Bender Gestalt Test using computational models. The aim of the Test is to measure the visomotor skill by analyzing the nine strokes. In total nine different computational models based on artificial neural networks, specifically convolutional neural networks were tested. In order to expand the dataset and to have data with different problematics, such as rotation, translation, scale and elastic distortion were implemented as a data augmentation strategy. In the same way, transfer learning strategies were implemented using different databases with the purpose to boost performance. The results achieved overcome 90 % accuracy in all folds, demonstrating that the use of computational models based con convolutional neural networks are able to automatically classify the nine Bender strokes.

Índice de Contenido

Índice de figuras

Índice de tablas

1

Introducción

Existen tareas como escribir, dibujar o realizar trazos a mano que implican llevar a cabo un complejo proceso donde se hace uso de habilidades cognitivas y motoras. Especialistas en psicología y neurología han desarrollado mecanismos y pruebas que ayudan en la examinación de partes específicas del cerebro humano para poder encontrar anomalías y deficiencias en los pacientes, de esta manera brindar el apoyo necesario para su tratamiento [1]. El Test Guestáltico Visomotor de Bender (BGT por sus siglas en inglés, *Bender-Gestalt test*) es un examen que ayuda a explorar posibles trastornos visomotores y casos con déficit en el desarrollo de niños [2, 3]. En el examen BGT, se le presentan nueve tarjetas al paciente las cuales contienen los trazos que debe de dibujar sobre una hoja en blanco. Estos trazos son interpretados por un especialista en el área otorgándole una calificación con base en los errores cometidos [4]. En la Figura 1.1 se muestran los nueve trazos a replicar por el niño [5].

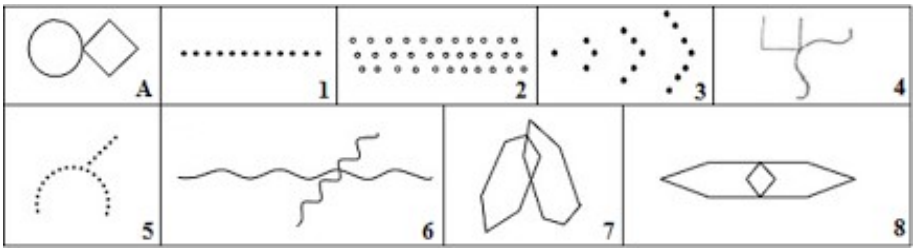


Figura 1.1: Trazos a replicar en el BGT.

El que un paciente realice el BGT, requiere tiempo del especialista tanto en la aplicación como en la revisión. Si se toma a consideración el aplicar el examen a 10 o más pacientes, existe la posibilidad de que detalles en los trazos pasen desapercibidos, originado posibles errores de interpretación. Es aquí, donde existe un área de oportunidad para automatizar el proceso de revisión utilizando la visión por computadora.

En los últimos años se han incrementado en gran medida las aplicaciones de visión por computadora [6]. En parte, gracias al desarrollo en la tecnología, permitiendo procesar una gran cantidad de datos en una computadora personal. En general, la visión por computadora comprende adquirir, procesar y analizar los datos para entender la información contenida en imágenes y videos. Entre las áreas favorecidas en su desarrollo debido a la visión por computadora, se encuentra la medicina en la detección de hemorragias cuando se opera a un paciente utilizando la técnica laparoscopia [7], la industria agrícola en la detección de materias extrañas en algodón [8], la vigilancia y seguridad avisando al usuario que hay movimiento y que objeto es el que se mueve [9], la biología como herramienta para la transcriptómica [10], la geología para caracterízar y cuantizar las propiedades del suelo y materia orgánica [11], entre otras [12, 13, 14]. Otra parte importante del desarrollo de la visión por computadora reside en el diseño y mejora de métodos y algoritmos de aprendizaje máquina (*machine learning*), principalmente, utilizando un enfoque basado en aprendizaje profundo (*deep learning*) con las que se han obtenido mejores resultados comparándose con los seres humanos en algunas tareas de clasificación basadas en imágenes [15, 16].

Las redes neuronales convolucionales (CNNs por sus siglas en inglés, *convolutional neural networks*) son una herramienta del aprendizaje profundo que ha sobresalido debido a su efectividad en tareas de visión. Basándose en la corteza visual primaria extrayendo características (bordes, manchas, colores, etc.) de las imágenes, tal y como lo hacen las células corticales primarias [17].

En este trabajo se busca clasificar automáticamente imágenes que contienen trazos realizados por niños (los correspondientes al BGT). Para esta tarea, se escogió utilizar CNNs debido a su desempeño en tareas similares y el hecho de no necesitar una extracción de características de las imágenes para poder adquirir una interpretación de los datos y realizar el reconocimiento. Con la finalidad de dar inicio a una herramienta que pudiera brindar apoyo a los profesionales de la psicología. Permitiendo reducir el tiempo que utilizan para interpretar, calificar y obtener los resultados arrojados por el examen.

El reconocer trazos mediante técnicas de reconocimiento óptico de caracteres (OCR por sus siglas en inglés, *optical character recognition*), no es una tarea sencilla, dado que existen problemáticas diferentes. Por ello se necesita conocer la manera en la que se desarrollaron diversas investigaciones referentes al tema. Abarcando desde el tipo de datos y las técnicas utilizadas para lograr el objetivo de reconocer trazos o caracteres.

La visión por computadora es un campo interdisciplinario en el que se busca obtener la mayor cantidad de información por medio de imágenes, tal y como lo hace un ser humano a través de la vista. Una de las aplicaciónes que se utilizan gracias a la visión por computadora es el OCR. En el OCR se ingresa la información a la computadora mediante un dispositivo de adquisición de imágenes. Utilizando métodos de reconocimiento de patrónes o aprendizaje máquina, la máquina se encarga de hacer el reconocimiento de cada caracter.

Existe una amplia variedad de métodos dentro de la rama del aprendizaje automático. Dentro los más populares destacan: máquinas de vectores de soporte (SVM por sus siglas en inglés, *support vector machine*) [18], análisis de discriminante lineal (LDA por sus siglas en inglés, *linear discriminant analysis*) [19], K vecinos más cercanos (K-NN por sus siglas en inglés, *K-nearest neighbor*) [20], redes neuronales (NN por sus siglas en inglés, *neural networks*) [21] y variantes de éstas, entre otros [22].

En 1998 LeCun *et al.* probaron distintos métodos para clasificar números arábigos. Entre

los métodos que utilizaron se encuentran redes neuronales lineales, *K*-NN, SVM, perceptrón multi capa, CNN, entre otros [23]. Dentro de los métodos utilizados por LeCun *et al.* se encuentra uno de los más simples, éste es el clasificador lineal, para el cuál, implementaron una red neuronal con 10 neuronas. Ésta red se entrenó con los datos del conjunto MNIST. El conjunto de datos MNIST [24], cuenta con 70,000 imágenes en total, 60,000 para entrenamiento y 10,000 para prueba. Se divide en 10 clases, teniendo 6,000 imágenes por categoría, las cuales corresponden a los dígitos numéricos arábigos. Las figuras del MNIST están preprocesadas a escala de grises con un tamaño de 28×28 pixeles.

La red cuenta con 7,850 parámetros libres (dado por el tamaño del vector de características y las conexiones que existen entre las neuronas), los datos de entrada corresponden al valor de los pixeles. En base a esto se obtuvo un error de clasificación del 12 %. Además, se entrenó la red con imágenes enderezadas y recortadas a 20×20 pixeles, reduciendo los parámetros libres a 4,010, y el error de clasificación a un 8.4 %.

En LeCun *et al.* (ibídem, 13) implementan un par de redes neuronales con el conjunto MNIST. A diferencia de la red anterior, éstas son perceptrones multicapa (MLP por sus siglas en inglés, *multilayer perceptron*). El MLP es un tipo de red neuronal artificial, de al menos una capa oculta [25]. La arquitectura de la primer red neuronal cuenta con 300 neuronas en su capa oculta, presentando un error de clasificación de 4.7 %. Los autores utilizaron una arquitectura similar para la segunda red, habiendo en ésta 1,000 neuronas en la capa oculta, obteniendo un error de 4.5 % en la clasificación de los caracteres. Estos resultado se mejoraron empleando técnicas para aumentar los datos. Dichas técnicas tienen como propósito modificar la imágen original para crear una nueva con distintas propiedades. Entre las técnicas más comunes están: rotación, distorsión elástica, difuminado, blanqueo, estandarizado de pixeles, entre otras [26]. De las 60,000 imágenes originales para el entrenamiento de las redes neuronales, se obtuvieron otras 540,000 imágenes mediante distorsiones elásticas.

A estas mismas redes, se les entrenó con las imágenes distorsionadas, bajando el error a un 3.6 % en la red con 300 neuronas y a un 3.8 % en la red con 1,000 neuronas. Una mejora en el desempeño de la red con 300 neuronas, se obtuvo cuando se entrenó con las imágenes enderezadas, alcanzando un error en la presición de 1.6 %.

Es posible utilizar un clasificador híbrido entre K-NN y SVM, tal y como presenta Zhang *et al.* en [27], el cuál fue utilizado en conjuntos de datos como MNIST y USPS. El conjunto de datos USPS [28] contiene 9,298 imágenes de caracteres numéricos, 7,291 para entrenar y 2,007 para evaluar. Las imágenes fueron recopiladas por el servicio postal de Búffalo en Estados Unidos. Durante el preprocesamiento se separaron los números y se etiquetaron, además de hacer binarias las imágenes con un tamaño de 16×16 pixeles.

Los autores utilizaron una variante del SVM anteriormente desarrollada por Platt *et al.* [29] llamada DAGSVM. Para la clasificación, Zhang *et al.*, midieron la distancia euclidiana hacia un conjunto de vecinos, de los cuáles, se escogieron los K vecinos más cercanos para calcular la distancia tangencial. Una vez obtenidas las distancias, se agrupan con la muestra de consulta y mediante una transformación del kernel se cambian los datos de una matriz de distancias a una matriz de kernel. Una vez que los datos de la matriz de kernel están acomodados al tipo de estructura necesario, se clasifica la muestra de consulta con el método DAGSVM. Con este clasificador, los autores obtuvieron como resultado un error en la clasificación del 1.66 %, esto en base a la distancia euclidiana y utilizando el conjunto MNIST. Mientras que con el conjunto USPS obtuvieron un error de 4.285 % en la distancia euclidiana.

En el trabajo de A. Cardoso y A. Wichert [16], se extraen características de imágenes del conjunto de datos de USPS y MNIST. Las características se extraen mediante un modelo biológico propuesto por Hubel y Wiesel, basado en la corteza visual para hacer una cascada de transformación de mapas (MTC por sus siglas en inglés, *map transformation cascade*). En el modelo implementado, existen células simples que prefieren un estímulo, mientras que

existen células complejas y su función es ser invariables a la posición del estímulo. Gracias al algoritmo *k-means* se aprende el estímo preferido de las células simples, de esta manera se extraen las características que va a utilizar un SVM lineal para clasificar las imágenes. Los autores reportan un error de 2.64 % en el conjunto USPS y un 0.71 % en el MNIST.

Los autores de [30] utilizaron un algoritmo Rprop para mejorar la función objetivo de la regresión lineal SSM-MCE que emplearon para clasificar letras chinas. El conjunto de datos consta de 9,252 caracteres chinos, los cuales están a escala de grises y normalizados, midiendo 64×64 pixeles. Obteniendo de éstas 128 características utilizadas para clasificar.

A. Boukharouba y A. Bennia [31], utilizaron el conjunto de datos de dígitos *Farsi* escritos a mano. Estos datos constan de 80,000 imágenes de números persas de imágenes binarias. Durante la fase de entrenamiento se utiliza un 75 % de las muestras totales, por lo que se deja un 25 % para las pruebas. A las imágenes se les extrajeron ciertas características, una parte de ellas se basan en el histograma de código de cadena, donde se describe estadísticamente la dirección del borde de cada dígito. Una de las ventajas de este algoritmo, es que no se requiere normalizar las imágenes. La segunda etapa de extracción de características yace en la información de la transición entre los colores blanco y negro en las direcciones horizontal y vertical de la imágen. Obteniendo un total de 46 características por imágen. Estas características se almacenan en un vector y son con las que se entrena el clasificador. El modelo para clasificar corresponde a un SVM con kernel de base radial (RBF) para su entrenamiento y posterior clasificación. Durante la etapa de prueba obtuvieron una exactitud del 98.48 %.

En [32] los autores utilizaron una CNN para extraer características de un conjunto de datos llamado HACDB que cuenta con 6,600 caracteres arábigos, contando con 5,280 imágenes para entrenamiento y 1,320 para prueba. También utilizaron los datos del conjunto IFN/ENIT que cuenta con 26,459 palabras arábigas con 1,120 de prueba, ambos conjuntos tienen imágenes de 28×28 pixeles a escala de grises y normalizadas. Las características obtenidas por

la red convolucional se emplearon como entrada para un clasificador SVM con kernel RBF y de esta manera poder reconocer los caracteres arábigos escritos a mano. La red neuronal convolucional es secuencial y se compone por las siguientes capas: convolución, submuestreo convolución, submuestreo, capa completamente conectada y finalmente el SVM. Con esta arquitectura obtuvieron un error en la clasificación de 6.59 % en los datos de prueba del conjunto HADCB y un 7.32 % con el conjunto IFN/ENIT.

En la investigación de P. Simard *et al.* [33] se utilizó una CNN para extraer características de las imágenes del conjunto MNIST. La primeras dos capas de la red son de convolución, ambas con un tamaño de filtro de 5×5. De la primer capa se obtienen seis mapas de características de 13×13 pixeles mientras que la segunda capa cuenta con 50 mapas de características de 5×5. Enseguida se tiene una capa completamente conectada de 100 neuronas que se conecta completamente a la última capa con 10 neuronas. Los autores obtuvieron un error de 0.4 % con un preprocesamiento en las imágenes, generando más imágenes para su entrenamiento mediante distorsiones elásticas.

Ranzato *et al.* [34] utilizaron una CNN para clasificar caracteres, la entrada a la red son imágenes del conjunto de datos MNIST. A estas imágenes se les rellenó con ceros hasta obtener imágenes de 32×32 pixeles. La primer capa son 50 filtros de 7×7 con una función de activación sigmoidal, dando como resultado 50 mapas de caracteristicas de 28×28. Posteriormente viene una capa de máxima agrupación, reduciendo los mapas de características a 14×14. La tercer capa es de convolución con función de activación sigmoidal, cuenta con 128 filtros de tamaño 5×5, conectando los 50 mapas de características de la agrupación máxima con los 128 mapas de características de la tercer capa, teniendo estos últimos un tamaño de 10×10. Enseguida se utilizó una capa de máxima agrupacion, reduciendo los mapas de características a 5×5. Una vez extraídas las características de cada imágen, se conectó la capa cuatro con una capa completamente conectada con 200 neuronas. El resultado de la capa

precedente se envió a la última capa de la red, la cuál está completamente conectada con la anterior y es en esta donde se obtienen los resultados. El mejor desempeño que obtuvieron Ranzato *et al.* fue de un 0.62 %.

Una de las redes convolucionales que ha marcado la base para la arquitectura de estas redes, se le conoce como *LeNet-5*, desarrollada por LeCun *et al.* [23]. Este modelo fue entrenado con el conjunto MNIST, a diferencia de los datos originales, se reescalaron las imágenes a 32×32 pixeles. Estas imágenes son binarias y se normalizaron de tal manera que el fondo tenga un valor de -0.1 y los pixeles del caracter un valor de 1.175, para acelerar el proceso de entrenamiento. La red cuenta con 7 capas, la primera es de convolución y tiene 6 mapas de características, cada mapa cuenta con un filtro de 5×5 pixeles que se traslada a lo largo y ancho de la imágen realizando operaciones de convolución. La segunda capa es de submuestreo reduciendo la salida de cada mapa de características en la capa anterior, a un nuevo mapa de características de tamaño 14×14 pixeles. La tercer capa es de convolución y tiene 16 mapas de características con filtros de 5×5. Estas operaciones de convolución se efectúan sobre la salida de la capa anterior de submuestreo. Enseguida se emplea otra capa de submuestreo con 16 mapas de características de 5×5, obtenidos de aplicar 16 filtros de 2×2 sobre la capa de convolución anterior. La última capa de convolución cuenta con 120 mapas de características de 5×5 con filtros de 1×1 aplicados sobre la capa anterior de submuestreo. Luego de formar el vector de características con los mapas de la capa anterior, se utilizó una capa completamente conectada con 84 neuronas. Esta capa se conectó a otra capa completamente conectada con 10 neuronas, obteniendo el resultado de la clasificación. Con este modelo se obtuvo un error del 0.95 %.

En 2017, Boufenar *et al.* [35] hablaron de tres maneras de entrenar una red neuronal convolucional para el reconocimiento de 40 caracteres arábigos contenidos en el conjunto OIHACDB-40. La red convolucional utilizada lleva el nombre de AlexNet [36] y se entre-

nó de tres maneras distintas: i) inicializando los pesos de la red con valores aleatorios; ii) pre-entrenar la red con imagenes del conjunto ImageNet y usarla como extractor de características para entrenar el clasificador y iii) nuevamente pre-entrenar la red con las imágenes del conjunto ImageNet, la diferencia es que ajustaron de manera mínima todos los pesos. Con estos enfoques y un preprocesamiento (recorte y normalizado), los autores obtuvieron un 100 % de acertividad en la red entrenada desde cero, un 86.96 % utilizando la red como extractor de características y un 98.41 % con el ajuste mínimo.

En la Tabla 1.1 se presenta una comparativa entre los métodos descritos con anterioridad. En la primer columna se mencionan los autores, en la segunda columna se indica el método que emplearon para la clasificación, en la tercer columna se enlista si hubo preprocesamiento y de que tipo, en la columna cuatro se hace alusión al conjunto de datos que utilizaron los autores para entrenar y probar sus métodos, finalmente en la quinta columna se hace referencia a los resultados obtenidos por los autores.

Tabla 1.1: Comparativa entre distintos métodos utilizados para aplicaciónes de OCR.

Autores	Métodos utilizados	Preprocesamiento	Conjunto de datos	Error
LeCun *et al.* [23] (1998)	NN (lineal)	-	MNIST	12 %
	NN (lineal)	Alineación vertical	MNIST	8.4 %
	MLP (300)	-	MNIST	4.7 %
	MLP (1000)	-	MNIST	4.5 %
	MLP (300)	Distorsiones elásticas	MNIST	3.6 %
	MLP (1000)	Distorsiones elásticas	MNIST	3.8 %
Zhang *et al.* [27] (2006)	K-NN con SVM	-	MNIST	1.66 %
			USPS	4.285 %
Cardoso y Wichert [16] (2013)	MTC y *k-means* con SVM	-	MNIST	0.71 %
			USPS	2.64 %
Boukharouba y Bennia [31] (2017)	SVM-RBF	-	Dígitos Farsi	1.52 %
Elleuch *et al.*[32] (2016)	CNN con SVM-RBF	-	HACDB	6.59 %
			IFN/ENIT	7.32 %
Simard *et al.* [33] (2003)	CNN-MLP	Distorsiones elásticas	MNIST	0.4 %
Ranzato *et al.* [34] (2007)	CNN-MLP	-	MNIST	0.62 %
LeCun *et al.* (1998)	CNN-MLP	-	MNIST	0.95 %
Boufenar [35] *et al.* (2017)	CNN-MLP	Recorte y normalizado	OIHACDB-40	0.0 %
	CNN-MLP (TA-EC)		OIHACDB-40	13.04 %
	CNN-MLP (TA-AM)		OIHACDB-l40	1.59 %

Basándose en la literatura, las CNNs han sido muy utilizadas para tareas de reconocimiento de caracteres y objetos. Existe una mayor cantidad de trabajos que utilizan CNNs, tal y como se muestra en [37] donde utilizan una CNN multicolumna para el reconocimiento de señales de tránsito, caracteres latinos y arábigos, imágenes estéreo de objetos, entre otros; en el trabajo de Jarrett *et al.* [38] se realiza una clasificación en varios conjuntos de datos que abarcan objetos inanimados, animales, juguetes de personas y caracteres; en la investigación de Bai *et al.* [39] se entrenaron tres redes convolucionales para reconocer caracteres del alfabeto inglés en letreros o anuncios encontrados en la calle y un conjunto de caracteres chinos extraídos de programas de televisión, por otro lado, en [40] se realizó una transferencia de aprendizaje a una CNN para detectar múltiples objetos, personas y animales contenidos en una imagen.

Es por esto, que las CNNs son una excelente herramienta para utilizarse en el reconocimiento de los trazos del BGT, de esta manera, dar inicio a un instrumento que auxilie al terapeuta para la revisión del examen. Es importante destacar que a la fecha no se han encontrado sistemas que empleen aprendizaje máquina o técnicas similares para identificar los trazos del BGT.

Para la implementación de este sistema, se realizaron diversas tareas, comenzando por la adquisición de las imágenes correspondientes a los trazos del BGT, así como el preprocesamiento de las imágenes para eliminar posible ruido. Una vez teniendo un conjunto de datos BGT, se implementaron arquitecturas de aprendizaje profundo, utilizando el conjunto BGT y se ajustaron los parámetros necesarios para que el sistema reconozca los trazos.

2

Marco teórico

En éste capitulo se desarrollarán los temas necesarios para entender la teoría que va a fundamentar el presente trabajo de investigación. Desde el preprocesamiento y aprendizaje máquina hasta el funcionamiento de las redes convolucionales, así como los parámetros necesarios para su correcta implementación.

2.1. Preprocesamiento

Al momento de digitalizar una imagen, hay ciertas variaciones que no se pueden evitar, como ruido, distorsión, información no contenida originalmente, etcétera. Es por esto que es necesario realizar un preprocesamiento, el cuál mejora la imagen para un futuro análisis [41]. Existen varios métodos para llevar a cabo el preprocesamiento en imágenes, principalmente se utilizan filtros u operadores, los que al aplicarse a la imagen, realzan o atenuan características sin aumentar la información inherente contenida en la imagen [42]. Entre los filtros más conocidos se encuentra el Laplaciano, Gaussiano, Sobel [43], Prewitt [44], el algoritmo Canny [45], entre otros.

2.1.1. Filtro Prewitt

Este filtro surge del intento de dar solución a diversos problemas que los filtros lineales no podían resolver en ese entonces [44]. Para el presente proyecto, se utilizó el filtro para detectar bordes en los trazos digitalizados. El filtro se compone por dos máscaras de detección, vertical y horizontal, ambas se muestran en la ecuación 2.1, dichas máscaras se convolucionan con la imagen que contiene todos los trazos en escala de grises $\mathbf{I}_g(x, y)$.

$$\mathbf{G}_r(x,y) = \frac{1}{3}\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} * \mathbf{I}_g(x,y) \qquad \mathbf{G}_c(x,y) = \frac{1}{3}\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix} * \mathbf{I}_g(x,y) \quad (2.1)$$

La magnitud del filtro Prewitt se define como $\mathbf{G}(x, y) = \sqrt{\mathbf{G}_r^2 + \mathbf{G}_c^2}$. Si la magnitud sobrepasa un umbral, el pixel se clasifica como borde $\mathbf{E}(x, y)$.

2.1.2. Algoritmo Canny

El algoritmo Canny es utilizado para detectar bordes, a diferencia del filtro Prewitt, Canny definió una serie de pasos para detectar los bordes contenidos en una imagen.

Paso 1: Se convoluciona un filtro gaussiano con la imagen para filtrar ruido.

Paso 2: Se aplica una máscara de Robert o Sobel para encontrar el gradiente de la imagen.

Paso 3: Encontrar la dirección del gradiente, utilizando:

$$\theta = \tan^{-} 1 \left(\frac{G_y}{G_x} \right) \quad (2.2)$$

Paso 4: Una vez conocido el ángulo del borde, se relaciona con el grado específico.

Paso 5: Adelgazar los bordes obtenidos con el gradiente hasta un pixel de ancho, aplicando supresión no máxima.

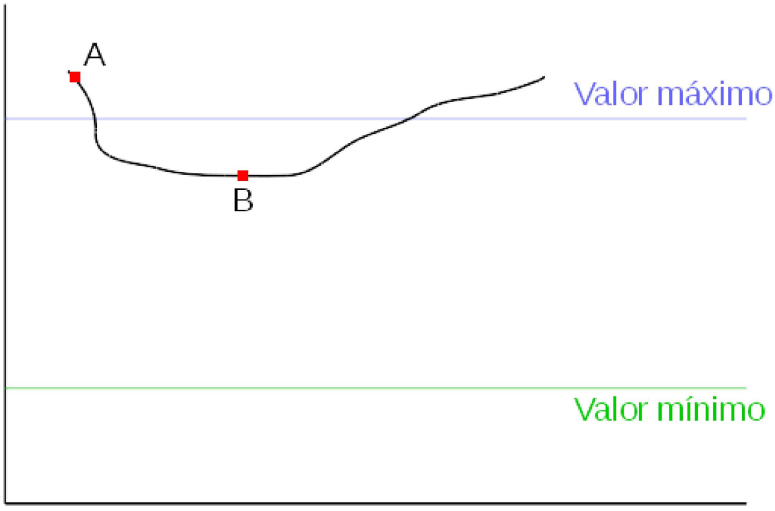


Figura 2.1: Representación de dos pixeles definidos como borde aunque el pixel B no pase el umbral de valor máximo.

Paso 6: Utilizar histéresis de umbral para reducir la aparición de rayas que no son bordes.

Para este último paso, se definen dos umbrales, uno máximo y uno mínimo, todo pixel que sobrepase el umbral máximo se considera borde, por el contrario, si se encuentra por debajo del mínimo, no se considera borde. Lo importante es que si un pixel se encuentra en medio de los dos umbrales, se toma en cuenta si el pixel está conectado con un pixel borde o no; por lo que si un pixel tiene un valor entre los umbrales y está conectado con algún pixel borde, éste tambien será considerado borde. En la Figura se muestra un ejemplo de histéresis de umbral en el algoritmo de Canny.

2.2. Aprendizaje máquina

Dentro de la ciencia computacional, existe un campo llamado aprendizaje máquina, el cuál se enfoca en comprender fundamentalmente el mundo que nos rodea. Para esto, la máquina debe aprender y desenredar los factores explicatorios ocultos en el medio observado de los datos sensoriales de bajo nivel [46].

El aprendizaje máquina consiste en hacer que las máquinas modifiquen o adapten sus acciones de manera que esas acciones se vuelvan más precisas, midiéndose la precisión en que tan bien se escojan las acciones correctas [47].

2.3. Redes neuronales artificiales

Las redes neuronales artificiales (ANN, por sus siglas en inglés, *Artificial Neural Networks*) han dado solución a tareas que hace décadas no se sabía como atacar, ya que estas redes se entrenan con ejemplos, aprenden representaciones de manera abstracta de la información que se les presenta. El primer modelo matemático de una ANN fue presentado por McCulloch y Pitts [48]. En esta ANN se multiplican las entradas x_i por los pesos w_i. Este modelo matemático se representa con la ecuación 2.3.

$$h = \sum_{i=1}^{m} w_i x_i \tag{2.3}$$

Las neuronas suman los valores de la multiplicación $w_i \cdot x_i$ y el resultado de la suma se compara con un umbral. Si la salida es mayor al umbral θ la neurona es activada con un 1, en caso contrario, no se activa y se tiene un 0 a la salida. Este umbral se encuentra en la ecuación 2.4.

$$y = y(h) = \begin{cases} 1, & \text{si } h > \theta \\ 0, & \text{si } h \leq \theta \end{cases} \tag{2.4}$$

Sin embargo, este modelo no resuelve problemáticas más complejas. Es por esto que surge el perceptrón, el cuál es básicamente una neurona de McCulloch y Pitts a la que se le actualizan los pesos conforme al error obtenido y el valor del peso anterior [47] con la ecuación 2.5.

$$w_{ij} \leftarrow w_{ij}^{t-1} + \eta(y_j - T_j) \cdot x_i \tag{2.5}$$

Donde w_{ij} es el peso de la entrada i conectada con la neurona j, w_{ij}^{t-1} es el peso anterior de la entrada i conectada con la neurona j. El valor de η indica la tasa de aprendizaje (*learning rate*), la cuál señala que tan rápido o lento se entrena el perceptrón. La y_j es el resultado de la neurona j. Mientras que T_j es el resultado deseado (*Target*) de la neurona j. Es aquí, en los pesos, en donde se encuentra el aprendizaje de la red.

Además, cuenta con un bias que sirve para ajustar el umbral y que las neuronas se puedan activar aunque las entradas tengan un valor de cero [47].

Si se agregan capas de perceptrones entre la capa de entrada y de salida, se crea un perceptrón multi capa (MLP, por sus siglas en inglés, *Multi-layer Perceptron*).

2.4. Retropropagación del error

Como se vió con anterioridad, el aprendizaje de las redes se encuentra en los pesos. Para poder variar los pesos y minimizar el error, es necesario conocer cuáles son los pesos que más error producen. Para esto, se obtiene el error por la diferencia de suma de cuadrados representado por la ecuación 2.6.

$$E(\mathbf{t}, \mathbf{y}) = \frac{1}{2} \sum_{k=1}^{N} (y_k - T_k)^2 \tag{2.6}$$

La idea es derivar la ecuación 2.6, sin embargo, solo se puede derivar con respecto a los pesos, ya que son las únicas variables que podemos modificar. Por lo que se sustituye en dicha ecuación para dejar los pesos como variable. La sustitución se observa en la ecuación 2.7.

$$E(\mathbf{w}) = -\sum_{k=1}^{N} \left[g\left(\sum_{j=0}^{M} w_{jk} a_j \right) - T_k \right]^2 \tag{2.7}$$

La ecuación 2.7 es diferenciable con respecto a los pesos mediante regla de la cadena para obtener un gradiente. Éste gradiente indica la dirección en la que se debe disminuir el error para dirigirse a un error mínimo, ya sea local o global; el gradiente se calcula con la suma del error obtenido en todos los datos. En la figura 2.2 se representa el comportamiento esperado utilizando dicha estrategia que lleva por nombre **gradiente descendente** (GD, por sus siglas en inglés *gradient descent*). Este error se calcula desde la última capa, pasando por las capas ocultas hasta llegar a la primera. Tal procedimiento es llamado **retropropagación** (*backpropagation*). Existen variantes del GD, la que se utilizó en el presente trabajo es llamada gradiente descendente estocástico por mini lotes (*mini-batch* SGD, por sus siglas y traducción del inglés, *Stochastic Gradient Descent*). A diferencia del GD, en éste se calcula y se actualizan los pesos cada cierta cantidada de datos (mini lote) definido por el usuario durante el entrenamiento que normalmente toma un valor entre 2 y 256.

Un parámetro útil cuando se utiliza SGD por mini lotes es el momento; el momento ayuda a no estancarse en mínimos locales ya que análogamente, cuando se reduce el error en cierta dirección, éste lleva cierto impulso, el cuál provoca que el comportamiento de convergencia hacia el mínimo global se vea menos afectado por mínimos locales. El momento ayuda a mejorar la celeridad del aprendizaje [49], generalmente se utiliza la letra α y toma un valor entre 0 y 1, $\alpha \in [0, 1]$ [50].

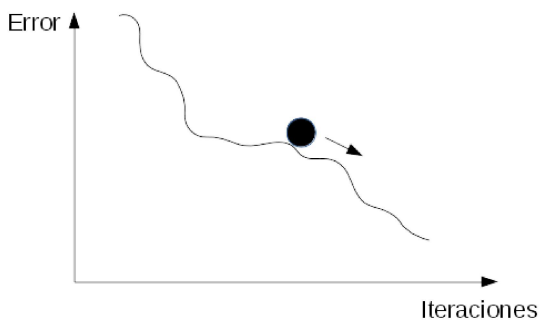


Figura 2.2: Representación del comportamiento esperado del error.

Para fines de la presente investigación, se sustituye la función de activación (2.4) por una función rectificada linealmente (ReLU, por sus siglas en inglés, *Rectified linear unit*), la razón de hacer esto, es que la función ReLU no es lineal y es fácilmente derivable. Los valores que toma la función ReLU se muestran en la ecuación 2.8.

$$y = g(h) = \begin{cases} h, & \text{si } h > 0 \\ 0, & \text{si } h \leq 0 \end{cases} \tag{2.8}$$

Es también agregada la función de activación *soft-max*, la cuál se utiliza en la última capa de la red. La función se define con la ecuación 2.9.

$$y_k = g(h_\kappa) = \frac{e^{h_\kappa}}{\sum_{k=1}^{N} e^{h_k}} \tag{2.9}$$

Esta función generalmente se utiliza en problemas de clasificación, entrega un valor entre 0 y 1, donde se divide el exponencial de la neurona κ entre la sumatoria de las exponenciales de la suma h de cada neurona k. Entregando el porcentaje de pertenecer a la clase.

Se cambia la función de error 2.6 para que tenga la forma de entropía cruzada (*cross-entropy*) como se presenta en la ecuación 2.10. Ésta función es útil ya que utiliza las probabilidades calculadas con la función *soft-max* y en base a ellas se calcula el error.

$$E_{ce} = -\sum_{i=1}^{N} T_i \ln(y_i) \tag{2.10}$$

La derivada parcial del error E_{ce} con respecto a los pesos se simboliza con la letra δ_o. Al momento de utilizar la función *soft-max* en la última capa de la red, se obtiene la derivada en la ecuación 2.11 como.

$$\delta_o(i) = y_i - T_i \tag{2.11}$$

Una vez obtenido el gradiente del error y agregando el término del momento, se pueden actualizar los pesos de las capas conforme a la siguiente ecuación.

$$w_{ij} \leftarrow w_{ij}^{t-1} + \eta\delta(i)y_i + \alpha\Delta w_{ij}^{t-1} \tag{2.12}$$

Donde w_{ij} es el peso a actualizar de la entrada i con la neurona j, w_{ij}^{t-1} es el peso anterior de la entrada i con la neurona j, η es la tasa de aprendizaje, $\delta(i)$ es la derivada del error obtenido, y_i es el resultado de activación de la neurona i, α es el momento y Δw_{ij}^{t-1} es la actualización previa que se le hizo a los pesos [47].

2.5. Aprendizaje Profundo

Siendo una rama del aprendizaje máquina, el aprendizaje profundo se caracteriza por la gran cantidad de parámetros y los múltiples niveles de operaciones no lineales que contiene cada red. Cada nivel de la red busca extraer información de los datos que se le presenten, obteniendo funciones complicadas que pueden representar abstracciones de alto nivel [51].

Las técnicas que abarcan el aprendizaje profundo han mejorado notablemente los resultados obtenidos por algoritmos de aprendizaje máquina [52]. Estas técnicas tienen la ventaja de no necesitar una ingeniería de características para manipular los datos y darles un preprocesamiento. Tampoco es imprescindible conocer cuáles características aportan mayor ganancia de información. Por ejemplo, no se necesita ser un experto en técnicas de procesamiento de imágenes para crear un programa que identifique algún objeto en una imagen. Incluso, no es necesario tener conocimientos acerca del objeto que se busca. Con las desventajas de necesitar más poder de procesamiento, memoria y almacenamiento dada la basta cantidad de datos que exige la red para entrenarse. Esto se debe a que las técnicas de aprendizaje profundo son computacionalmente costosas. Entre las técnicas de aprendizaje profundo destaca la máquina

de Boltzmann, redes neuronales recurrentes, autocodificadores, redes neuronales profundas de creencias y redes neuronales convolucionales (CNNs) [53]. Enseguida se describirá en que consiste y cuáles son los parámetros fundamentales que distinguen las CNNs de otras redes neuronales.

2.6. Redes Neuronales Convolucionales

Se les llama redes neuronales convolucionales a las redes neuronales artificiales que contienen al menos una capa de convolución [54]. Estas capas se utilizan para extraer información que se encuentra en los datos, principalmente imágenes, permitiendo al investigador analizar datos en crudo, sin dejar de lado un buen desempeño. Este tipo de redes surgió de la idea de reducir los parámetros variables en las redes [55], una opción, fue compartir variables entre neuronas [56], esto causa que la capa de convolución tenga equivalencia al traslado [54].

La manera en la que operan las capas de convolución es aplicando distintos filtros (*kernels*) a lo largo y ancho de la imágen. Se realizan varias operaciones de convolución entre el filtro y la imágen hasta abarcar toda el área de la imágen. El resultado de cada operación se almacena en una matriz llamada mapa de características, cada mapa corresponde al resultado de utilizar filtros con distintos valores [57]. Cabe destacar que los valores de cada filtro se modifican conforme se entrena la red. En la Figura 2.3 se muestran 96 filtros de tamaño $11 \times 11 \times 3$ pixeles. Estos filtros se obtuvieron de la primer capa de convolución después del entrenamiento de una red desarrollada por Krizhevsky *et al.* [36] para un reto llamado ILSVRC-2010 [58]. Este reto se basa en que los concursantes deben desarrollar métodos para que la computadora encuentre objetos en fotos o videos, además de la clasificación de los objetos. Entre los objetos a detectar y clasificar se encuentran 1,000 categorias. En estas categorias se encuentran animales, plantas y objetos sin vida, tales como perros, hongos, barcos,

botellas, artículos de oficina, entre otros.

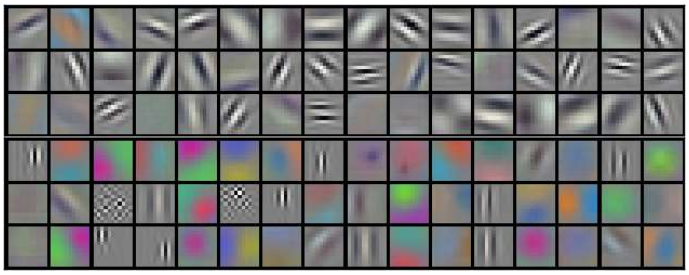


Figura 2.3: Filtros obtenidos por Krizhevsky en la primer capa de convolución para el reto ILSVRC-2010.

Existen tres parámetros principales que se utilizan al implementar CNNs, estos son: tamaño de filtro (*kernel size*), paso (*stride*) y relleno (*padding*).

Los filtros son los responsables de interactuar con la imagen mediante la operación de convolución y su tamaño se indica en pixeles. Por lo general se utilizan filtros de tamaño impar, como 1×1, 3×3, 5×5, 7×7 y 11×11 pixeles en imágenes a escala de grises o binarias. En la Figura 2.4 se muestran ejemplos de distintos tamaños de filtros.

El segundo parámetro es el de paso. Éste define la cantidad de pixeles que se va a desplazar el filtro de convolución. Si el valor de paso es mayor o igual a 2, ayuda a reducir la dimensionalidad de los mapas de características.

El parámetro relleno, señala si se agregan o no se agregan filas y columnas con valor cero alrededor de la imagen. Con el propósito de modificar las dimensiones del mapa de características y obtener una representación distinta con cada filtro de convolución. El relleno puede ser válido, idéntico o completo. En la Figura 2.5 se ilustran distintos valores del relleno con un filtro de 3×3.

Además, generalmente se utiliza una capa de agrupación, ésta es una capa en la red que precede a la capa de convolución y busca reducir la dimensionalidad del mapa de características. Esto se hace aplicando una ventana sobre el mapa y realizando distintas operaciones

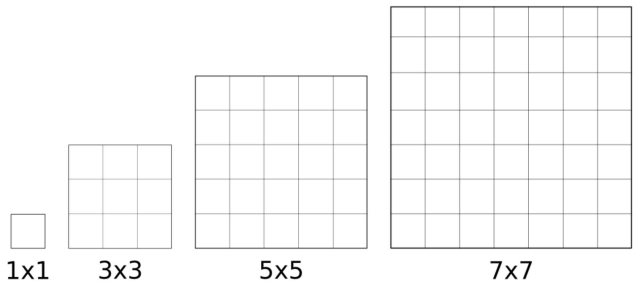


Figura 2.4: Ejemplos de filtros de distintas dimensiones en pixeles.

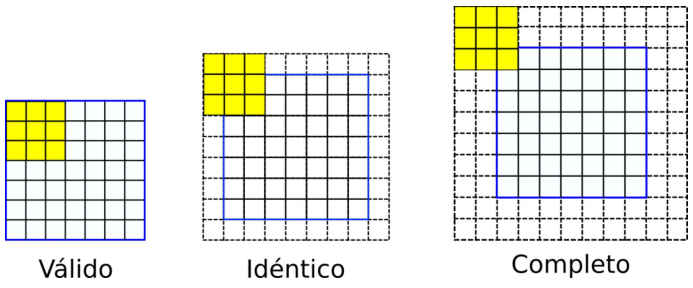


Figura 2.5: Ejemplos de relleno con diferentes valores usando un filtro de 3×3 pixeles.

como pueden ser la suma, resta, multiplicación, división, etcétera. Entre las funciones más utilizadas está el valor máximo (*max pooling*) y el promedio (*average pooling*).

Estos parámetros son importantes debido a que muestran como se va a comportar la capa de convolución, afectando las dimensiones de las salidas, así como el desempeño.

2.7. Pruebas estadísticas

Las pruebas estadísticas que se realizaron son dos, Prueba de Friedman [59] y la Prueba de Nemenyi [59]; éstas pruebas ayudan a determinar si hay diferencia estadística en el desempeño o no, entre los modelos probados y que tanta es la diferencia entre ellos. La prueba de Friedman es una prueba de rangos, donde se comparan los valores jerárquicamente obtenidos según el desempeño del modelo; bajo la hipótesis nula (2.13), ésta indica que todos los modelos son equivalentes y por lo tanto sus rangos tambien lo son.

$$H_0 : R_1 = R_2 = ... = R_{J-1} = R_J \tag{2.13}$$

Donde $R_j = \sum_i r_i^j$ es la suma de los rangos obtenidos por los métodos en cada iteración [60]. La estadística de Friedman (2.14) se distribuye conforme a χ^2 con $k-1$ grados de libertad.

$$F_r = \left[\frac{12}{Nk(k+1)} \sum_{j=1}^{k} R_j^2 \right] - 3N(k+1) \tag{2.14}$$

Donde N es la cantidad de iteraciones y k el número de modelos utilizados en dichas iteraciones. En caso de existir empate entre modelos, se asigna el promedio de los dos rangos que pudieran tomar dichos modelos (*e.g.* si los modelos cayeran entre el rango 5 y 6, se asigna un 5.5 para cada uno). De la misma manera, si hay modelos empatados en las iteraciones, se sustituye la Ecuación 2.14 por la Ecuación 2.15 que se presenta a continuación.

$$F_r = \frac{12 \sum_{j=1}^{k} R_j^2 - 3N^2 k(k+1)^2}{Nk(k+1) + \frac{\left(Nk - \sum_{i=1}^{N} \sum_{j=1}^{g_i} t_{i,j}^3 \right)}{(k-1)}} \tag{2.15}$$

g_i = número de conjuntos de rangos empatados en el i-ésimo grupo $t_{i,j}$ = el tamaño del j-ésimo conjunto de rangos empatados en el i-ésimo grupo Si el resultado de (2.14) es mayor que el valor de χ^2 se rechaza la hipótesis nula y se realiza la Prueba de Nemenyi, es una prueba *post-hoc* para calcular la distancia crítica (CD, por sus siglas en inglés, *Critical Difference*), ésta prueba nos permite conocer que tanto se parece el desempeño de los modelos, estadísticamente hablando.

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}} \tag{2.16}$$

Donde q_α es el valor crítico encontrado en la Tabla 5(a) presentada por Demšar [59] el

cuál depende de la cantidad de clasificadores y el porcentaje de confianza deseado; N y k son las mismas utilizadas que en la ecuación 2.14. La ecuación (2.16) entrega el valor de la distancia crítica para poder comparar el promedio de los rangos de los modelos; si la resta del promedio de dos rangos es menor o igual a la distancia crítica (2.17), los dos rangos son estadísticamente iguales, por lo tanto, se asume que el desempeño de los métodos tambien lo son.

$$|R_i - R_j| \leq CD \tag{2.17}$$

3

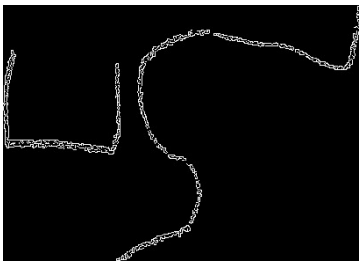
Metodología

A continuación se describirán las etapas de la metodología que se llevaron a cabo para cumplir con los objetivos del presente proyecto de investigación. Se iniciará con la población de estudio, posteriormente la recolección de los datos, su procesamiento y el diseño de los experimentos para obtener resultados.

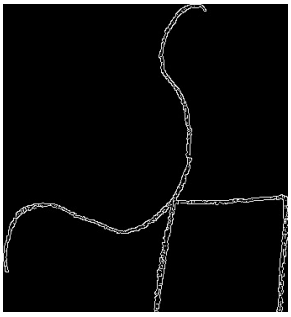
Debido a que el BGT es una prueba para explorar problemas visomotrices, se realizó un acercamiento al Instituto José David, A.C. Éste Instituto brinda tratamientos integrales para personas (principalmente niños) que presenten problemas de audición, lenguaje, autismo y otros trastornos del desarrollo. Se cuenta con una vinculación entre el Instituto y la UACH, lo que nos permitió llevar a cabo una colaboración para el presente proyecto de investigación. Además, fue necesario buscar vinculación con la primaria Alfonso N. Urueta Carrillo, perteneciente al municipio de Chihuahua con la finalidad de aumentar la cantidad de datos necesarios para la investigación.

3.1. Adquisición del conjunto de datos

Una parte de la recolección del conjunto de datos se llevó a cabo por medio del personal del Instituto José David, A.C. Estas personas aplicaron el BGT a una población de 86 niños entre 4 y 11 años. Además se aplicó el BGT en la primaria Alfonso N. Urueta Carrillo,



(a) Trazo hecho por niño del Instituto.



(b) Trazo realizado por niño de la primaria.

Figura 3.1: Muestra de trazos realizados por un niño del Instituto y un niño de la primaria.

contando con alrededor de 240 niños para tener un conjunto más amplio y diverso. La figura 3.1 muestra un ejemplo de los trazos realizados por los niños del Instituto y la Primaria.

Los datos son los trazos realizados, estos se encuentran en las áreas de interés en hojas de máquina digitalizadas mediante un escáner con una resolución de 300 pixeles por pulgada. Los trazos se separaron automáticamente utilizando detección de bordes, dilataciones, erosiones y rellenado de áreas. En base a esto se escogieron las áreas de mayor tamaño y se segmentaron los objetos de interés. Posteriormente, se extrajeron estas zonas de todo el conjunto de hojas de máquina recopiladas. Una vez se obtenidos los trazos segmentados, se etiquetaron y fueron almacenados.

A continuación se explica a detalle la etapa del pre-procesamiento de los datos.

Entonces

3.2. Pre-procesamiento

Una parte importante de los experimentos es el pre-procesamiento de los datos. Esto puede ayudar de distintas maneras, una de ellas es reducir el tiempo requerido para el entrenamiento de la red, consiguiendo resultados similares a cuando se entrena la red sin usar pre-procesamiento. Otra forma de pre-procesamiento es reducir el ruido en los datos, con una posible mejora en el desempeño ante los resultados obteniEntoncesdos del clasificador sin

pre-procesamiento. Para esta etapa, se utilizaron distintos filtros y funciones sobre los datos.

La segmentación de las imágenes es parte del pre-procesamiento, ya que en la etapa experimental es necesario contar con cada trazo de una manera individual. Dado lo mencionado con antelación, se transforma la imagen escaneada de los trazos a escala de grises para aplicar el algoritmo de Prewitt, convolucionando las máscaras de detección en 2.1 con la imagen, detectando los bordes de los trazos en la imagen que contiene los nueve trazos. Enseguida, se aplica el operador morfológico de la dilatación seguido por las funciones de rellenado de agujeros y erosión para realizar una mejor detección de los trazos. Finalmente, se ejecuta un análisis de áreas donde se encuentra aquellas regiones que tienen una alta probabilidad de pertenecer a uno de los nueve trazos. Para esto, se identifica el área con el objeto más grande, posteriormente se dividen las áreas de interés restantes entre el área con el objeto mas grande. Únicamente aquellas áreas cuyo resultado sobrepase el valor de 0.1 son consideradas como posibles secciones con un trazo del BGT. En la figura 3.2 se presenta un diagrama a bloques de dicha etapa del pre-procesamiento.

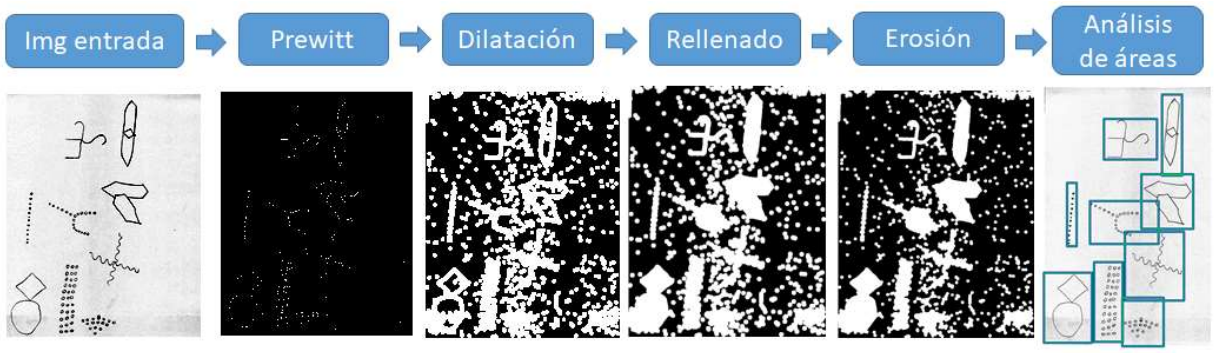


Figura 3.2: Diagrama a bloques de una correcta identificación automática de los trazos del BGT.

En caso de faltar trazos por identificar debido a ciertos factores como baja calidad del trazo o papel, que los trazos estén muy cercanos, incompletos o que exista algo escrito en la parte de atrás, haciendo difícil su correcta identificación, el usuario definirá las áreas donde se

encuentren los trazos faltantes. Como etapa final del segmentado, cada área seleccionada es recortada en la imagen original (RGB) y se le asigna un nombre aleatorio para ser almacenado en el directorio correspondiente a la case del trazo. En la Figura 3.3 se presenta un diagrama a bloques de la segmentación incorrecta del algoritmo.

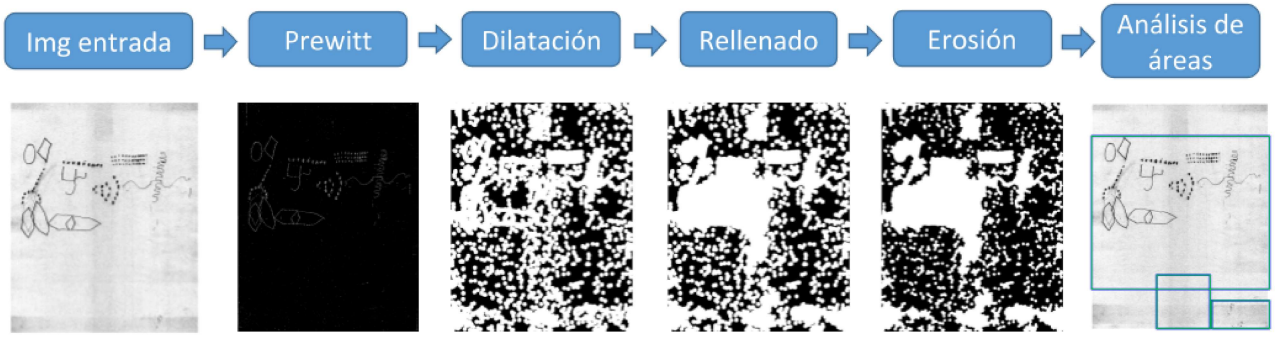
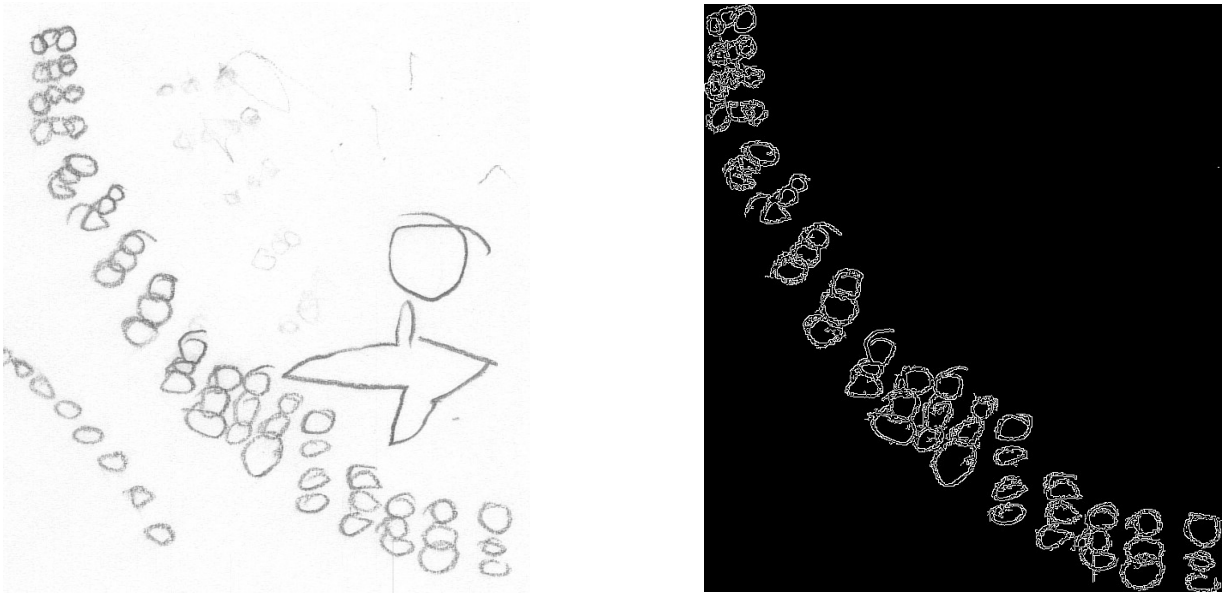


Figura 3.3: Diagrama a bloques de una incorrecta identificación automática de los trazos del BGT.

Una vez que se tienen los trazos sEntonceseparados por clases, se utiliza el algoritmo de Canny (mencionado en la sección 2.1.2) para detectar bordes, binarizar las imágenes y eliminar parcial o totalmente el ruido. Si las imágenes persisten en contener ruido (e.g. anotaciones del evaluador) se eliminan manualmente. Terminada la etapa de detección de bordes con el algoritmo de Canny, se recortan los bordes de las imágenes de tal manera que el trazo esté centrado en la imagen. En la figura 3.4 se muestra un ejemplo de una imagen original y una imagen pre-procesada y recortada.

3.3. Aumento del conjunto de datos

La mejor manera de que un modelo de aprendizaje máquina generalice es entrenándolo con más datos. Sin embargo, en la práctica, la cantidad de información disponible es limitada [54]. Es por esto, que se pueden agregar copias con modificaciones (e.g. desplazamiento, rotaciones, distorsiones elásticas, etcétera) siempre y cuando no se cambie la clase del objeto



(a) Imagen sin pre-procesamiento (b) Imagen con pre-procesamiento

Figura 3.4: Imágenes con y sin pre-procesamiento

[54].

En contraste, el conjunto de datos MNIST contiene 60,000 imágenes para entrenar, lo que corresponde a 6,000 imágenes por clase. El conjunto HACDB [61] incluye 6,600 datos con 66 clases, 5,280 para entrenar y 1,320 para probar, esto es 100 imágenes por clase. También se encuentra el conjunto OIHACDB-40 [35], éste incluye 30,000 imágenes divididas en 40 clases, teniendo 750 imágenes por clase.

En vista de las distinas investigaciones donde se han utilizado técnicas para aumentar la cantidad de datos, se adoptó por aumentar artificialmente la cantidad de datos del conjunto BGT. Principalmente se les aplicó distorsión elástica aleatoria y rotación aleatoria, generando 10 imágenes artificiales por cada imagen original. En la Tabla 3.1 se muestra la cantidad de datos antes y después de las modificaciones realizadas a las imágenes.

La primer columna indica la clase del trazo, la segunda columna señala la cantidad de imágenes originales, mientras que la tercera muestra la cantidad de imágenes extra que se han hecho a partir de modificar a las imágenes originales.

Tabla 3.1: Número de muestras para cada clase en el conjunto BGT

Trazo	# Muestras Originales	# Muestras Aumentadas
C-A	320	3200
C-1	318	3180
C-2	318	3180
C-3	314	3140
C-4	316	3160
C-5	316	3160
C-6	315	3150
C-7	315	3150
C-8	318	3180
Total	2,850	28,500

En la Figura 3.5 se presentan diez imágenes creadas artificialmente a partir de la imagen que se encuentra encima de esas diez; se utilizó rotacion, traslación y distoriones elásticas para crearlas. De manera que se parecieran lo menos posible, los parámetros de las modificaciones son aleatorios para cada imagen.

3.4. Transferencia de aprendizaje

Para realizar la transferencia de aprendizaje (TL, por sus siglas en inglés, *Transfer Learning*), es necesario entrenar la red con la arquitectura y conjunto de datos seleccionado. Una vez entrenada, se toman los pesos de la prueba en la que se haya obtenido el mejor desempeño y se reentrena la red con el conjunto de datos sobre el que se quiere trabajar. Principalmente, existen dos enfoques para realizar una transferencia de aprendizaje, utilizar la red como extractor de características (FE, por sus siglas en inglés, *Feature Extractor*) o ajustando muy poco los pesos (FT, por sus siglas en inglés, *Fine Tuning*).

3.4.1. Extractor de características

Cuando se utiliza este enfoque, se cargan los pesos de la red previamente entrenada y por lo general, se remueve la última capa y se agrega otra capa con tantas neuronas como cantidad

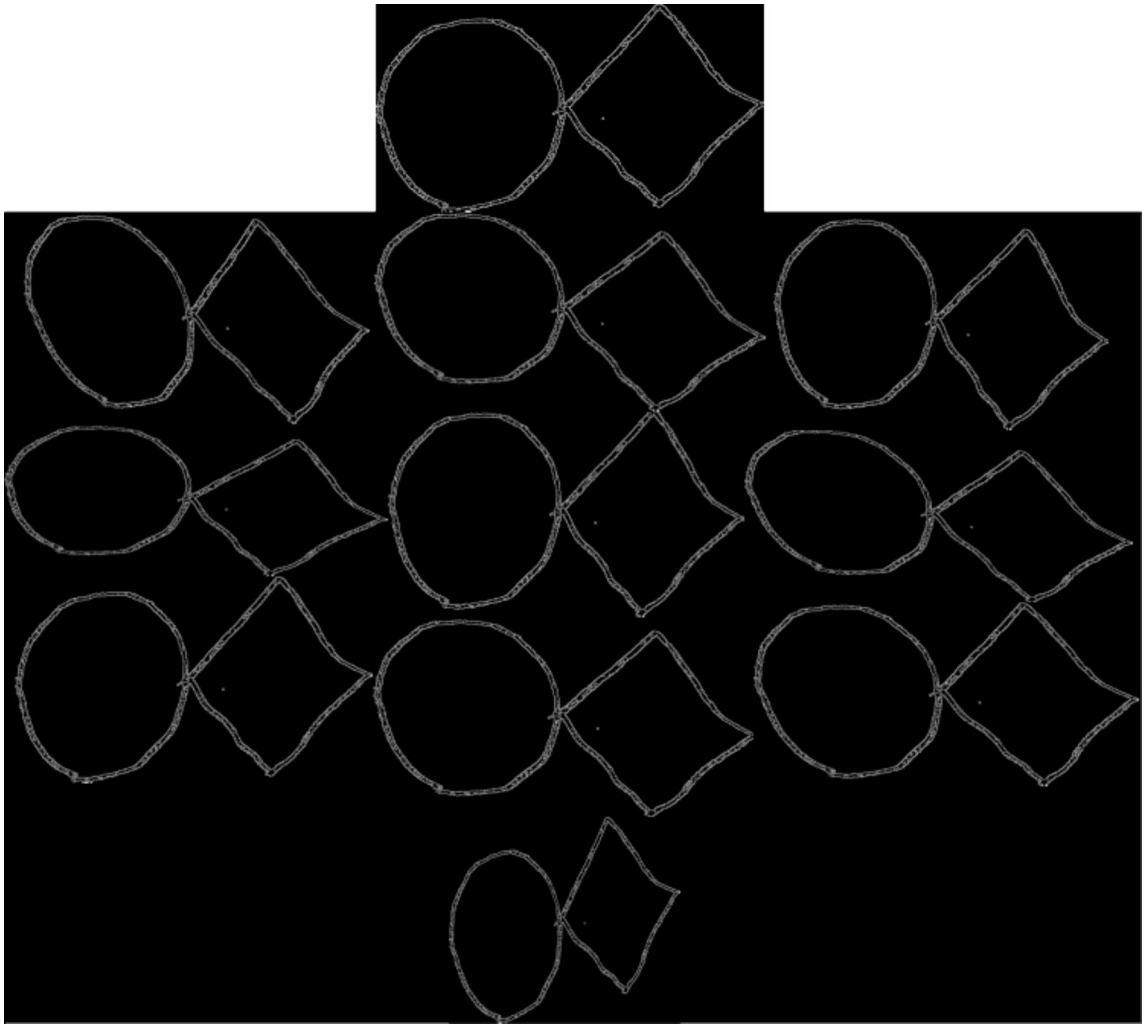


Figura 3.5: Imagen original con sus respectivas distorsiones elásticas.

de clases que se encuentran en el nuevo conjunto, debido a que no necesariamente se tiene la misma cantidad de objetos a clasificar. Una vez hecho esto, es necesario remover el parámetro de entrenamiento de cada capa de la red que no se desee entrenar (capas convolucionales). Teniendo lo anterior, se procede a entrenar la red con el nuevo conjunto de datos.

3.4.2. Ajuste mínimo de los pesos

El segundo enfoque de transferencia de aprendizaje tomado en esta investigación, consiste en cargar los pesos de la red entrenada con el conjunto de datos escogido por el usuario y entrenar de nuevo la red con un conjunto distinto. Lo importante de este enfoque es que la

taza de aprendizaje sea menor para que los pesos se modifiquen lentamente y en teoría, se aproveche el conocimiento obtenido con el conjunto de datos anterior.

3.5. Desarrollo de experimentos

En la primer etapa se realizaron experimentos con un MLP para tomarlo de referencia, así como experimentos con distintas arquitecturas de CNNs utilizando los datos del conjunto BGT. Las pruebas se realizaron lo más uniforme posible, de tal manera que los experimentos conserven los parámetros de la red, se entrenen y prueben con los mismos datos. Además de una comparación estadística de los resultados.

Durante esta fase de experimentos, se entrenaron cinco redes desde cero, inicializando los pesos de una manera aleatoria. El entrenamiento se llevó a cabo utilizando el conjunto de datos BGT con datos originales y aumentados durante el entrenamiento, mientras que las pruebas se realizaron únicamente con datos originales.

La segunda fase de experimentos consiste en efectuar una transferencia de aprendizaje; tomando como base la red en la que se haya obtenido el mejor resultado promedio puntual durante la primer fase. Dicha red se entrenó con los conjuntos MNIST y OIHACDB-40. Terminado el entrenamiento con estos conjuntos, se almacenaron los pesos para utilizarlos en los experimentos siguientes. En esta fase, se utilizó la red como extractor de características y se le realizó un ajuste mínimo a los pesos. El reentrenamiento se desempeñó utilizando los datos del conjunto BGT.

3.5.1. Implementación de arquitecturas

Para la implementación, se selccionaron cinco arquitecturas, la primera es de un MLP, la segunda es una CNN sencilla (CNN1), la tercera es similar a *LeNet*-5 (CNN2), la cuarta lleva por nombre *AlexNet* (CNN3) y la quinta es propuesta en este trabajo de investigación

(CNN4). Las redes se implementaron con el lenguaje de programación *python*, ayudado de Keras, un API de redes neuronales de alto nivel [62].

Arquitecturas

Se utilizará cierta notación para representar cada capa de las redes. *K*@*d*-***C****s* refiere a una capa convolucional con *K* kernels de dimensión *d* y un paso de *s* pixeles. M**P**s indica una capa de *maxpooling* con una ventana de $M \times M$ con paso de s pixeles y n**FC** denota una capa completamente conectada con n neuronas.

Se describirá la arquitectura de cada red en el orden mencionado en la sección 3.5.1.

MLP: 512**FC** ReLU, 512**FC** ReLU, 9**FC** SoftMax.

CNN1: 20@*5*-***C1*** ReLU, 2**P**2, 9**FC** SoftMax.

CNN2: 6@*5*-***C1*** ReLU, 2**P**2, 16@*5*-***C1*** ReLU, 2**P**2, 120@*1*-***C1*** ReLU, 84**FC** ReLU, 9**FC** SoftMax.

CNN3: 96@*11*-***C1*** ReLU, 2**P**2, 256@*5*-***C1*** ReLU, 2**P**2, 384@*3*-***C1*** ReLU, 384@*3*-***C1*** ReLU, 4096**FC** ReLU, 4096**FC** ReLU, 9**FC** SoftMax.

CNN4: 48@*11*-***C1*** ReLU, 2**P**2, 128@*7*-***C1*** ReLU, 2**P**2, 256@*3*-***C1*** ReLU, 1024**FC** ReLU, *Dropout*, 1024**FC** ReLU, 9**FC** SoftMax.

A continuación se complementará la descripción de las arquitecturas con un diagrama de cada una de ellas, así como las dimensiones de los tensores después de realizarse las operaciones en dicha capa de la red.

En la Figura 3.6 se presentan las capas que componen el MLP utilizado como base para comparar los resultados.

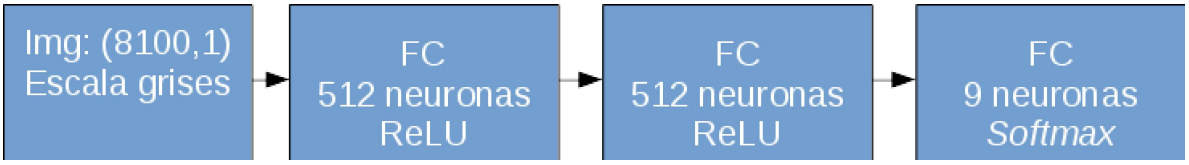


Figura 3.6: Diagrama de las capas que conforman el MLP.

En la Figura 3.7 se muestra que la CNN1 consta de una capa de convolución, *maxpooling* y las nueve neuronas pertenecientes al número de clases.

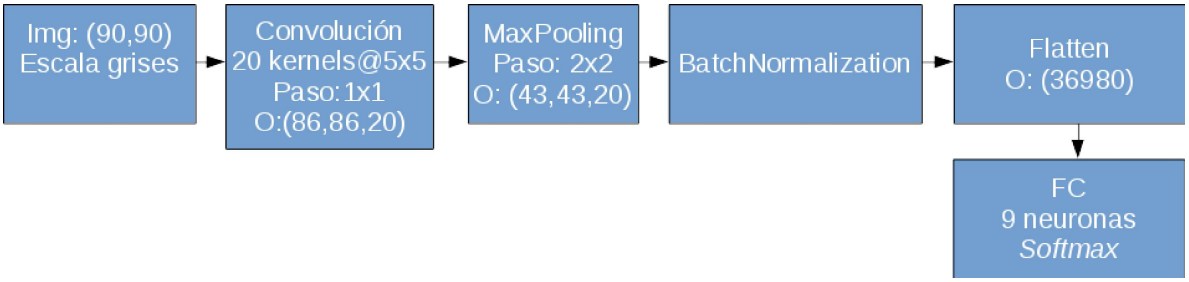


Figura 3.7: Diagrama de las capas que conforman la CNN1.

Por otro lado, en la Figura 3.9 se expone el diagrama perteneciente a la CNN2, la cuál se constituye de tres capas de convolución y una de neuronas completamente conectadas.

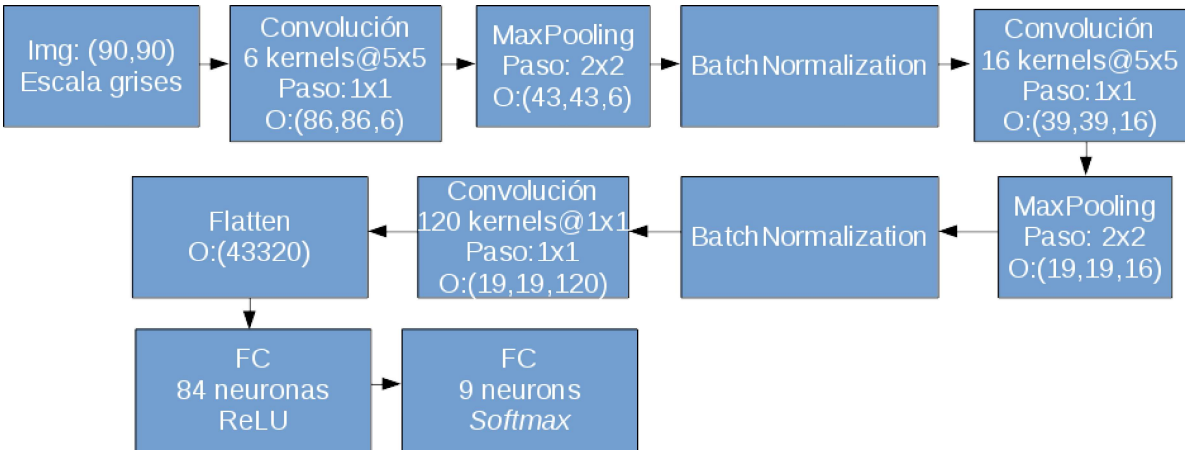


Figura 3.8: Diagrama de las capas que conforman la CNN2.

La arquitectura de la cuarta red se exhibe en la Figura 3.9, es la más compleja de las cinco redes (hablando de parámetros), posee cinco capas convolucionales con una gran cantidad de mapas de características y dos capas completamente conectadas que contienen arriba de

4,000 neuronas.

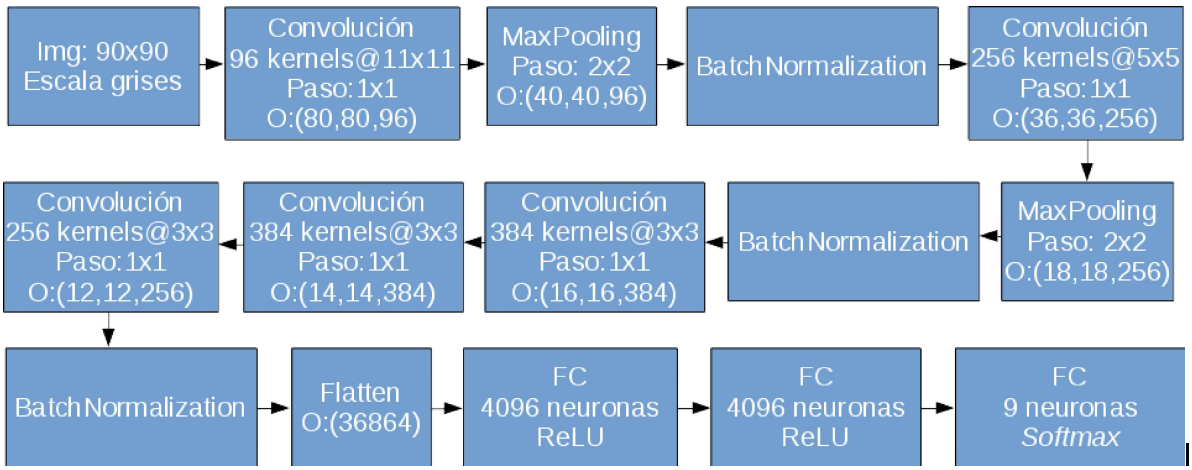


Figura 3.9: Diagrama de las capas que conforman la CNN3.

Por último, se representa la arquitectura de la red CNN4, ésta red se diseñó con el propósito de tener un desempeño similar al de la red CNN3 a un costo computacional menor (que cuente con menos parámetros).

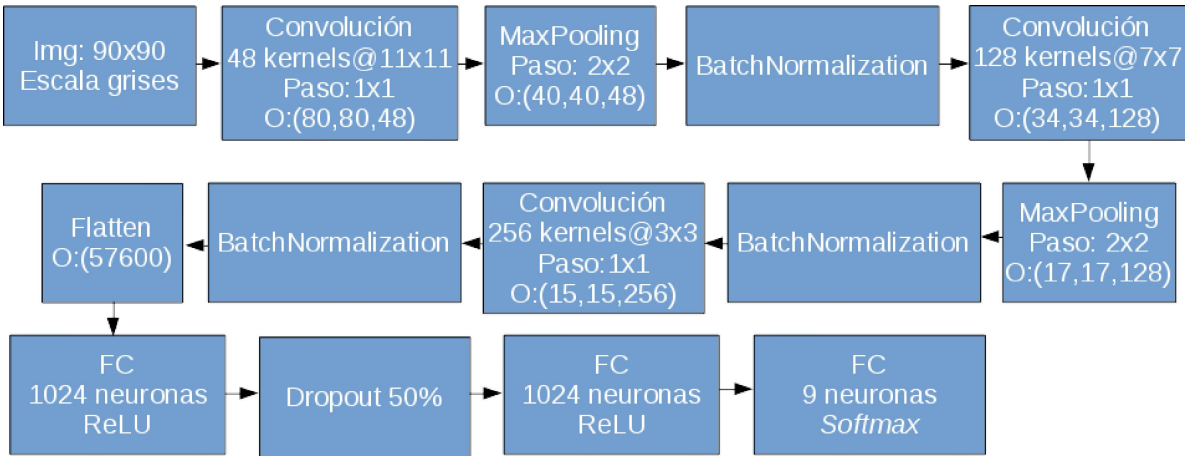


Figura 3.10: Diagrama de las capas que conforman la CNN4.

Todas las redes se optimizan con gradiente descendente estocástico (SGD, por sus siglas en inglés *Stochastic gradient descent*), cuentan una tasa de aprendizaje con valor 0.01 y un *momentum* de 0.9. La entrada a la red posee un tamaño de $90 \times 90 \times 1$ pixeles, en el caso del MLP se utiliza un vector de características con el valor de cada pixel.

4

Experimentación y resultados

En este capítulo se hablará de los experimentos que se llevaron a cabo durante el proyecto de investigación; asímismo se presentarán los resultados que de éstos emanan. Se utilizaron tres conjuntos de datos distintos y tres técnicas empleadas para el desarrollo de los experimentos.

4.1. Conjuntos de datos

Dentro de los conjuntos de datos usados para los experimentos, se cuenta con el MNIST, el cuál es un conjunto ampliamente utilizado para entrenar múltiples clasificadores. Se emplea como punto de partida debido a que tiene el preprocesamiento necesario y se presta para aquellas personas que quieran aprender métodos de aprendizaje máquina y reconocimiento de patrones. Otro conjunto se llama OIHACDB-40, éste conjunto contiene trazos arábigos que a simple vista parecen ser similares a los del BGT. Finalmente, tenemos al conjunto BGT, donde se ralizarán los experimentos de mayor interés.

4.2. Selección de arquitectura CNN

Los tipos de experimentos que se llevaron a cabo son tres y se presentan como un diagrama a bloques en la figura 4.1. Donde las lineas punteadas indican un entrenamiento en los

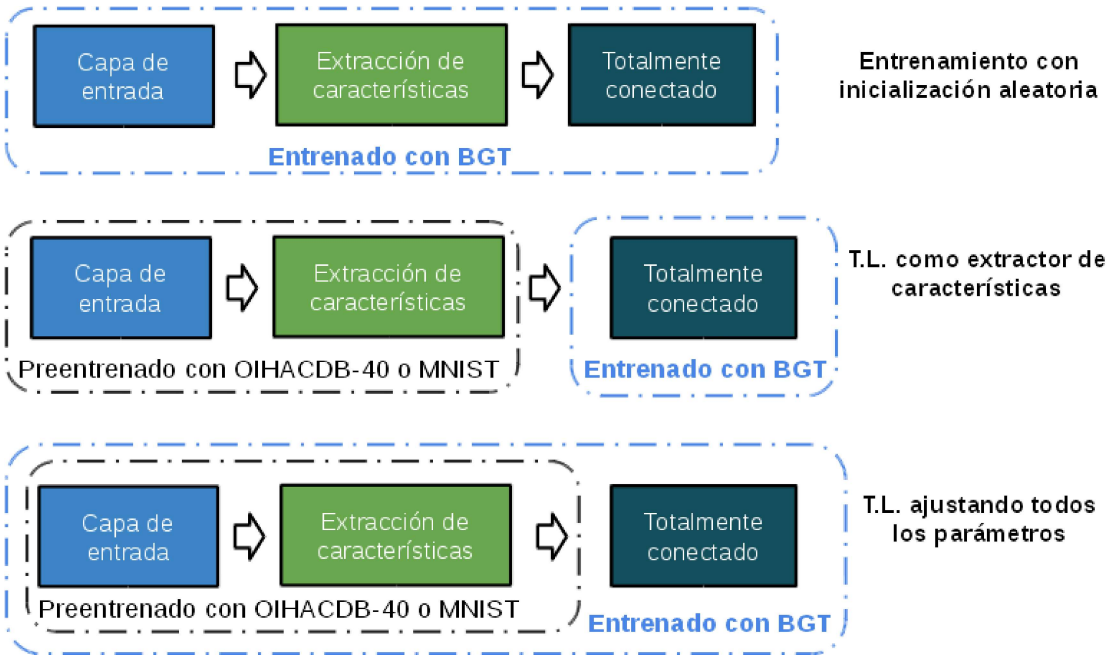


Figura 4.1: Diagrama a bloques de las principales tres estrategias utilizadas en los experimentos.

pesos, las lineas punteadas negras señalan que esos pesos fueron entrenados con los conjuntos MNIST y OIHACDB-40, mientras que las lineas azules denotan un entrenamiento de los pesos utilizando el conjunto BGT. Se entrenaron las cinco arquitecturas inicializando aleatoriamente los pesos, escogiendo para los experimentos posteriores, la que muestre un balance entre tiempo de entrenamiento y desempeño.

Una vez seleccionada la red, se entrenó dicha arquitectura con los otros conjuntos de datos y se hizo una transferencia de aprendizaje para comparar los resultados de los tres enfoques de entrenamiento: pesos aleatorios, extractor de características y ajuste total de los pesos.

4.3. Pruebas con los conjuntos de datos BGT, MNIST y OIHACDB-40

Al finalizar la etapa de selección de arquitectura, se entrenaron las redes con los datos del BGT. Para esto, fue necesario definir multiples parámetros. Entre ellos, se fijó el tamaño de

lote (*batch size*) en 128, el tamaño de las imágenes de $90\times90\times1$ pixeles en escala de grises debido a que fue el tamaño más gránde soportado por la RAM de la tarjeta de video sin que se interrumpiera el entrenamiento. En el caso del MLP se utilizó un vector de características con el valor de cada pixel. Se asignó un valor de 40 para el número de épocas por cada experimento, esto se debe a que se ha encontrado que después de 20 épocas se observa si el método se está entrenando o no, tomando como medida la exactitud. Permitiendo que el método siga entrenando durante un máximo de 40 épocas para favorecer el desempeño.

Además, se utilizó la técnica de validación cruzada (*cross validation*) con cinco iteraciones, dividiendo un 80 % de los datos para entrenar y un 20 % para probar con datos originales. De manera que la redes sean entrenadas y probadas con los mismos datos, se crearon listas con los nombres de las imágenes para el entrenamiento y prueba de cada una de las cinco iteraciones de la validación cruzada.

Definidos los parámetros, ulteriormente se cargan las imágenes de entrenamiento y prueba, cada una agregada a su lista respectiva. Éstas listas constan de 4 dimensiones (# imágenes, # canales de las imágenes, # renglones, # columnas) mientras que en otras dos listas se almacenan las etiquetas de cada imagen para cada etapa (entrenamiento y prueba). Posteriormente se normalizan los valores de cada pixel. Teniendo las imágenes listas, se crea la arquitectura que se va a utilizar para comenzar el entrenamiento de la red.

Concluido el entrenamiento se toman los datos de prueba y se le pide a la red que los clasifique, es aquí donde se compara la lista de los resultados obtenidos con la lista de los datos reales. Terminados los experimentos, se contrapondrán los promedios de dos métricas (exactitud y medida F) de cada arquitectura. Conservando los pesos con los que se haya obtenido un mejor desempeño puntual.

Para obtener resultados con los conjuntos MNIST y OIHACDB-40 se utilizaron los mismos parámetros de la red escogida con el conjunto BGT. Esta red se entrenó con los conjuntos

anteriormente mencionados. El conjunto MNIST fue probado con las 10,000 imágenes que se tienen originalmente destinadas para ello, mientras que para el conjunto OIHACDB-40 se utilizó una validación cruzada con 10 iteraciones debido a que no cuenta con datos exclusivamente para probar.

Observandose resultados similares a los señalados en la literatura, se guardaron los pesos para su posterior transferencia de aprendizaje.

4.3.1. Pesos aleatorios

Para los experimentos en los que se inicializa aleatoriamente los pesos se comenzó con el conjunto BGT, seguido por el MNIST y OIHACDB-40. De esta manera se seleccionó una arquitectura para la transferencia de aprendizaje.

Conjunto BGT

Para el primer experimento, se entrenaron las cinco arquitecturas con el conjunto BGT, inicializando los pesos de manera aleatoria. La Tabla 4.1 muestra los resultados de las cinco redes entrenadas con el conjunto BGT; donde la primer columna refiere a la arquitectura, la segunda columna muestra el tamaño de imagen utilizado, en la tercer columna se encuentra el resultado de la métrica exactitud y en la cuarta columna aparece el valor de la medida-F. Las últimas dos columnas es el resultado promedio de las cinco iteraciones en cada métrica.

Las redes convolucionales CNN2, CNN3 y CNN4 obtuvieron mejores resultados que el MLP, este comportamiento era esperado. La red CNN2 y CNN3 obtuvieron resultados similares, sin embargo, la CNN3 consta de una mayor cantidad de parámetros por lo que le toma más tiempo el ser entrenada, los tiempos que le toma a cada arquitectura entrenarse por una época se encuentra en la Tabla 4.2.

Además, en la Tabla 4.3 se agrega como calcular los parámetros por capa, en la primer

Tabla 4.1: Resultados comparativos de las cuatro arquitecturas con pesos aleatorios en el conjunto BGT

CNN Arq.	Tamaño de imagen	Exactitud	Medida-F
MLP	90×90	81.81 %	81.76 %
CNN1	90×90	78.69 %	78.72 %
CNN2	90×90	89.11 %	89.10 %
CNN3	90×90	89.81 %	89.80 %
CNN4	90×90	91.57 %	91.56 %

Tabla 4.2: Parámetros y tiempos realizados para entrenar cada arquitectura durante una época.

CNN Arq.	# Parámetros	Tiempo aproximado por época
MLP	4,677,641	1s
CNN1	333,429	4s
CNN2	3,644,429	4s
CNN3	171,543,625	46s
CNN4	60,646,185	21s

columna se indica el nombre de la capa mientras que en la segunda columna se presenta que hacer con las variables necesarias.

Tabla 4.3: Ecuaciones para calcular los parámetros de cada capa.

Capa	Ecuación
FC	$\mathrm{I}\times\mathrm{O} + \mathrm{bias}$
Convolución	$\mathrm{fm}_{-1} \times \mathrm{fm}_{1} \times \mathrm{m} \times \mathrm{n} + \mathrm{bias}$
Batch Normalization	$4 \times \mathrm{fm}_{-1}$

Donde I es el tamaño de la entrada (*Inputs*), O corresponde al tamaño de la salida (*Outputs*), fm_{-1} indica el número de mapas de características (*Feature maps*) de la capa anterior, fm_{1} es el número de mapas de características de la capa analizada, por último m y n concier-

nen al tamaño del filtro (base y altura en pixeles).

Dados los resultados de este experimento, se optó por utilizar la CNN4 para los experimentos posteriores. Aunque dure aproximadamente cinco veces más en el entrenamiento que la CNN2, puntualmente la supera por un 2.46 % en el promedio y se encuentra por encima del 90 % en todas las iteraciones.

En la Tabla 4.4 se presenta la matriz de confusión generada con los resultados obtenidos con la red CNN4 en una de las cinco iteraciones. En dicha Tabla se muestra que el desempeño de la red fue bastante bueno con excepción del trazo **C-3** y **C-5**, donde la red se confundió principalmente con los trazos **C-2** y **C-5**, debido a que ambos tienen círculos distribuidos en líneas. Por otro lado, el trazo **C-5** se confundió con **C-1** y **C-6**, se piensa que éste comportamiento se debe a que hay ocasiones en que los niños no hacen los círculos del trazo **C-5**, sino que trazan una línea, lo mismo ocurre con el trazo **C-1**, y por esta razón se cree que se confunde la red.

Tabla 4.4: Matriz de confusión con datos de prueba BGT. Red CNN4 entrenada desde cero.

	C-A	**C-1**	**C-2**	**C-3**	**C-4**	**C-5**	**C-6**	**C-7**	**C-8**	**Acc. (%)**
C-A	59	0	0	0	0	0	0	2	0	96.72
C-1	0	54	0	0	0	1	0	0	0	98.18
C-2	0	1	56	1	0	0	1	0	0	94.91
C-3	0	1	3	60	0	5	0	0	1	85.71
C-4	1	0	0	0	56	0	2	2	0	91.80
C-5	0	2	0	0	1	57	4	0	1	87.69
C-6	0	0	2	0	1	1	62	0	0	93.93
C-7	4	0	0	0	0	0	0	50	2	89.28
C-8	0	0	0	0	0	0	0	3	74	96.10
									Promedio	92.70

OIHACDB-40 y MNIST

Debido a que en los experimentos subsiguientes se va a ejecutar una transferencia de aprendizaje, se utilizaron como base los pesos de la red CNN4 que se entrenó con el conjunto OIHACDB-40 y MNIST. Los pesos se inicializaron aleatoriamente y se tomaron los

parámetros mencionados en la Sección 4.3.

En la Tabla 4.5 se presentan los resultados obtenidos por la red CNN4 y otras redes en la literatura, donde se puede observar que la segunda columna indica el conjunto con el que fue entrenado la red, en la tercer columna se presenta el tamaño de imagen definido, mientras que en la cuarta columna se expone el resultado porcentual con la métrica exactitud.

Tabla 4.5: Desempeño de las red CNN4 y comparación con la literatura utilizando los conjuntos OIHACDB-40 y MNIST.

CNN	Conjunto de datos	Tamaño de imagen	Exactitud
CNN4	MNIST	90×90	99.3677 %
LeNet-5 [23]	MNIST	32×32	99.05 %
CNN4	OIHACDB-40	90×90	98.3332 %
ALexNet [35]	OIHACDB-40	227×227	100 %

El resultado en ambos conjuntos es bueno debido a que es similar a los registrados en la literatura.

4.3.2. Transferencia de aprendizaje

En esta etapa se presentan dos tipos de experimentos, ambos parten de la transferencia de aprendizaje de una red que fue previamente entrenada con un conjunto de datos a otra red que va a ser entrenada con un conjunto de datos distinto; esto se logra modificando los pesos obtenidos con anterioridad.

Los dos experimentos refieren a entrenar la red CNN4 con enfoques distintos, uno donde se entrena toda la red utilizando una taza de aprendizaje pequeña y otro donde se utilizan las capas de convolución para extraer características y únicamente se entrena el clasificador que de la red.

Extractor de características

Para este experimento se toman los pesos de la red CNN4 entrenada con los conjuntos MNIST y OIHACDB-40, se cargan los pesos y se remueve el parámetro de entrenamiento de las capas de convolución. Una vez hecho esto, se entrena la red con el conjunto BGT, modificando los valores de los pesos en el clasificador de la red.

En las Tablas 4.6 y 4.7 se presentan las matrices de confusión generadas en una de las cinco pruebas realizadas para la transferencia de aprendizaje con el conjunto BGT.

Cuando se entrenó sobre el conjunto OIHACDB-40, se observó que las clases que más se confunden son **C-2** y **C-3**, se cree que la razon es que para **C-2** se dibujan líneas de círculos y en **C-3** más líneas de círculos, con diferencia en la distribución de las líneas, ya que en **C-3** las líneas comienzan de un punto en particular.

Tabla 4.6: Matriz de confusión con datos de prueba BGT. Transferencia de aprendizaje como extractor de características con OIHACDB-40.

	C-A	**C-1**	**C-2**	**C-3**	**C-4**	**C-5**	**C-6**	**C-7**	**C-8**	**Acc. (%)**
C-A	53	1	0	0	1	0	0	2	1	91.37
C-1	1	58	2	2	0	0	0	0	0	92.06
C-2	0	4	58	3	0	0	0	0	1	87.87
C-3	0	1	4	54	0	0	2	0	1	87.09
C-4	4	1	0	0	58	0	1	0	2	87.87
C-5	0	4	0	3	1	58	1	0	0	86.56
C-6	0	0	1	0	3	2	55	3	0	85.93
C-7	3	0	0	0	0	0	0	54	1	93.10
C-8	2	0	0	0	1	0	0	1	62	93.93
									Promedio	89.53

Por otro lado, cuando se entrenó sobre el conjunto MNIST, a pesar de obtener más errores en **C-3** y **C-7**, la clase **C-6** se confundió con **C-1**, **C-4**, **C-5**, **C-7** Y **C-8**. Mientras que la clase **C-3** se confundió mucho con **C-2** y muy poco con **C-4**. Es importante remarcar que la red encuentra un poco de similitud entre **C-7** y **C-A**, así como con **C-7** y **C-8** en ambos casos, aunque con distinta precisión.

Tabla 4.7: Matriz de confusión con datos de prueba BGT. Transferencia de aprendizaje como extractor de características con MNIST.

	C-A	**C-1**	**C-2**	**C-3**	**C-4**	**C-5**	**C-6**	**C-7**	**C-8**	**Acc. (%)**
C-A	54	1	0	0	2	0	0	1	0	93.10
C-1	0	54	5	1	1	0	2	0	0	85.71
C-2	0	2	63	1	0	0	0	0	0	95.45
C-3	0	0	8	53	1	0	0	0	0	85.48
C-4	2	0	0	0	57	0	4	0	3	86.36
C-5	0	4	0	3	1	59	0	0	0	88.05
C-6	0	1	0	0	1	3	57	1	1	89.06
C-7	4	0	1	0	0	0	0	49	4	84.48
C-8	0	0	0	0	1	1	0	1	63	95.45
									Promedio	89.24

Ajuste mínimo

Para éstos experimentos, se cargan los mismos pesos que se cargaron en el experimento anterior, a diferencia de éste, todos los pesos se ajustan con una tasa de aprendizaje baja con valor 0.0001. Se considera que si dos conjuntos de datos tienen elementos en común, entonces los pesos deberían ajustarse un poco al problema, sin olvidar lo aprendido del otro conjunto de datos.

En éste último experimento, se obtuvo una precisión ligeramente mayor en comparación con las matrices de confusión contenidas en las Tablas 4.6 y 4.7. De igual manera, sigue existiendo cierta confusión entre los trazos **C-1** y **C-2**, los trazos **C-7** con **C-A** y **C-8** y mucha confusión al clasificar el trazo **C-5** y confundirlo con **C-4** y **C-6** por igual en la Tabla 4.8 cuando se entrena con OIHACDB-40.

En estos últimos resultados, se alcanza a apreciar la confusión de la red CNN4 entre **C-7** y **C-4**, **C-8** y **C-A**, se cree que se debe a que en **C-7** se deben dibujar líneas rectas y en los trazos **C-A**, **C-4** y **C-8** tambien existen líneas rectas que se intersectan, ésto provoca que existan ángulos y ocasione una clasificación errónea.

En la Figura 4.2 se muestra un ejemplo de una imagen de clase **C-A** y la red la clasificó como **C-7**, por otro lado, en la Figura 4.3 se tiene una imagen de clase **C-7** que a simple vista

Tabla 4.8: Matriz de confusión con datos de prueba BGT. Transferencia de aprendizaje para ajuste de pesos con OIHACDB-40.

	C-A	**C-1**	**C-2**	**C-3**	**C-4**	**C-5**	**C-6**	**C-7**	**C-8**	**Acc. (%)**
C-A	55	1	0	0	1	0	0	1	0	94.82
C-1	0	56	4	1	0	0	1	1	0	88.88
C-2	0	3	59	3	0	0	0	1	0	89.39
C-3	0	0	4	57	1	0	0	0	0	91.93
C-4	1	1	1	0	58	0	1	2	2	87.87
C-5	0	1	0	2	4	55	4	1	0	82.08
C-6	0	0	0	0	2	2	57	3	0	89.06
C-7	2	0	0	0	1	0	0	53	2	91.37
C-8	0	0	0	0	0	0	0	6	60	90.90
									Promedio	89.59

Tabla 4.9: Matriz de confusión con datos de prueba BGT. Transferencia de aprendizaje para ajuste de pesos con MNIST.

	C-A	**C-1**	**C-2**	**C-3**	**C-4**	**C-5**	**C-6**	**C-7**	**C-8**	**Acc. (%)**
C-A	54	1	0	0	0	0	0	2	1	93.10
C-1	0	59	1	2	1	0	0	0	0	93.65
C-2	0	2	60	2	0	0	0	2	0	90.90
C-3	0	0	3	55	1	1	2	0	0	88.70
C-4	1	0	0	0	58	0	0	1	6	87.87
C-5	0	1	0	2	0	60	3	0	1	89.55
C-6	0	0	0	0	1	4	58	1	0	90.6
C-7	5	0	0	0	1	0	0	48	4	82.75
C-8	0	0	0	0	0	0	0	6	60	90.90
									Promedio	89.78

es similar a la mostrada en 4.2 por lo que incluso un humano pudiera confundirse debido a la cantidad de picos del trazo encontrado en el lado derecho de cada Figura y el trazo redondo que se encuentra en el lado izquierdo.

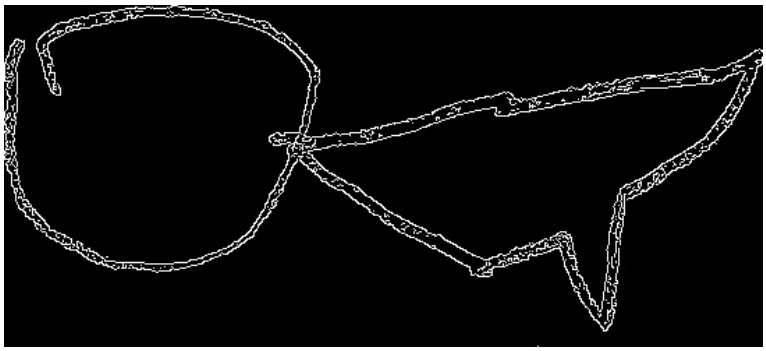


Figura 4.2: Trazo de clase **C-A** confundido con clase **C-7**.

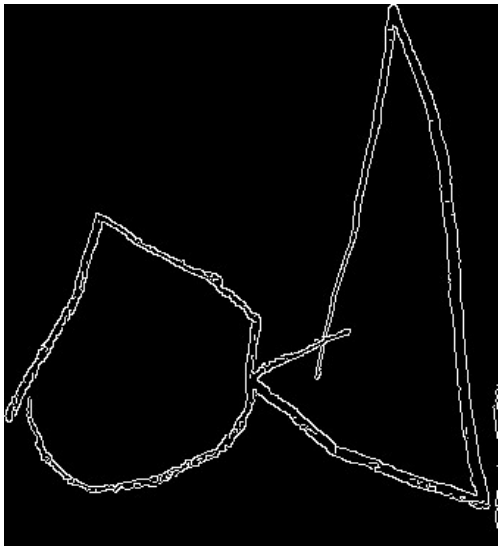


Figura 4.3: Trazo de clase **C-7**.

En la Tabla 4.10 se presenta un resumen de los resultados obtenidos con los enfoques de la transferencia de aprendizaje. En la primer columna se especifica que se utilizó la red CNN4, en la segunda columna se menciona el conjunto de datos con el que fue previamente entrenada la red, seguido del conjunto de datos con el quen se reentrenó. En la columna tercera se muestra el tamaño de imagen que se utilizó para cada prueba, la cuarta columna manifiesta el enfoque utilizado para inicializar los pesos, mientras que en la quinta columna demuestra la exactitud promedio obtenida de cada prueba realizada.

Tabla 4.10: Resultados obtenidos con la estrategia de Transferencia de aprendizaje

Arq. CNN	Conjunto	Tamaño imagen	Inicialización de pesos.	Exactitud
CNN4	OIHACDB-40 y BGT	90×90	TL-FE	89.614 %
CNN4	MNIST y BGT	90×90	TL-FE	88.982 %
CNN4	OIHACDB-40 y BGT	90×90	TL-FT	89.789 %
CNN4	MNIST y BGT	90×90	TL-FT	90.701 %

Analizando la tabla 4.10 se puede notar que fue beneficioso para el reconocimiento de los trazos, partir de los pesos obtenidos con el conjunto OIHACDB-40 y que se utilizaran como

extractor de características. A pesar de haber obtenido un buen resultado, no supera al que se obtuvo cuando se inicializaron los pesos de manera aleatoria.

4.3.3. Pruebas estadísticas

Para la Prueba de Friedman se acomodó la Tabla 4.11 conforme a los rangos obtenidos en los resultados y se agregó la suma de los rangos de cada método (R_j).

Tabla 4.11: Resultados acomodados jerárquicamente conforme al resultado obtenido por iteración

	Rangos								
Iteración	MLP	CNN1	CNN2	CNN3	CNN4	CNN4 MNIST-FC	CNN4 MNIST-FT	CNN4 OIHACDB-FC	CNN4 OIHACDB-FT
1	8	9	7	2	1	6	3	4.5	4.5
2	8	9	6	4	1	3	2	7	5
3	8	9	6	5	1	7	2	3.5	3.5
4	9	8	6	3	2	7	1	4	5
5	8	9	6	7	1	5	2	4	3
Suma	41	44	31	21	6	28	10	23	21
Promedio	8.2	8.8	6.2	4.2	1.2	5.6	2	4.6	4.2

Con la Tabla ordenada, se realizaron los cálculos necesarios, tal y como se muestra a continuación.

$$F_r = \frac{12\sum_{j=1}^{k} R_j^2 - 3N^2k(k+1)^2}{Nk(k+1) + \frac{\left(Nk - \sum_{i=1}^{N}\sum_{j=1}^{g_i} t_{i,j}^3\right)}{(k-1)}}$$

$$\sum_{i=1}^{N}\sum_{j=1}^{g_i} t_{i,j}^3 = 57$$

$$F_r = \frac{12(41^2 + 44^2 + 31^2 + 21^2 + 6^2 + 28^2 + 10^2 + 23^2 + 21^2) - 3(5)^2(9)(9+1)^2}{5(9)(9+1) + \frac{5(9) - 57}{9-1}}$$

$$= \frac{82,908 - 67,500}{450 + \frac{-12}{8}}$$

$$= 34.3545$$

El valor que se encuentra en la tabla de χ^2 con $k-1$ grados de libertad y un α de 0.05 es de 15.5073, el valor obtenido de F_r es 34.3545; se rechaza la hipótesis nula ya que el valor F_r es mayor al de χ^2 y la prueba indica que el rango de los modelos no son iguales, por lo tanto los modelos tampoco lo son, sin embargo, es necesario conocer cuáles modelos no son iguales y para ésto se realiza la Prueba de Nemenyi.

El cálculo de la distancia crítica para la Prueba de Nemenyi se presenta a continuación.

$$\begin{aligned} CD &= q_\alpha \sqrt{\frac{k(k+1)}{6N}} \\ &= 3.102 \sqrt{\frac{9(9+1)}{6(5)}} \\ &= 3.102\sqrt{3} \\ &= 5.3728 \end{aligned}$$

Partiendo de que la Prueba de Friedman reveló que existe diferencia entre los modelos utilizados, es primordial conocer cuáles modelos son estadísticamente iguales y cuáles no. Por lo que se realizó la Prueba de Nemenyi, la cuál arrojó un valor de la distancia crítica de 5.3728; ésto indica que si la diferencia de los promedios que se encuentran en la Tabla 4.11 es menor a la distancia crítica, no existe diferencia estadística entre esos modelos en específico.

En la Figura 4.4 se presentan los resultados obtenidos de la Prueba de Nemenyi, donde se acomodan los modelos sobre un eje numerado según el promedio jerárquico obtenido y la línea roja tiene la longitud de la distancia crítica, uniendo los modelos estadísticamente iguales. En general, los modelos que utilizan CNNs entregaron resultados estadísticamente iguales, por lo que no importa cuál modelo se utilice; a pesar de ello, el BGT es una prueba que se utiliza para detectar posibles problemas visomotores y es necesario utilizar el modelo con el mejor desempeño, éste es el que se encuentra más a la derecha y es la red CNN4

únicamente entrenada con el conjunto BGT.

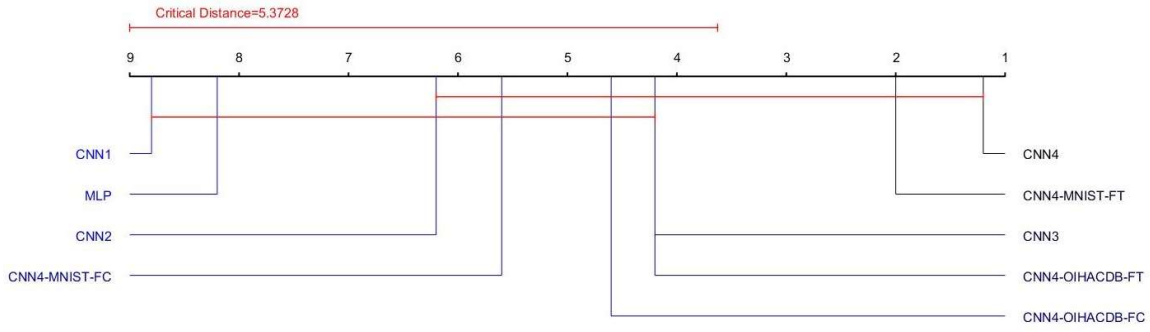


Figura 4.4: Resultados de la Prueba de Nemenyi para la diferencia estadística.

5

Conclusiones y trabajo futuro

Este trabajo de investigación ha sido provechoso, dado que a la fecha, no se encontró un trabajo similar donde se haya intentado detectar de manera automática y clasificar los trazos de un test cognitivo utilizando aprendizaje máquina.

Se consiguió crear un conjunto de datos del test BGT, con una cantidad considerable para probar distintos métodos de clasificación. El conjunto tiene un preprocesamiento, por lo que hace más sencilla la tarea de entrenar algún clasificador. Se piensa que las distorsiones elásticas ayudaron al entrenamiento de la red, aunque sean imágenes artificiales creadas de datos originales, se asemejan mucho a trazos reales realizados por otras personas.

Es posible que con los enfoques de transferencia de aprendizaje utilizados, se haya logrado generalizar la clasificación, sin embargo, la manera de comprobarlo eficazmente es adquiriendo más datos.

Se considera que se ha asentado la base para crear un sistema completo que pueda identificar y calificar los trazos del BGT, ayudando a que las personas tengan un mejor acceso a este tipo de pruebas, ya que son de suma importancia para detectar a tiempo problemas visomotores.

5.1. Trabajo futuro

Dentro del trabajo futuro existen múltiples opciones, sin embargo, las principales son:

- Adquirir una mayor cantidad de datos, tanto como para probar, como para aumentar la generalización del aprendizaje.
- Continuar con la investigación, crear un sistema que califique los trazos, debido a que cada trazo se debe calificar de una manera distinta, partir de la clasificación automática para una calificación automática.
- Probar con distintos clasificadores como SVM, KNN, árboles de desición, entre otros. Así como utilizar otro tipo de configuraciones de redes profundas como autoencoders.

Referencias

[1] P. Bonifacci, "Children with low motor ability have lower visual-motor integration ability but unaffected perceptual skills," *Human movement science*, vol. 23, no. 2, pp. 157–168, 2004.

[2] A. A. A. d. Santos and L. M. d. Jorge, "Bender test with dyslexics: comparison of two systems of punctuation," *Psico-USF*, vol. 12, no. 1, pp. 13–21, 2007.

[3] B. Böhm, A. Lundequist, and A.-C. Smelder, "Visual-motor and executive functions in children born preterm: The bender visual motor gestalt test revisited," *Scandinavian Journal of Psychology*, vol. 51, no. 5, pp. 376–384, 2010.

[4] E. M. Koppitz, "The bender gestalt test and learning disturbances in young children," *Journal of Clinical Psychology*, vol. 14, no. 3, pp. 292–295, 1958.

[5] M. Moetesum, I. Siddiqi, U. Masroor, and C. Djeddi, "Automated scoring of bender gestalt test using image analysis techniques," in *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*, pp. 666–670, IEEE, 2015.

[6] M. Leo, G. Medioni, M. Trivedi, T. Kanade, and G. Farinella, "Computer vision for assistive technologies," *Computer Vision and Image Understanding*, vol. 154, no. Supplement C, pp. 1 – 15, 2017.

[7] A. Garcia-Martinez, J. M. V. Samper, and J. M. Sabater-Navarro, "Automatic detection of surgical haemorrhage using computer vision," *Artificial Intelligence in Medicine*, 2017.

[8] H. Zhang and D. Li, "Applications of computer vision techniques to cotton foreign matter inspection: A review," *Computers and Electronics in Agriculture*, vol. 109, no. Supplement C, pp. 59 – 70, 2014.

[9] C. G. García, D. Meana-Llorián, B. C. P. G-Bustelo, J. M. C. Lovelle, and N. Garcia-Fernandez, "Midgar: Detection of people through computer vision in the internet of things scenarios to improve the security in smart cities, smart towns, and smart homes," *Future Generation Computer Systems*, vol. 76, no. Supplement C, pp. 301 – 313, 2017.

[10] T. Stoeger, N. Battich, M. D. Herrmann, Y. Yakimovich, and L. Pelkmans, "Computer vision for image-based transcriptomics," *Methods*, vol. 85, no. Supplement C, pp. 44 – 53, 2015. Inferring Gene Regulatory Interactions from Quantitative High-Throughput Measurements.

[11] B. Sudarsan, W. Ji, A. Biswas, and V. Adamchuk, "Microscope-based computer vision to characterize soil texture and soil organic matter," *Biosystems Engineering*, vol. 152, pp. 41–50, 2016.

[12] A. Al-Kaff, D. Martín, F. García, A. de la Escalera, and J. M. Armingol, "Survey of computer vision algorithms and applications for unmanned aerial vehicles," *Expert Systems with Applications*, 2017.

[13] R. Dandoš, K. Mozdřeň, and H. Staňková, "A new control mark for photogrammetry and its localization from single image using computer vision," *Computer Standards & Interfaces*, 2017.

[14] M. Heimberger, J. Horgan, C. Hughes, J. McDonald, and S. Yogamani, "Computer vision in automated parking systems: Design,implementation and challenges," *Image and Vision Computing*, 2017.

[15] M. Kosinski and Y. Wang, "Deep neural networks are more accurate than humans at detecting sexual orientation from facial images.," Sep 2017.

[16] A. Cardoso and A. Wichert, "Handwritten digit recognition using biologically inspired features," *Neurocomputing*, vol. 99, pp. 575–580, 2013.

[17] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *The Journal of physiology*, vol. 160, no. 1, pp. 106–154, 1962.

[18] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[19] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of human genetics*, vol. 7, no. 2, pp. 179–188, 1936.

[20] A. Mucherino, P. J. Papajorgji, and P. M. Pardalos, *k-Nearest Neighbor Classification*, pp. 83–106. New York, NY: Springer New York, 2009.

[21] M. H. Hassoun, *Fundamentals of artificial neural networks*. MIT press, 1995.

[22] D. Barber, *Bayesian reasoning and machine learning*. Cambridge University Press, 2012.

[23] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[24] Y. LeCun, "The mnist database of handwritten digits," *http://yann. lecun. com/exdb/mnist/*, 1998.

[25] M. W. Gardner and S. Dorling, "Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences," *Atmospheric environment*, vol. 32, no. 14, pp. 2627–2636, 1998.

[26] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1097–1105, Curran Associates, Inc., 2012.

[27] H. Zhang, A. C. Berg, M. Maire, and J. Malik, "Svm-knn: Discriminative nearest neighbor classification for visual category recognition," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 2, pp. 2126–2136, IEEE, 2006.

[28] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[29] J. C. Platt, N. Cristianini, and J. Shawe-Taylor, "Large margin dags for multiclass classification," in *Advances in neural information processing systems*, pp. 547–553, 2000.

[30] J. Du and Q. Huo, "A discriminative linear regression approach to adaptation of multi-prototype based classifiers and its applications for chinese ocr," *Pattern Recognition*, vol. 46, no. 8, pp. 2313–2322, 2013.

[31] A. Boukharouba and A. Bennia, "Novel feature extraction technique for the recognition of handwritten digits," *Applied Computing and Informatics*, vol. 13, no. 1, pp. 19–26, 2017.

[32] M. Elleuch, R. Maalej, and M. Kherallah, "A new design based-svm of the cnn classifier architecture with dropout for offline arabic handwritten recognition," *Procedia Computer Science*, vol. 80, pp. 1712–1723, 2016.

[33] P. Y. Simard, D. Steinkraus, J. C. Platt, *et al.*, "Best practices for convolutional neural networks applied to visual document analysis.," in *ICDAR*, vol. 3, pp. 958–962, 2003.

[34] F. J. Huang, Y.-L. Boureau, Y. LeCun, *et al.*, "Unsupervised learning of invariant feature hierarchies with applications to object recognition," in *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pp. 1–8, IEEE, 2007.

[35] C. Boufenar, A. Kerboua, and M. Batouche, "Investigation on deep learning for off-line handwritten arabic character recognition," *Cognitive Systems Research*, 2017.

[36] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[37] D. C. Cireşan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," *CoRR*, vol. abs/1202.2745, 2012.

[38] K. Jarrett, K. Kavukcuoglu, Y. LeCun, *et al.*, "What is the best multi-stage architecture for object recognition?," in *Computer Vision, 2009 IEEE 12th International Conference on*, pp. 2146–2153, IEEE, 2009.

[39] J. Bai, Z. Chen, B. Feng, and B. Xu, "Image character recognition using deep convolutional neural network learned from different languages," in *2014 IEEE International Conference on Image Processing (ICIP)*, pp. 2560–2564, Oct 2014.

[40] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and transferring mid-level image representations using convolutional neural networks," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1717–1724, June 2014.

[41] A. Rehman and T. Saba, "Neural networks for document image preprocessing: state of the art," *Artificial Intelligence Review*, vol. 42, pp. 253–273, Aug 2014.

[42] S. T. Bow, *Pattern recognition and image preprocessing*. CRC press, 2002.

[43] I. Sobel and G. Feldman, "A 3x3 isotropic gradient operator for image processing," *a talk at the Stanford Artificial Project in*, pp. 271–272, 1968.

[44] J. M. Prewitt, "Object enhancement and extraction," *Picture processing and Psychopictorics*, vol. 10, no. 1, pp. 15–19, 1970.

[45] J. CANNY, "A computational approach to edge detection," in *Readings in Computer Vision*, pp. 184 – 203, San Francisco (CA): Morgan Kaufmann, 1987.

[46] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[47] S. Marsland, *Machine learning: an algorithmic perspective*. CRC press, 2015.

[48] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[49] N. Qian, "On the momentum term in gradient descent learning algorithms," *Neural networks*, vol. 12, no. 1, pp. 145–151, 1999.

[50] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *International conference on machine learning*, pp. 1139–1147, 2013.

[51] Y. Bengio, "Learning deep architectures for AI," *Foundations and Trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[52] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[53] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[54] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*, vol. 1. MIT press Cambridge, 2016.

[55] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," tech. rep., California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[56] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel, "Handwritten digit recognition with a back-propagation network," in *Advances in neural information processing systems*, pp. 396–404, 1990.

[57] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," *arXiv preprint arXiv:1603.07285*, 2016.

[58] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[59] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine learning research*, vol. 7, no. Jan, pp. 1–30, 2006.

[60] J. A. Linebach, B. P. Tesch, L. M. Kovacsiss, *et al.*, *Nonparametric statistics for applied research*. Springer, 2014.

[61] A. Lawgali, M. Angelova, and A. Bouridane, "Hacdb: Handwritten arabic characters database for automatic character recognition," in *Visual Information Processing (EUVIP), 2013 4th European Workshop on*, pp. 255–259, IEEE, 2013.

[62] F. Chollet *et al.*, "Keras." `https://github.com/fchollet/keras`, 2015.