Centro de Investigación y de Estudios Avanzados
del Instituto Politécnico Nacional

Unidad Zacatenco

Departamento de Computación

Análisis comparativo de algoritmos de clasificación de imágenes basados en aprendizaje automático tradicional y aprendizaje profundo

Tesis que presenta

Balam Garcia Morgado

para obtener el Grado de

Maestro en Ciencias en Computación

Director de la tesis:

Dr. Luis Gerardo de la Fraga

Ciudad de México **Septiembre del 2024**

Resumen

Este trabajo presenta un análisis comparativo entre algoritmos de aprendizaje automático tradicional y aprendizaje profundo para la clasificación de imágenes. Uno de los problemas frecuentes en la literatura es la omisión de parámetros y arquitecturas de los algoritmos utilizados, lo que dificulta la replicación de resultados. Por ello, este estudio se enfoca en documentar todos los parámetros utilizados durante la fase experimental.

Para la experimentación, se generó una base de datos compuesta por 480 imágenes de ocho figuras distintas, es decir, ocho clases. Se emplearon los métodos de extracción de características Momentos de Hu y Histograma de Gradientes Orientados (HOG). En particular, se experimentó con HOG para encontrar una fórmula que calcule el número de características que extraerá de cada imagen, basado en los parámetros del descriptor. Los algoritmos de aprendizaje automático tradicional seleccionados fueron: Bayes ingenuo, Máquina de Vectores de Soporte (MVS), redes neuronales directas y redes neuronales de estado eco.

Para los algoritmos de aprendizaje profundo, se eligieron arquitecturas de redes neuronales convolucionales preentrenadas disponibles en el sitio de Pytorch, seleccionando aquellas con menos de 10 millones de parámetros. Posteriormente, se modificaron estas arquitecturas utilizando únicamente las primeras capas convolucionales y transfiriendo los pesos y sesgos de la red original a la nueva, aplicando así la técnica de transferencia de aprendizaje.

Estas redes neuronales convolucionales modificadas se utilizaron para clasificar la base de datos original (BDO). Debido al limitado número de imágenes, los modelos mostraron un evidente sobreajuste, lo que hizo necesario aplicar la técnica de aumento de datos para incrementar el tamaño del conjunto de entrenamiento y evaluación. Los modelos que lograron clasificar correctamente todas las imágenes de la base de datos se emplearon para la clasificación del conjunto de datos MNIST, que contiene manuscritos de números del 0 al 9, y se documento exactitud y tiempo de entrenamiento de los modelos.

Finalmente, tomando como referencia el artículo Automatic Classification of Melanoma Skin Cancer with Deep Convolutional Neural Networks de Aljohani et al (2022), se utilizaron cuatro de nuestras redes neuronales modificadas para desarrollar un clasificador binario de imágenes de melanomas y lesiones benignas de queratosis, utilizando la base de datos ISIC 2019 y se reporta la exactitud y tiempo de entrenamiento.

Abstract

This work presents a comparative analysis between traditional machine learning algorithms and deep learning for image classification. A common issue in the literature is the omission of parameters and architectures used in the algorithms, which makes it difficult to replicate results. Therefore, this study focuses on documenting all the parameters used during the experimental phase.

For the experimentation, a database composed of 480 images of eight different shapes, that is, eight classes, was generated. The feature extraction methods used were Hu Moments and Histogram of Oriented Gradients (HOG). Specifically, experiments were conducted with HOG to find a formula that calculates the number of features to be extracted from each image based on the descriptor's parameters. The selected traditional machine learning algorithms were: Naive Bayes, Support Vector Machine (SVM), feedforward neural networks, and echo state networks.

For deep learning algorithms, pre-trained convolutional neural network architectures available on the Pytorch site were chosen, selecting those with less than 10 million parameters. These architectures were then modified by using only the first convolutional layers and transferring the weights and biases from the original network to the new one, thus applying the transfer learning technique.

These modified convolutional neural networks were used to classify the original database. Due to the limited number of images, the models showed evident overfitting, making it necessary to apply data augmentation techniques to increase the size of the training and evaluation set. The models that correctly classified all the images in the database were then used to classify the MNIST dataset, which contains handwritten digits from 0 to 9, and the accuracy and training time of the models were documented.

Finally, taking the article Automatic Classification of Melanoma Skin Cancer with Deep Convolutional Neural Networksby Aljohani et al. (2022) as a reference, four of our modified neural networks were used to develop a binary classifier for melanoma and benign keratosis lesion images, using the ISIC 2019 database, and the accuracy and training time were reported.

Agradecimientos

A CONAHCYT por el apoyo económico para estudiar la maestría al CINVESTAV por los conocimientos y espacios brindados para mi formación profesional.

A mis asesor, el Dr. Luis Gerardo de la Fraga por sus atenciones en la realización de la presente.

A mis sinodales, la Dra. Brisbane Ovilla Martínez y al Dr. Eduardo López Domínguez por sus atentas instrucciones para mejora el trabajo realizado.

A mis padres y hermano por ser siempre un pilar en mi formación profesional y personal.

Índice general

Resumen III

Abstract V

Índice de figuras XI

Índice de tablas XX

1. Introducción 3
- 1.1. Motivación del estudio . . . 3
- 1.2. Planteamiento del problema . . . 4
- 1.3. Objetivos de la investigación . . . 4
- 1.4. Estructura de la tesis . . . 5

2. Marco teórico 7
- 2.1. Aprendizaje automático tradicional . . . 7
  - 2.1.1. Bayes Ingenuo . . . 8
  - 2.1.2. Máquina de Vectores de Soporte . . . 8
  - 2.1.3. Redes neuronales . . . 10
  - 2.1.4. Redes neuronales de estado eco (RNEE) . . . 11
- 2.2. Aprendizaje profundo . . . 12
  - 2.2.1. Convolución . . . 13
  - 2.2.2. Agrupamiento . . . 14
  - 2.2.3. Capa completamente conectada . . . 16
- 2.3. Clasificación de imágenes . . . 17
- 2.4. Revisión de literatura . . . 17

3. Generación de base de datos de imágenes propia 21
- 3.1. Diseño . . . 21
- 3.2. Digitalización . . . 24
- 3.3. Aumento de datos . . . 26

4. Algoritmos de aprendizaje automático tradicional 29
- 4.1. Métodos de extracción de características . . . 29
  - 4.1.1. Momentos de Hu . . . 29

4.1.2. Histograma de Gradientes Orientados (HOG) 29
4.2. Bayes ingenuo 31
4.3. MVS 32
4.4. Red neuronal directa 32

5. Algoritmos de aprendizaje profundo **35**
5.1. Selección 35
5.2. Modificaciones de las arquitecturas 36
5.3. Comparativa de tamaño entre las arquitecturas del sitio de Pytorch y nuestras propuestas 37
5.4. Transferencia de aprendizaje 37

6. Desempeño de los modelos tradicionales **39**
6.1. Momentos de Hu 39
6.1.1. Bayes ingenuo 39
6.1.2. Máquina de Vectores de Soporte 43
6.1.3. Clasificador con redes neuronales directas 43
6.1.4. Clasificador con redes neuronales recurrentes de estado eco . . 45
6.2. HOG 49
6.2.1. Bayes ingenuo 49
6.2.2. Redes neuronales directas 50
6.2.3. Máquina de Vectores de Soporte 53
6.3. Aumento de datos 54
6.3.1. Bayes Ingenuo (Hu) 54
6.3.2. Máquina de Vectores de Soporte (Hu) 55
6.3.3. Redes neuronales (Hu) 56
6.3.4. Clasificador con redes neuronales recurrentes de estado eco (Hu) 56
6.3.5. Bayes Ingenuo (HOG) 57
6.3.6. Máquina de Vectores de Soporte (HOG) 58
6.3.7. Redes neuronales (HOG) 58
6.4. Sumario 59

7. Desempeño de los modelos de aprendizaje profundo **61**
7.1. Sobreajuste y aumento de datos 61
7.2. Experimentación con los optimizadores 64
7.3. Experimentación con 96,000 imágenes 66
7.4. DenseNet121 67
7.5. EfficientNet_B0 69
7.6. GoogLeNet 70
7.7. MNASNet0_5 71
7.8. MobileNet_V3_Large (V2) 72
7.9. RegNet_X_1_6GF 73
7.10. ShuffleNet_V2_X0_5 74

7.11. Mejores modelos de RNC para clasificar nuestra base de datos de imágenes . . . 75
7.12. MNIST . . . 75
7.12.1. DenseNet121 . . . 76
7.12.2. EfficientNet_B0 . . . 77
7.12.3. MobileNet_V3_Large . . . 78
7.13. International Skin Imaging Collaboration (ISIC) . . . 79
7.13.1. DenseNet121 . . . 80
7.13.2. EfficientNet_B0 . . . 81
7.13.3. GoogLeNet . . . 82
7.13.4. MobileNet_V3_Large . . . 83
7.14. Sumario . . . 84

8. Conclusiones **87**
8.1. Trabajo futuro . . . 91

Índice de figuras

2.1. Ejemplo de MVS. 9
2.2. Representación del Perceptrón. 10
2.3. Ejemplo del proceso de convolución con una imagen de entrada de 5×5 píxeles, un filtro de tamaño 3×3 y como resultado un mapa de características de tamaño 3×3. 13
2.4. Ejemplo del proceso de agrupación máxima (arriba) y el de agrupación promedio (abajo), ambas con un filtro de tamaño 2×2. 15
2.5. Ejemplo del proceso de agrupación promedio adaptativo 1×1. 16
2.6. Ejemplo de capa completamente conectada. 16

3.1. Formas de la base de datos. 22
3.2. Instancias de la figura óvalo. 22
3.3. Instancias de la figura rectángulo. 23
3.4. Instancias de la figura triángulo. 23
3.5. Cámara de la PlayStation 3 utilizada para generar la base de datos. . 23
3.6. Estadísticas del fondo original y un ejemplo de los generados aleatoriamente. 27

6.1. Matriz de confusión del modelo de Bayes Ingenuo utilizando como características los momentos invariantes de Hu. 40
6.2. Matrices de confusión del primer clasificador de Bayes Ingenuo con los momentos de Hu. Ambas matrices muestran valores distintos a cero únicamente en la diagonal principal, lo que indica una correcta clasificación de todas las imágenes. 41
6.3. Matrices de confusión del segundo clasificador de Bayes Ingenuo con las características c_1, c_2 y relación de áreas. Ambas matrices muestran valores distintos a cero únicamente en la diagonal principal, lo que indica una correcta clasificación de todas las imágenes. 42
6.4. Matrices de confusión de los modelos de Bayes Ingenuo con características de los momentos invariantes de Hu y las características c1,c2 y área. En la matriz para el conjunto de entrenamiento se observa un único error en la clasificación de una imagen de la clase línea, etiquetado como de la clase i. La matriz para el conjunto de prueba muestra una correcta clasificación para todas las imágenes. 42

6.5. Matrices de confusión del modelo de la MVS.SVC con momentos de Hu. En ambas matrices se observa la presencia de números diferentes a cero únicamente en la diagonal principal, lo que indica una correcta clasificación de todas las imágenes. 43
6.6. Arquitectura de la red neuronal. 44
6.7. Matrices de confusión de la red neuronal directa. En ambas matrices se observa la presencia de números distintos a cero únicamente en la diagonal principal, lo que señala la correcta clasificación de todas las imágenes. 44
6.8. Matriz de confusión de la red neuronal de estado eco con los momentos de Hu como características. Se observa en ambas matrices que las únicas clases las etiquetadas son óvalo y rectángulo. 45
6.9. Matriz de confusión de la primera red neuronal de estado eco con los momentos de Hu como características (dos clasificadores en total). La presencia de valores distintos de cero únicamente en la diagonal principal denota la correcta clasificación de todas las imágenes. 46
6.10. Matriz de confusión de la segunda red neuronal de estado eco con los momentos de Hu como características (dos clasificadores en total). Ambas matrices indican la clasificación incorrecta de algunas imágenes de las clases óvalo. 47
6.11. Matriz de confusión de la primera red neuronal de estado eco con los momentos de Hu como características (tres clasificadores en total). Las matrices indican la correcta clasificación de todas las imágenes. 47
6.12. Matriz de confusión de los tres clasificadores con redes neuronal de estado eco con los momentos de Hu como características. 48
6.13. Matrices de confusión de la red neuronal con 20 neuronas en su única capa oculta. Se observa en la matriz del conjunto de entrenamiento una correcta clasificación de todas las imágenes. En la matriz del conjunto de prueba se muestra la incorrectas clasificación de únicamente tres imágenes de la clase triángulo. 52
6.14. Matrices de confusión del modelo con un par de redes neuronales concatenadas. Ambas matrices muestran valores distintos a cero únicamente en su diagonal principal, representando la correcta clasificación de todas las imágenes. 52
6.15. Matrices de confusión del modelo de MVS con características HOG. Para el conjunto de entrenamiento todas las imágenes fueron clasificadas correctamente. La matriz de confusión del conjunto del prueba muestra la incorrecta clasificación de algunas imágenes de las clases óvalo y triángulo etiquetándolas como pertenecientes a la clase línea. 53
6.16. Matriz de confusión del modelo con Bayes Ingenuo para la BDA, se observa la incorrecta clasificación de imágenes de las clases pentágono etiquetándolas como rectángulo y de la clase triángulo como pentágono y rectángulo. 54

6.17. Matriz de confusión de los modelos de Bayes Ingenuo utilizando como características los momentos invariantes de Hu y las características c_1, c_2 y área. Se observa una clasificación errónea de imágenes de las clases estrella, pentágono y rectángulo. 55

6.18. Matriz de confusión del modelo con MVS utilizando como características los momentos invariantes de Hu de la nueva base de datos. . . . 55

6.19. Matriz de confusión del modelo con redes neuronales utilizando como características los momentos invariantes de Hu de la BDA. La presencia de valores distintos de cero únicamente en la diagonal principal representa la exactitud igual a 1 del modelo. 56

6.20. Matriz de confusión del modelo con tres redes neuronales de estado eco utilizando como características los momentos invariantes de Hu de la DBA. Se observa que siete imágenes de la clase estrella fueron etiquetados incorrectamente como pertenecientes a la clase rectángulo. 57

6.21. Matriz de confusión del modelo de Bayes Ingenuo utilizando las características HOG extraídas de la BDA con reducción a 64 dimensiones mediante ACP. La distribución de valores alejados de la diagonal principal indican la incapacidad del modelo de clasificar correctamente la mayoría de las imágenes. 57

6.22. Matriz de confusión del modelo de MVS utilizando las características HOG extraídas de la BDA con reducción a 64 dimensiones mediante ACP. Debido a la distribución de valores distintos a cero fuera de la diagonal principal se observa una incorrecta clasificación para la mayoría de las imágenes. 58

6.23. Matriz de confusión del modelo de dos redes neuronales utilizando las características HOG extraídas de la BDA con reducción a 64 dimensiones mediante ACP. Se observa una incorrecta clasificación para la mayoria de las imágenes. 59

7.1. Gráficas de exactitud y error durante el reentrenamiento de la SimpleNet con nuestra base de datos de imágenes. A la izq. la gráfica azul representa la exactitud para el conjunto de entrenamiento y la naranja para el conjunto de prueba, es evidente un sobreajuste del modelo. A la der. se muestra la grafica de error del modelo, en azul para el conjunto de entrenamiento y en naranja para el conjunto de prueba. 62

7.2. A la izq. las gráficas de exactitud de entrenamiento (azul) y prueba (naranja) y a la der. las gráficas de error de entrenamiento (azul) y prueba (naranja) durante el reentrenamiento de la SimpleNet con una capa oculta de 20 neuronas. La línea punteada vertical representa la obtención del mejor modelo para esa prueba. 63

7.3. A la izq. las gráficas de exactitud de entrenamiento (azul) y prueba (naranja) y a la der. las gráficas de error de entrenamiento (azul) y prueba (naranja) durante el reentrenamiento de la SimpleNet con una capa oculta de 32 neuronas. La línea punteada vertical representa la obtención del mejor modelo para esa prueba. 63

7.4. En la columna izq. las gráficas de exactitud de entrenamiento (azul) y prueba (naranja) y en la columna der. las gráficas de error de entrenamiento (azul) y prueba (naranja) durante el reentrenamiento de la SimpleNet con los optimizadores Adadelta, Adam y DGE, respectivamente. La línea punteada vertical representa la obtención del mejor modelo para esa prueba. 65

7.5. A la izq. las gráficas de exactitud de entrenamiento (azul) y prueba (naranja) y a la der. el error de entrenamiento (azul) y prueba (naranja) durante el reentrenamiento de nuestra modificación de la RNC SimpleNet y nuestra base de datos con 96,000 imágenes. La línea punteada vertical representa la obtención del mejor modelo para esa prueba, el cual muestra valores de exactitud sin sobreajuste. 66

7.6. Matrices de confusión de nuestra modificación de nuestra modificación de la RNC SimpleNet_5m_m2 para la clasificación de la base de datos de 96,000 imágenes. La matriz de la izq. muestra una correcta clasificación de todas las imágenes del conjunto de entrenamiento, en la matriz de la der. se observa que solo las imágenes del conjunto de prueba de las clases cruz, estrella, línea y pentágono fueron clasificadas correctamente por este modelo. 67

7.7. A la izq. las gráficas de exactitud de entrenamiento (azul) y prueba (naranja) y a la der. el error de entrenamiento (azul) y prueba (naranja) durante el reentrenamiento de nuestra modificación de la RNC DenseNet121 y nuestra base de datos con 96,000 imágenes. La línea punteada vertical representa la obtención del mejor modelo para esa prueba, el cual muestra valores de exactitud sin sobreajuste. 68

7.8. Matrices de confusión de nuestra modificación de nuestra modificación de la RNC DenseNet121 para la clasificación de la base de datos de 96,000 imágenes. Ambas matrices muestran una correcta clasificación de las imágenes por este modelo. 68

7.9. A la izq. las gráficas de exactitud de entrenamiento (azul) y prueba (naranja) y a la der. el error de entrenamiento (azul) y prueba (naranja) durante el reentrenamiento de nuestra modificación de la RNC EfficientNet_B0 y nuestra base de datos con 96,000 imágenes. La línea punteada vertical representa la obtención del mejor modelo para esa prueba, el cual muestra valores de exactitud sin sobreajuste. 69

7.10. Matrices de confusión de nuestra modificación de nuestra modificación de la RNC EfficientNet_B0 para la clasificación de la base de datos de 96,000 imágenes. Ambas matrices muestran una correcta clasificación de las imágenes por este modelo. 69

7.11. A la izq. las gráficas de exactitud de entrenamiento (azul) y prueba (naranja) y a la der. el error de entrenamiento (azul) y prueba (naranja) durante el reentrenamiento de nuestra modificación de la RNC GoogLeNet y nuestra base de datos con 96,000 imágenes. La línea punteada vertical representa la obtención del mejor modelo para esa prueba, el cual muestra valores de exactitud sin sobreajuste. 70

7.12. Matrices de confusión de nuestra modificación de nuestra modificación de la RNC GoogLeNet para la clasificación de la base de datos de 96,000 imágenes. La matriz de la izq. muestra una correcta clasificación de todas las imágenes del conjunto de entrenamiento, en la matriz de la der. se observa que solo las imágenes del conjunto de prueba de la clase rectángulo fueron mal clasificada por este modelo. 70

7.13. A la izq. las gráficas de exactitud de entrenamiento (azul) y prueba (naranja) y a la der. el error de entrenamiento (azul) y prueba (naranja) durante el reentrenamiento de nuestra modificación de la RNC MNASNet0_5 y nuestra base de datos con 96,000 imágenes. La línea punteada vertical representa la obtención del mejor modelo para esa prueba, el cual muestra valores de exactitud sin sobreajuste. 71

7.14. Matrices de confusión de nuestra modificación de nuestra modificación de la RNC MNASNet0_5 para la clasificación de la base de datos de 96,000 imágenes. La matriz de la izq. muestra una correcta clasificación de todas las imágenes del conjunto de entrenamiento, en la matriz de la der. se observa que solo las imágenes del conjunto de prueba de la clase óvalo fueron mal clasificada por este modelo. 71

7.15. A la izq. las gráficas de exactitud de entrenamiento (azul) y prueba (naranja) y a la der. el error de entrenamiento (azul) y prueba (naranja) durante el reentrenamiento de nuestra modificación de la RNC MobileNet_V3_Large (V2) y nuestra base de datos con 96,000 imágenes. La línea punteada vertical representa la obtención del mejor modelo para esa prueba, el cual muestra valores de exactitud sin sobreajuste. 72

7.16. Matrices de confusión de nuestra modificación de nuestra modificación de la RNC MobileNet_V3_Large (V2) para la clasificación de la base de datos de 96,000 imágenes. Ambas matrices muestran una correcta clasificación de las imágenes por este modelo. 72

7.17. A la izq. las gráficas de exactitud de entrenamiento (azul) y prueba (naranja) y a la der. el error de entrenamiento (azul) y prueba (naranja) durante el reentrenamiento de nuestra modificación de la RNC RegNet_X_1_6GF y nuestra base de datos con 96,000 imágenes. La línea punteada vertical representa la obtención del mejor modelo para esa prueba, el cual muestra valores de exactitud sin sobreajuste. . . . 73
7.18. Matrices de confusión de nuestra modificación de nuestra modificación de la RNC RegNet_X_1_6GF para la clasificación de la base de datos de 96,000 imágenes. La matriz de la izq. muestra una correcta clasificación de todas las imágenes del conjunto de entrenamiento, en la matriz de la der. se observa que solo las imágenes del conjunto de prueba de las clases óvalo y rectángulo fueron mal clasificada por este modelo. . 73
7.19. A la izq. las gráficas de exactitud de entrenamiento (azul) y prueba (naranja) y a la der. el error de entrenamiento (azul) y prueba (naranja) durante el reentrenamiento de nuestra modificación de la RNC ShuffleNet_V2_X0_5 y nuestra base de datos con 96,000 imágenes. La línea punteada vertical representa la obtención del mejor modelo para esa prueba, el cual muestra valores de exactitud sin sobreajuste. 74
7.20. Matrices de confusión de nuestra modificación de nuestra modificación de la RNC ShuffleNet_V2_X0_5 para la clasificación de la base de datos de 96,000 imágenes. La matriz de la izq. muestra una correcta clasificación de todas las imágenes del conjunto de entrenamiento, en la matriz de la der. se observa que solo las imágenes del conjunto de prueba de las clases óvalo y rectángulo fueron mal clasificada por este modelo. 74
7.21. Imágenes de la base de datos MNIST [15]. 75
7.22. A la izq. las gráficas de exactitud de entrenamiento (azul) y prueba (naranja) y a la der. el error de entrenamiento (azul) y prueba (naranja) durante el reentrenamiento de nuestra modificación de la RNC DenseNet121 con la base de datos MNIST. La línea punteada vertical representa la obtención del mejor modelo para esa prueba, el cual muestra valores de exactitud sin sobreajuste. 76
7.23. Matrices de confusión de nuestra modificación de nuestra modificación de la RNC DenseNet121 para la clasificación de la base de datos MNIST. Ambas matrices muestran una definida coloración en la diagonal principal, señal de una correcta clasificación para la mayoría de las imágenes. 76
7.24. A la izq. las gráficas de exactitud de entrenamiento (azul) y prueba (naranja) y a la der. el error de entrenamiento (azul) y prueba (naranja) durante el reentrenamiento de nuestra modificación de la RNC EfficientNet_B0 con la base de datos MNIST. La línea punteada vertical representa la obtención del mejor modelo para esa prueba, el cual muestra valores de exactitud sin sobreajuste. 77

7.25. Matrices de confusión de nuestra modificación de nuestra modificación de la RNC EfficientNet_B0 para la clasificación de la base de datos MNIST. Ambas matrices muestran una definida coloración en la diagonal principal, señal de una correcta clasificación para la mayoría de las imágenes. 77

7.26. A la izq. las gráficas de exactitud de entrenamiento (azul) y prueba (naranja) y a la der. el error de entrenamiento (azul) y prueba (naranja) durante el reentrenamiento de nuestra modificación de la RNC MobileNet_V3_Large con la base de datos MNIST. La línea punteada vertical representa la obtención del mejor modelo para esa prueba, el cual muestra valores de exactitud sin sobreajuste. 78

7.27. Matrices de confusión de nuestra modificación de nuestra modificación de la RNC MobileNet_V3_Large para la clasificación de la base de datos MNIST. Ambas matrices muestran una definida coloración en la diagonal principal, señal de una correcta clasificación para la mayoría de las imágenes. 78

7.28. Imágenes de la base de datos ISIC 2019 [5]. 79

7.29. Gráficas de exactitud y error durante el reentrenamiento de nuestra modificación de la RNC DenseNet121 con la base de datos de imágenes de melanomas ISIC 2019. A la izq. podemos observar un sobreajuste del modelo al incrementar únicamente la exactitud de modelo para el conjunto de entrenamiento (azul) pero no para el conjunto de prueba (naranja). A la der. observamos un aumento en la gráfica del error para el conjunto de prueba. La línea punteada vertical representa la obtención del mejor modelo para esa prueba, el cual muestra valores de exactitud sin sobreajuste. 80

7.30. Matriz de confusión de nuestra modificación de la RNC DenseNet121 para la clasificación de la base de datos de imágenes de melanomas ISIC 2019. Se pueden observar falsos positivos debido a la clasificación errónea de imágenes de lesiones benignas de queratosis como melanomas, así como falsos negativos por la clasificación errónea de imágenes de melanomas como lesiones benignas de queratosis. 80

7.31. Gráficas de exactitud y error durante el reentrenamiento de nuestra modificación de la RNC EfficientNet_B0 con la base de datos de imágenes de melanomas ISIC 2019. A la izq. podemos observar un sobreajuste del modelo al incrementar únicamente la exactitud de modelo para el conjunto de entrenamiento (azul) pero no para el conjunto de prueba (naranja). A la der. observamos un aumento en la gráfica del error para el conjunto de prueba. La línea punteada vertical representa la obtención del mejor modelo para esa prueba, el cual muestra valores de exactitud sin sobreajuste. 81

7.32. Matriz de confusión de nuestra modificación de la RNC EfficientNet_B0 para la clasificación de la base de datos de imágenes de melanomas ISIC 2019. Se pueden observar falsos positivos debido a la clasificación errónea de imágenes de lesiones benignas de queratosis como melanomas, así como falsos negativos por la clasificación errónea de imágenes de melanomas como lesiones benignas de queratosis. 81
7.33. Gráficas de exactitud y error durante el reentrenamiento de nuestra modificación de la RNC GoogLeNet con la base de datos de imágenes de melanomas ISIC. A la izq. podemos observar un sobreajuste del modelo al incrementar únicamente la exactitud de modelo para el conjunto de entrenamiento (azul) pero no para el conjunto de prueba (naranja). A la der. observamos un aumento en la gráfica del error para el conjunto de prueba. La línea punteada vertical representa la obtención del mejor modelo para esa prueba, el cual muestra valores de exactitud sin sobreajuste. 82
7.34. Matriz de confusión de nuestra modificación de la RNC GoogLeNet para la clasificación de la base de datos de imágenes de melanomas ISIC 2019. Se pueden observar falsos positivos debido a la clasificación errónea de imágenes de lesiones benignas de queratosis como melanomas, así como falsos negativos por la clasificación errónea de imágenes de melanomas como lesiones benignas de queratosis. 82
7.35. Gráficas de exactitud y error durante el reentrenamiento de nuestra modificación de la RNC MobileNet_V3_Large con la base de datos de imágenes de melanomas ISIC 2019. A la izq. podemos observar un sobreajuste del modelo al incrementar únicamente la exactitud de modelo para el conjunto de entrenamiento (azul) pero no para el conjunto de prueba (naranja). A la der. observamos un aumento en la gráfica del error para el conjunto de prueba. La línea punteada vertical representa la obtención del mejor modelo para esa prueba, el cual muestra valores de exactitud sin sobreajuste. 83
7.36. Matriz de confusión de nuestra modificación de la RNC MobileNet_V3_Large para la clasificación de la base de datos de imágenes de melanomas ISIC 2019. Se pueden observar falsos positivos debido a la clasificación errónea de imágenes de lesiones benignas de queratosis como melanomas, así como falsos negativos por la clasificación errónea de imágenes de melanomas como lesiones benignas de queratosis. 83

8.1. Ejecución en tiempo real de nuestra modificación de DenseNet121 para la clasificación de las figuras de la BDO. 89

Índice de tablas

3.1. Parámetros del fondo utilizado para generar el modelo de una gaussiana que genera los fondos aleatorios. 27

4.1. Resultados de las pruebas variando los parámetros del extractor HOG y la cantidad de características que extrajo de la imagen. 31

5.1. Modelos seleccionados del sitio web de Pytorch, las exactitudes que reportan para la clasificación de ImageNet1K y la cantidad de parámetros de estos[20]. 35
5.2. Número de parámetros de los modelos originales, el de nuestras modificaciones y el porcentaje que estas ultimas representas respecto al modelo original. 37

6.1. Valores de los parámetros de los clasificadores obtenidos con las redes neuronales de estado eco. 48
6.2. Exactitud del algoritmo Bayes Ingenuo en la clasificación de nuestra base de datos de imágenes utilizando las características HOG sin y con reducción a 128 y 64 dimensiones mediante ACP. 49
6.3. Exactitud del algoritmo Bayes Ingenuo en la clasificación de nuestra base de datos de imágenes utilizando las características HOG con reducción a 32 y 16 dimensiones mediante ACP. 50
6.4. Parámetros de la red neuronal para clasificar la base de datos con características HOG (ACP 64). 51

7.1. Resultados de nuestras modificaciones de RNC en la clasificación de nuestra base de datos aumentada 200 veces. 84
7.2. Resultados de nuestras modificaciones de RNC en la clasificación de la base de datos MNIST. 85
7.3. Resultados de nuestras modificaciones de RNC y los resultados reportados por Aljohani et al. [1] en la clasificación de la base de datos de melanomas ISIC. 85

8.1. Resultados de la clasificación en tiempo real con nuestra modificación la RNC DenseNet121. 90

Acrónimos

ABCDE: Asimetría, Borde, Color, Diámetro y Evolución.
ACP: Análisis de componentes principales.
BDA: Base de datos aumentada.
BDO: Base de datos original.
BFGS: Broyden-Fletcher-Goldfarb-Shanno.
DGE: Descenso de gradiente estocástico.
HOG: Histograma de Gradientes Orientados.
IA: Inteligencia Artificial.
ISIC: Colaboración Internacional de Imágenes de Piel.
LBP: Patrón binario local.
M: Millones.
MS: Milisegundo.
MVS: Máquinas de Vectores de Soporte.
NaN: Not a number.
ORT: Óvalo, rectángulo, triángulo.
POR: Pentágono, óvalo, rectángulo.
RBF: Núcleo gaussiano.
ReLU: Unidad lineal rectificada.
RNC: Red neuronal convolucional.
RND: Redes neuronales directas.
RNEE: Redes neuronales de estado eco.
SIFT: Transformación de características invariantes a la escala.
UPG: Unidad de procesamiento gráfico.

Capítulo 1

Introducción

1.1. Motivación del estudio

La investigación [29] presentada por Wang et al. sobre la comparativa de eficiencia entre Máquinas de Vectores de Soporte (MVS) y Redes Neuronales Convolucionales (RNC) en la clasificación de imágenes, destaca por su enfoque en el tiempo de ejecución y la exactitud. No obstante, la omisión de detalles críticos como los parámetros utilizados, el algoritmo de extracción de características para la MVS y la arquitectura específica de la RNC, plantea un desafío significativo para la reproducibilidad y la validación independiente de los resultados.

Este estudio se propone reproducir y extender los hallazgos de [29], proporcionando una documentación completa de los procedimientos y configuraciones. Se llevará a cabo un análisis comparativo entre algoritmos de aprendizaje automático tradicionales y de aprendizaje profundo, enfocándose en la exactitud y el tiempo de entrenamiento bajo condiciones controladas y uniformes.

Con el fin de evaluar la generalización y exactitud de los algoritmos, se generará una nueva base de datos de imágenes, diseñada para controlar el número de clases y el tamaño de las imágenes. Este enfoque permitirá una comparación más precisa y relevante de la exactitud de los modelos y el tiempo de entrenamiento.

Con el propósito de analizar que los algoritmos no solo sean eficientes para nuestra base de datos de imágenes, sino también puedan aplicarse de manera efectiva en otros conjuntos de imágenes, se seleccionaran los algoritmos que reporten mayor exactitud en la clasificación de nuestra base de datos de imágenes para clasificar las bases de datos de imágenes MNIST e ISIC 2019.

La base de datos ISIC 2019 contiene imágenes de melanoma y lesiones benignas de queratosis y ha sido utilizada para entrenar algoritmos de clasificación [1]. Nosotros utilizaremos nuestra metodología comparativa en este contexto de gran relevancia

social debido que la detección temprana del cáncer de piel es vital, y mediante este estudio, se busca demostrar cómo la Inteligencia Artificial (IA) puede convertirse en una herramienta valiosa para los profesionales de la salud, potenciando la precisión diagnóstica y, en consecuencia, mejorando el pronóstico y la calidad de vida de los pacientes.

1.2. Planteamiento del problema

La eficacia de los algoritmos de IA en la clasificación de imágenes es intrínsecamente dependiente de la calidad y la naturaleza de los datos utilizados para entrenarlos [16]. Diferentes conjuntos de datos, compuestos por variados tipos de imágenes, número de categorías y dimensiones, pueden resultar en divergencias significativas en los resultados de clasificación [12]. La selección del algoritmo influye directamente en el tiempo requerido para el entrenamiento y en la precisión alcanzada, especialmente cuando se emplean los mismos datos para entrenar y probar el modelo.

Es crucial entender los parámetros de implementación de los modelos de aprendizaje automático, tanto tradicionales como de aprendizaje profundo, así como el proceso de extracción de características y las especificidades de la base de datos en cuestión. Además, es esencial conocer los detalles de cómo se generan los conjuntos de entrenamiento y prueba para poder replicar con exactitud los resultados reportados.

A menudo, la literatura existente omite la divulgación completa de los parámetros utilizados en la comparación de tiempos de entrenamiento y precisión entre diferentes algoritmos de aprendizaje automático y aprendizaje profundo aplicados a la clasificación de imágenes [29]. Por lo tanto, este trabajo busca reproducir dichos resultados y proporcionar una descripción detallada de todos los parámetros involucrados, llenando así un vacío en el campo de estudio.

1.3. Objetivos de la investigación

La presente investigación se centra en la evaluación crítica de algoritmos de aprendizaje automático, tanto tradicionales como de aprendizaje profundo, con el fin de determinar su eficacia en la clasificación de imágenes. Este análisis no solo busca cuantificar la exactitud y los tiempos de ejecución, sino también proporcionar una guía práctica para la selección del algoritmo más adecuado según las necesidades específicas de cada caso. Los objetivos listados a continuación son el eje central sobre el cual gira este estudio comparativo.

Objetivo general

Calcular la exactitud y tiempos de ejecución de algoritmos de aprendizaje automático tradicionales y aprendizaje profundo en la clasificación de imágenes.

Objetivos particulares

1. Utilizar una Máquina de Vectores de Soporte (MVS), una red neuronal directa y una red neuronal recurrente tipo eco para realizar la clasificación.

2. Buscar las arquitecturas de redes profundas para realizar la clasificación de imágenes.

3. Utilizar una unidad de procesamiento gráfico (UPG) para entrenar redes neuronales convencionales y profundas.

4. Crear una nueva base de datos de imágenes para entrenamiento y prueba de los algoritmos.

5. Evaluar la exactitud y el tiempo de entrenamiento de los distintos algoritmos (Máquinas de Vectores de Soporte, redes directas, redes tipo eco y redes profundas).

6. Poder determinar cuándo usar un algoritmo de aprendizaje automático tradicional o un algoritmo de aprendizaje profundo.

1.4. Estructura de la tesis

La presente se estructura en ocho capítulos que guían al lector a través del desarrollo y los hallazgos de esta investigación. El capítulo 1 introduce la motivación detrás del estudio y define los objetivos específicos que se persiguen. El capítulo 2 se dedica al marco teórico, donde se detallan los conceptos fundamentales que forman la base del trabajo realizado.

En el capítulo 3, se describe el proceso de creación de la base de datos de imágenes propia, elemento crucial para el entrenamiento y evaluación de los modelos de clasificación. El capítulo 4 narra el trabajo experimental realizado con algoritmos de aprendizaje automático tradicional, detallando las metodologías y técnicas empleadas.

La exploración del aprendizaje profundo se presenta en el capítulo 5, donde se expone el desarrollo experimental de algoritmos avanzados, como las redes neuronales convolucionales. Se discute cómo estas fueron adaptadas para clasificar eficazmente nuestra base de datos de imágenes y se introduce el concepto de transferencia de

aprendizaje. Además, este capítulo aborda la problemática del sobreajuste en los modelos y presenta el método de aumento de datos como solución viable.

Los resultados obtenidos se documentan en los capítulos 6 y 7, donde se evalúa y compara el desempeño de los modelos de aprendizaje tradicional y profundo, respectivamente. El capítulo 8 sintetiza las conclusiones derivadas del estudio y propone sugerencias orientadas a investigaciones futuras.

Finalmente, se recopila las referencias bibliográficas que han sustentado teórica y metodológicamente la investigación.

Capítulo 2

Marco teórico

En el campo de la inteligencia artificial, se distinguen dos corrientes principales de algoritmos: el aprendizaje automático tradicional y el aprendizaje profundo, ambos aplicados con frecuencia en la clasificación de imágenes.

El aprendizaje automático tradicional se refiere a algoritmos que pueden aprender de los datos y hacer predicciones o tomar decisiones basadas en esos datos. Los datos se extraen de características específicas de procesamiento de imagen, o de mediciones adaptadas específicamente para resolver el problema bajo estudio (por ejemplo, para resolver un problema de clasificación). Los algoritmos suelen ser más simples y se basan en técnicas estadísticas que analizan patrones en los datos.

El aprendizaje profundo utiliza redes neuronales profundas con muchas capas y reciben como entrada los píxeles de las imágenes. Estas redes son capaces de aprender representaciones de datos de alto nivel de abstracción de manera automática, sin necesidad de extracción específicas de características. Esto es posible gracias a la capacidad de las redes neuronales para realizar un aprendizaje jerárquico, donde cada capa aprende a transformar sus entradas en un nivel de abstracción más alto que la capa anterior.

2.1. Aprendizaje automático tradicional

En el contexto de la clasificación de imágenes, el aprendizaje automático tradicional podría utilizar características extraídas manualmente de las imágenes, como bordes, texturas o colores, mediante algoritmos de extracción de características, para entrenar un modelo que pueda reconocer diferentes categorías de imágenes. Sin embargo, estos métodos requieren una intervención humana considerable para definir y extraer las características relevantes.

Entre los algoritmos de aprendizaje automático para clasificar imágenes sobresale

la MVS, las redes neuronales directas y las redes neuronales recurrentes de estado eco.

2.1.1. Bayes Ingenuo

Los clasificadores de Bayes Ingenuo son una familia de clasificadores probabilísticos simples y efectivos que se basan en aplicar el teorema de Bayes con la suposición de independencia entre las características.

El teorema de Bayes es el fundamento de este enfoque y se expresa matemáticamente como

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{2.1}$$

donde $P(A|B)$ es la probabilidad posterior de la clase A dado el predictor B, $P(B|A)$ es la probabilidad de que el predictor B pertenezca a la clase A, $P(A)$ es la probabilidad previa de la clase A y $P(B)$ es la probabilidad previa del predictor B [2].

En el contexto de clasificación, el objetivo es encontrar la clase más probable dado un conjunto de características. Esto se logra maximizando la probabilidad posterior $P(A|B)$.

Los clasificadores de Bayes Ingenuo son muy eficientes y requieren un pequeño número de datos de entrenamiento para estimar los parámetros necesarios para la clasificación. Sin embargo, su suposición de independencia entre las características es a menudo su mayor limitación, ya que en la práctica rara vez se cumple.

2.1.2. Máquina de Vectores de Soporte

La MVS es un algoritmo de aprendizaje supervisado que identifica un hiperplano óptimo para clasificar datos en un espacio de alta dimensión. El hiperplano es una generalización de una línea recta a múltiples dimensiones y se define matemáticamente como el conjunto de puntos que satisfacen la ecuación $\mathbf{w} \cdot \mathbf{x} + b = 0$, donde $\mathbf{w}$ es el vector de pesos, b es el sesgo y $\mathbf{x}$ es el vector de características [25].

El objetivo principal de MVS es maximizar el margen, que es la distancia entre el hiperplano y los puntos más cercanos de cada clase, conocidos como vectores de soporte; véase un ejemplo simple en la figura 2.1. Matemáticamente, esto se logra minimizando $\frac{1}{2}||\mathbf{w}||^2$ sujeto a las restricciones $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1$ para todos los puntos de datos i, donde y_i son las etiquetas de clase (igual a $+1$ para una clase y -1 para la otra clase en un problema de clasificación de dos clases) [3].

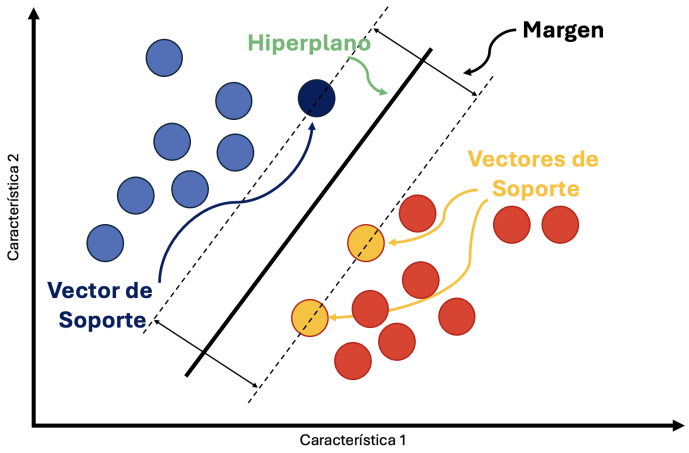


Figura 2.1: Ejemplo de MVS.

Es eficaz en casos donde el número de dimensiones es mayor que el número de muestras. Este modelo es particularmente útil para problemas de clasificación complejos y no lineales. Utiliza un núcleo para transformar datos no linealmente separables y maximiza el margen entre clases con la ayuda de vectores de soporte, resolviendo un problema de optimización cuadrática.

Cuando los datos no son linealmente separables, MVS utiliza el "truco del núcleo", que permite trabajar en un espacio de características de mayor dimensión sin tener que calcular las coordenadas de los datos en ese espacio. Esto se logra mediante el uso de funciones núcle, como el núcleo polinómico o el núcleo gaussiano (RBF), que calculan el producto interno de los datos en el espacio de características transformado.

El parámetro de regularización C equilibra la maximización del margen con la penalización de errores de clasificación, permitiendo cierta flexibilidad para manejar datos superpuestos y datos atípicos. Este enfoque proporciona una solución robusta y eficiente para problemas complejos de clasificación y regresión en Aprendizaje Automático. Un valor bajo de C hace que el margen sea grande, pero permite más violaciones del margen (en inglés el término es soft margin), mientras que un valor alto de C busca clasificar todos los puntos de entrenamiento correctamente a costa de un margen más pequeño [3].

Una vez entrenado el modelo, la clasificación de nuevos puntos se realiza evaluando el signo de la función de decisión $f(x) = \mathbf{w} \cdot \mathbf{x} + b$. Si $f(x) > 0$, el punto se clasifica en una clase, y si $f(x) < 0$, en la otra.

2.1.3. Redes neuronales

Para comprender cómo funcionan las redes neuronales, es esencial definir primero la unidad fundamental de estas: el Perceptrón. Este modelo simple es el precursor de las complejas estructuras que constituyen las redes neuronales modernas. El Perceptrón, inventado por Frank Rosenblatt en 1957, es un algoritmo de clasificación binaria que representa la forma más básica de una neurona artificial.

El Perceptrón opera recibiendo múltiples señales de entrada, cada una de las cuales se multiplica por un peso específico, y luego se suman para producir una señal única. Esta señal se pasa a través de una función de activación, que en el caso del Perceptrón es una función escalón, para determinar la salida. La expresión matemática de este proceso es:

$$f(\mathbf{x}) = \begin{cases} 1 & \text{si } \mathbf{w} \cdot \mathbf{x} + b > 0 \\ 0 & \text{en otro caso} \end{cases}$$

donde $\mathbf{w}$ representa el vector de pesos, $\mathbf{x}$ es el vector de entrada, b es el sesgo y $\cdot$ denota el producto punto entre dos vectores [22].

La decisión de si una entrada pertenece a una clase u otra se toma calculando el producto punto entre el vector de pesos y el vector de entrada, sumando el sesgo y aplicando una función escalón (véase figura 2.2).

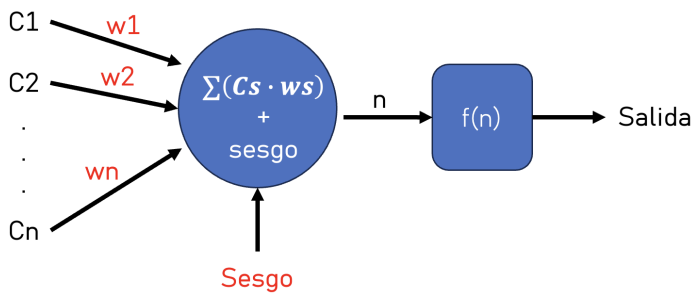


Figura 2.2: Representación del Perceptrón.

El proceso de aprendizaje del Perceptrón se realiza mediante el ajuste iterativo de los pesos $\mathbf{w}$ y el sesgo b, basándose en las etiquetas de entrada y los errores cometidos en la predicción de esta durante la fase de entrenamiento. La regla de actualización, conocida como la regla de aprendizaje de los pesos Perceptrón se expresa de la siguiente manera:

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \alpha(y - \hat{y})\mathbf{x}$$

y la actualización del sesgo por:

$$b(t+1) = b(t) + \alpha(y - \hat{y})$$

donde t es el índice de la época actual, α es la tasa de aprendizaje, y es la etiqueta verdadera y $\hat{y}$ es la predicción realizada por el Perceptrón [31].

El objetivo del aprendizaje es encontrar los valores de $\mathbf{w}$ y b que minimicen el número de clasificaciones incorrectas.

El Perceptrón es incapaz de resolver problemas que no son linealmente separables, como la función XOR. Esta limitación llevó al desarrollo de arquitecturas más avanzadas, utilizando múltiples capas y funciones de activación no lineales.

El Perceptrón sienta las bases para entender las redes neuronales, que son sistemas compuestos por múltiples neuronas interconectadas trabajando en conjunto para realizar tareas de clasificación y regresión más complejas.

Una red neuronal es una serie de capas compuestas por múltiples Perceptrones o neuronas artificiales. Cada neurona en una capa está conectada a todas las neuronas en la capa siguiente, formando una red densamente conectada que puede aprender patrones complejos en los datos.

Mientras que el Perceptrón utiliza una función de activación escalón, las redes neuronales modernas emplean una variedad de funciones de activación no lineales como la función sigmoide, la tangente hiperbólica y la función de unidad lineal rectificada (ReLU) [26]. Estas funciones permiten a las redes neuronales modelar relaciones no lineales y aprender de una manera más efectiva.

El aprendizaje en las redes neuronales se realiza a través de un proceso llamado retropropagación, donde los errores se propagan hacia atrás desde la salida hasta las capas de entrada, permitiendo ajustes finos en los pesos y sesgos de todas las neuronas [21].

Las redes de neuronales son aproximadores universales donde es muy importante el sesgo y que la función de salida de la red sea no lineal [24].

2.1.4. Redes neuronales de estado eco (RNEE)

A diferencia de las redes neuronales tradicionales, las RNEE se caracterizan por tener tres partes principales: la capa de entrada, el contenedor y la capa de salida. La capa de entrada conecta el mundo exterior con el contenedor. La capa de salida es donde se leen los estados del contenedor y se genera el vector de salida de la red [27].

El contenedor es el núcleo de la RNEE, compuesto por un gran número de neuronas con conexiones aleatorias y pesos no entrenables. La entrada se proyecta en este espacio dinámico, permitiendo que la red capture y procese la información temporal. Está diseñado para reflejar la historia de las entradas anteriores y sus interacciones. La dinámica del contenedor se describe mediante la ecuación:

La dinámica del contenedor se describe con la siguiente ecuación:

$$\mathbf{x}(t+1) = \tanh(\mathbf{W}^{in}\mathbf{u}(t) + \mathbf{W}\mathbf{x}(t))$$

donde $\mathbf{u}(t)$ es el vector de entrada en el tiempo t, $\mathbf{x}(t)$ es el vector de estado del contenedor en el tiempo t, $\mathbf{W}^{in}$ es la matriz de pesos de entrada y $\mathbf{W}$ es la matriz de pesos del contenedor, fija y aleatoria [11].

El entrenamiento de una RNE se centra en ajustar los pesos de la capa de salida $\mathbf{W}^{out}$, utilizando métodos como la regresión lineal para minimizar la diferencia entre la salida deseada y la salida producida por la red.

La salida de la red se calcula con la siguiente ecuación:

$$\mathbf{y}(t) = \mathbf{W}^{out}\mathbf{x}(t)$$

donde $\mathbf{y}(t)$ es el vector de salida en el tiempo t y $\mathbf{W}^{out}$ es la matriz de pesos de salida que se ajusta durante el entrenamiento.

La RNEE es una herramienta poderosa para el análisis y procesamiento de datos temporales, ofreciendo una alternativa eficiente a las RNR tradicionales [11].

2.2. Aprendizaje profundo

En la clasificación de imágenes, el aprendizaje profundo ha revolucionado el campo, ya que las redes neuronales convolucionales (RNC) pueden aprender directamente de los píxeles de la imagen y mejorar significativamente la precisión de la clasificación. Las RNC son especialmente poderosas porque pueden capturar la estructura espacial de las imágenes y aprender características complejas sin intervención humana.

La arquitectura de una RNC está diseñada con capas convolucionales, que procesan la entrada para extraer características, capas de agrupación, que reducen la dimensionalidad y concentran la información, y finalmente, una capa completamente conectada, que integra estas características para realizar la clasificación [30]. Esta estructura jerárquica permite que las RNC manejen con destreza tanto la concentración de los detalles como la abstracción de alto nivel, lo que las convierte en herramientas poderosas dentro del campo [6].

2.2.1. Convolución

Las capas convolucionales se caracterizan por sus filtros o núcleos, que actúan como detectores de rasgos. Estos filtros, generalmente de dimensiones 3×3, definen el tamaño del campo receptivo. Durante la convolución, se calcula el producto punto entre los píxeles de la imagen y el filtro (véase figura 2.3), generando así un mapa de características por cada filtro aplicado. Tras la convolución, se aplica la función de activación ReLU al mapa de características, introduciendo no linealidades al modelo.

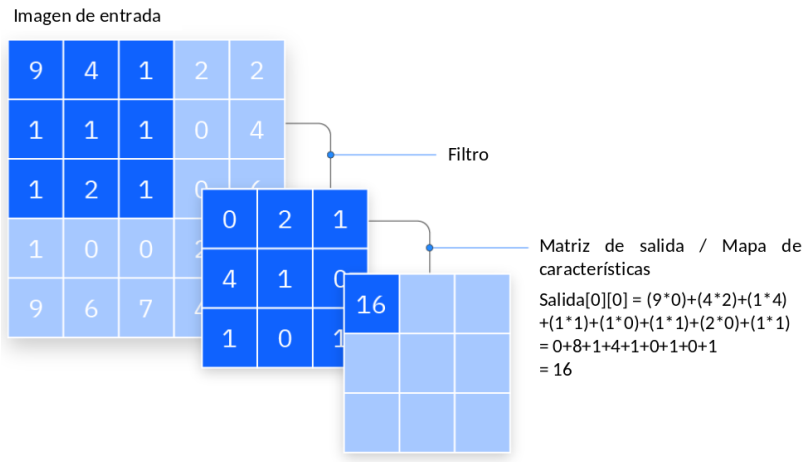


Figura 2.3: Ejemplo del proceso de convolución con una imagen de entrada de 5×5 píxeles, un filtro de tamaño 3×3 y como resultado un mapa de características de tamaño 3×3.

En PyTorch [20], la capa de convolución se implementa mediante la clase nn.Conv2d. Esta clase toma varios parámetros, entre ellos:

- in channels: Número de canales de entrada.
- out channels: Número de canales a la salida.
- kernel size: Tamaño del filtro (o kernel).
- stride: Número de píxeles por los que se desplaza el filtro en cada operación.
- padding: Cantidad de píxeles añadidos alrededor de la imagen para permitir que el filtro se aplique en los bordes.
- groups: Controla las conexiones entre entradas y salidas (valor 1 por defecto).

La capa nn.Conv2d aplica una convolución 2D sobre una señal de entrada compuesta por varios planos de entrada. Para un tamaño de entrada (N, C_{in}, H, W) y un

tamaño de salida $(N, C_{\text{out}}, H_{\text{out}}, W_{\text{out}})$, el valor de salida de la capa se puede describir como:

$$\text{out}(N_i, C_{\text{out}_j}) = \text{bias}(C_{\text{out}_j}) + \sum_{k=0}^{C_{\text{in}}-1} \text{weight}(C_{\text{out}_j}, k) * \text{input}(N_i, k)$$

donde N es el tamaño del lote (batch size), C denota el número de canales, H y W son la altura y el ancho de los planos de entrada en píxeles y $*$ es el operador de correlación cruzada 2D válido [20].

En el contexto de las redes neuronales convolucionales, el término *bias* se refiere a un sesgo que se añade a la salida de cada filtro para mejorar la capacidad de ajuste del modelo. Por otro lado, *weight* denota los pesos asociados a cada filtro, los cuales se aplican durante el proceso de convolución.

En cuanto a los grupos (groups), este parámetro controla las conexiones entre las entradas y las salidas [20]. Con un valor de *groups* igual a 1, cada entrada se convoluciona con todas las salidas. Con groups igual a 2, la operación se vuelve equivalente a tener dos capas convolucionales en paralelo, cada una procesando la mitad de los canales de entrada y generando la mitad de los canales de salida.

Por defecto el valor para los grupos es igual a uno [20], por lo que cada filtro se aplica a todos los mapas de características de entrada, y el resultado de cada convolución se suma para producir un único mapa de características de salida por filtro.

Es importante destacar que los valores resultantes en un mapa de características pueden exceder el rango típico de los píxeles de una imagen (0-255). Los mapas de características representan valores intermedios de cálculos matemáticos complejos y, por lo tanto, pueden asumir cualquier valor real, dependiendo de la entrada y de los parámetros definidos en la red.

2.2.2. Agrupamiento

La capa de agrupación reduce la dimensionalidad de los datos, disminuyendo la cantidad de parámetros y, por ende, la complejidad computacional [14]. Al igual que en la convolución, un filtro recorre la entrada, pero sin pesos asociados. Con un tamaño común de 2×2, este filtro reduce a la mitad los parámetros resultantes, simplificando así la representación de los datos.

Dependiendo de la función de agregación utilizada, se pueden obtener diferentes tipos de agrupación. Por ejemplo, la agrupación máxima selecciona el valor más alto dentro de la ventana del filtro, mientras que la agrupación promedio calcula el valor

medio. La figura 2.4 ilustra el proceso de agrupación máxima y agrupación promedio.

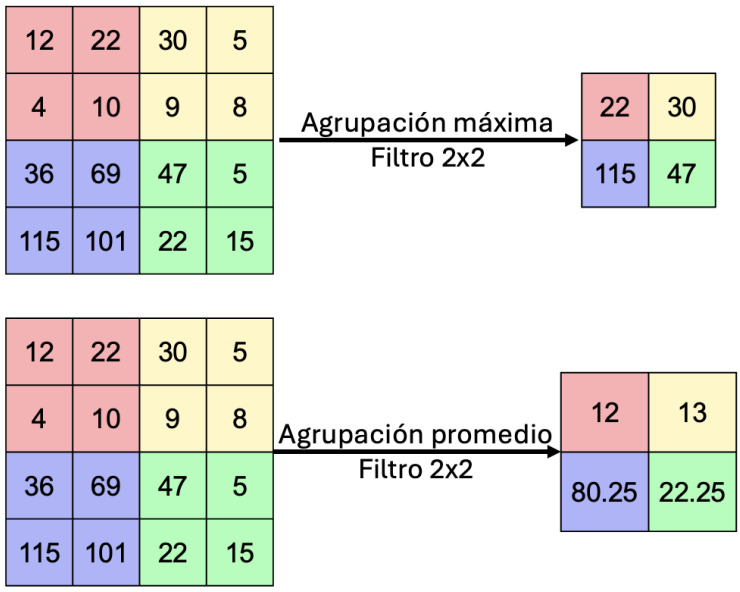


Figura 2.4: Ejemplo del proceso de agrupación máxima (arriba) y el de agrupación promedio (abajo), ambas con un filtro de tamaño 2×2.

Por otro lado, la capa nn.AdaptiveAvgPool2d de PyTorch implementa una técnica de agrupamiento promedio adaptativo [20]. A diferencia del agrupamiento promedio estándar, que requiere un tamaño de ventana predefinido, esta capa adapta dinámicamente el tamaño de la ventana para producir una salida con las dimensiones especificadas.

Al utilizar $nn.AdaptiveAvgPool2d((1,1))$, la capa agrupa cada canal del mapa de características de entrada a un único valor promedio, independientemente de las dimensiones de entrada originales [20]. Esta característica es particularmente útil para agrupar las dimensiones espaciales (ancho y alto) a un tamaño de 1×1, preservando la profundidad (número de canales). Esta operación es comúnmente utilizada antes de una capa completamente conectada para reducir la dimensionalidad y preparar los datos para la clasificación, véase un ejemplo en la figura 2.5.

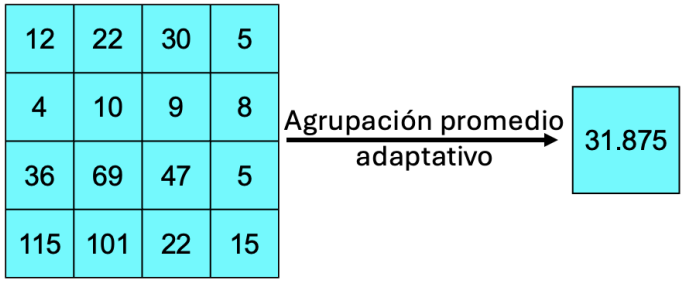


Figura 2.5: Ejemplo del proceso de agrupación promedio adaptativo 1×1.

2.2.3. Capa completamente conectada

La capa completamente conectada constituye un elemento esencial en la arquitectura de las redes neuronales, caracterizada por una interconexión total entre sus neuronas (véase figura 2.6). Esta capa se encarga de la clasificación final, sintetizando las características relevantes identificadas por las capas convolucionales previas mediante la aplicación de diversos filtros. Comúnmente, la función softmax se emplea en la capa de salida para asignar probabilidades a cada clase potencial, generando un vector cuyos valores se encuentran en el intervalo [0, 1] y cuya suma total es 1.

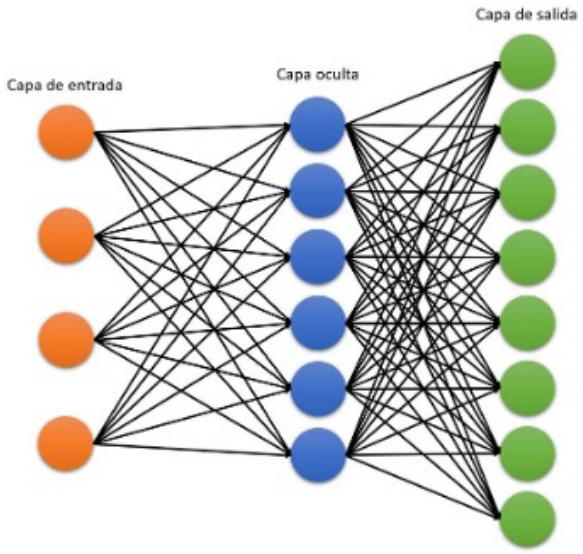


Figura 2.6: Ejemplo de capa completamente conectada.

2.3. Clasificación de imágenes

La clasificación de imágenes es una rama del aprendizaje automático y el aprendizaje profundo que se centra en la asignación de etiquetas a imágenes basadas en su contenido visual, es decir, en la identificación y análisis de características visuales que son distintivas y relevantes para diferenciar entre diversas categorías de imágenes. Estas características pueden incluir texturas, colores, formas y patrones espaciales. Los algoritmos de clasificación procesan estas características para asignar una etiqueta de clase a cada imagen.

La clasificación de imágenes es un proceso influenciado por diversos elementos, entre ellos, la elección del algoritmo de clasificación [19], además es imprescindible disponer de un conjunto de imágenes adecuado en cantidad y calidad para garantizar una clasificación eficaz.

En el aprendizaje automático tradicional, el algoritmo MVS ha sido ampliamente utilizada para la clasificación de imágenes. Este algoritmo requiere una etapa de extracción de características manual, donde los descriptores como SIFT, HOG y LBP, que son descriptores generales para imágenes, se emplean comúnmente [18, 24].

Con la llegada del aprendizaje profundo, las RNC han revolucionado el campo de la clasificación de imágenes. Las RNC son capaces de aprender jerarquías de características de manera automática, desde las más simples hasta las más complejas, lo que les permite capturar la esencia visual de las imágenes con gran precisión [13].

2.4. Revisión de literatura

En el estudio de Wang et al [29], se llevó a cabo una comparativa entre el desempeño de MVS y RNC en la tarea de clasificación de imágenes, utilizando las bases de datos MNIST y COREL1000. MNIST, compuesta por 70,000 imágenes de dígitos manuscritos de dimensiones 28×28 píxeles, y COREL1000, que incluye imágenes de 10 categorías distintas, cada una con 100 imágenes de dimensiones 384×256 o 256×384 píxeles.

Para la experimentación, se dividió MNIST en un conjunto de entrenamiento de 7,500 imágenes y un conjunto de prueba de 2,500 imágenes. En el caso de COREL1000, se etiquetaron las imágenes en 10 categorías, numeradas del 0 al 9, y se conformaron conjuntos de entrenamiento y prueba con 800 y 200 imágenes respectivamente, redimensionadas a 96×96 píxeles.

Se normalizaron las imágenes de COREL1000 a resoluciones de 64×64, 128×128 y 256×256 píxeles para evaluar el rendimiento de los algoritmos con distintos tamaños

de imagen. Además, se realizaron pruebas con diferentes combinaciones de categorías de imágenes, manteniendo un tamaño uniforme de 128×128 píxeles y seleccionando aleatoriamente grupos de 2, 4 y 6 categorías.

Los resultados indicaron que el tamaño de las imágenes influye significativamente en la precisión de la clasificación. La MVS alcanzó una exactitud del 86 % en la clasificación de COREL1000, superando el 83 % obtenido por la RNC. En contraste, para la base de datos de mayor volumen, MNIST, la RNC demostró ser superior, logrando una exactitud del 98 %, en comparación con el 88 % de la MVS.

No obstante, [29] no proporciona detalles cruciales como la arquitectura específica de la RNC ni el método de extracción de características utilizado para la MVS, lo que limita la capacidad de replicar sus hallazgos.

La extracción efectiva de características es un proceso crítico para el éxito de cualquier clasificador de imágenes. Técnicas como el Histograma de Gradientes Orientados (HOG), el Patrón Binario Local (LBP) y la Transformación de Características Invariantes a la Escala (SIFT) son utilizadas en el aprendizaje automático para el reconocimiento de patrones y la extracción de características.

En el trabajo de [23], se presenta una innovadora combinación de los descriptores de características HOG y LBP para la clasificación de imágenes en la base de datos COREL1000. Considerando imágenes de formato $227 \times 227 \times 3$, la dimensión del vector de características para HOG es de 1×26244, en contraste con 1×59 para LBP. Esta disparidad sugiere que HOG podría dominar el proceso de clasificación. Para equilibrar la influencia de ambos descriptores, se aplica el Análisis de Componentes Principales (ACP) para reducir la dimensionalidad del vector de HOG, igualándolo al de LBP, lo que permite que ambos contribuyan equitativamente al descriptor final. El ACP también optimiza el rendimiento computacional al simplificar el modelo y reducir los tiempos de entrenamiento.

El estudio de [8] propone un modelo tridimensional para la clasificación del conjunto de datos MNIST, que comprende preprocesamiento, extracción de características mediante HOG y clasificación a través de MVS lineal multiclase. La extracción de características se realiza dividiendo la imagen en celdas de 9×9 y calculando el histograma de orientaciones de gradiente, resultando en un vector de 81 características por dígito. Con 70,000 imágenes, divididas en 60,000 para entrenamiento y 10,000 para pruebas, el modelo alcanza una exactitud del 97.25 % en la clasificación.

La aplicación de algoritmos de clasificación de IA se extiende al ámbito médico, como en la detección de melanomas mediante análisis de imágenes digitales. La base de datos de cáncer de piel de la Colaboración Internacional de Imágenes de Piel (CIIP, o ISIC en inglés) facilita la capacitación, evaluación y comparación de algoritmos de

aprendizaje automático y profundo.

A pesar de la experiencia de los dermatólogos y el uso de criterios clínicos como Asimetría, Borde, Color, Diámetro y Evolución (ABCDE), las sensibilidades de prueba raramente superan el 80 % [4]. Las RNC han demostrado ser capaces de superar estos diagnósticos [9].

El cáncer de piel melanoma es una amenaza significativa para la vida humana, siendo crucial la detección temprana para mejorar el pronóstico del paciente. La supervivencia a cinco años para melanomas cutáneos menores de 1 mm es del 95 %, mientras que para melanomas ulcerados mayores de 4 mm es del 45 % [28].

Capítulo 3

Generación de base de datos de imágenes propia

3.1. Diseño

Se generó una base de datos de imágenes con el objetivo de tener un control del número de clases, la cantidad de imágenes por clase y las dimensiones en píxeles de cada una. Estas imágenes serán clasificadas mediante una variedad de métodos de aprendizaje automático, tanto tradicionales como de aprendizaje profundo.

Utilizando Inkspace, el software profesional de edición de gráficos vectoriales, se elaboraron ocho diseños de formas (véase figura 3.1), dentro de un círculo con un diámetro de 5 cm para delimitar el tamaño. Las ocho formas han sido etiquetadas de la siguiente manera:

- Cruz
- Estrella
- I
- Línea
- Óvalo
- Pentágono
- Rectángulo
- Triángulo

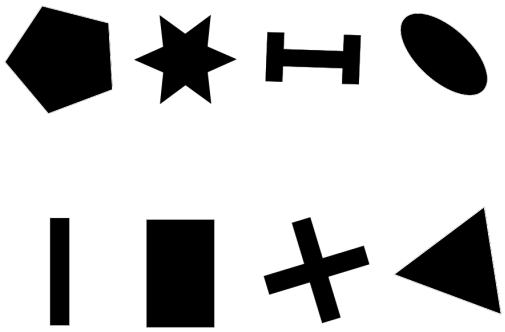


Figura 3.1: Formas de la base de datos.

Para las figuras Óvalo, Rectángulo y Triángulo se generaron seis instancias diferentes de cada forma (véase figuras 3.2, 3.3 y 3.4, respectivamente); el resto de las figuras únicamente fueron rotadas.

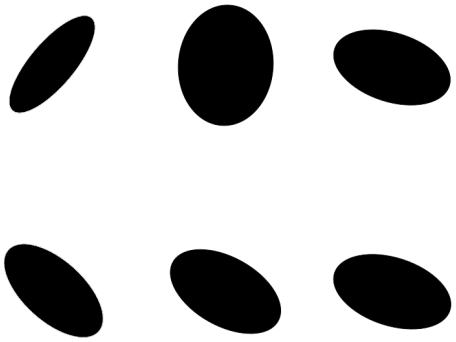


Figura 3.2: Instancias de la figura óvalo.

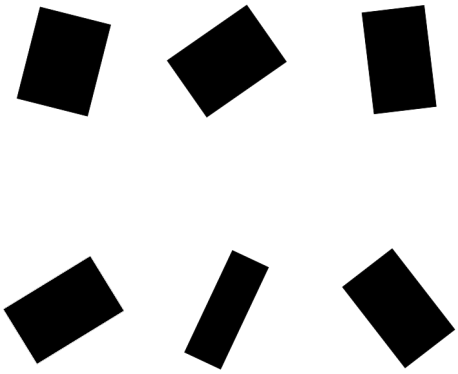


Figura 3.3: Instancias de la figura rectángulo.

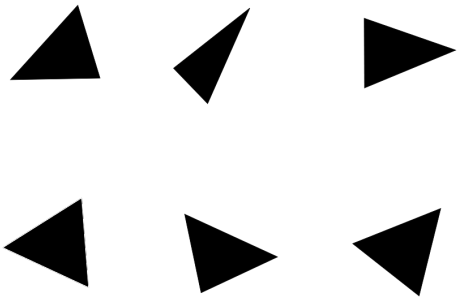


Figura 3.4: Instancias de la figura triángulo.

Estás fueron impresas con tecnología láser y las fotografías fueron realizadas mediante una cámara para la consola PlayStation 3 (véase figura 3.5), que tiene un valor de foco igual a 525.

Figura 3.5: Cámara de la PlayStation 3 utilizada para generar la base de datos.

3.2. Digitalización

Para que el tamaño de las imágenes de las formas fuese de 55×55 píxeles, la cámara se fijó en un trípode a una distancia de 47.72 cm perpendicular al fondo blanco sobre el que se colocaron las impresiones de las formas. Los cálculos de esta distancia se realizaron mediante el modelo de la cámara obscura.

Con base en el modelo de la cámara obscura:

$$\lambda p = K[R|t]P \tag{3.1}$$

donde λ es un factor de escala que se utiliza para normalizar la coordenada homogénea resultante después de la proyección; p es el punto en la imagen proyectada, en coordenadas homogéneas $(u, v, 1)^T$, con (u, v) como las coordenadas de la imagen 2D.

La matriz K contiene información sobre las características internas de la cámara, tales como la distancia focal y el punto principal. Tiene la siguiente forma:

$$K = \begin{pmatrix} f & 0 & -u_0 \\ 0 & f & -v_0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.2}$$

donde f es la distancia focal; u_0 y v_0 son las coordenadas del centro principal (o punto principal) de la imagen.

El término $[R|t]$ es la concatenación de una matriz de rotación R y un vector de traslación t. Este término describe la transformación del sistema de coordenadas del mundo 3D al sistema de coordenadas de la cámara. R es una matriz de rotación 3×3 que alinea el sistema de coordenadas del mundo con el sistema de coordenadas de la cámara. t es un vector de traslación 3×1 que desplaza el sistema de coordenadas del mundo al origen del sistema de coordenadas de la cámara.

P es el punto en el espacio 3D, representado en coordenadas homogéneas $[x, y, z, 1]^T$.

En este trabajo se utilizó un modelo tridimensional con un punto P sobre el eje y ubicado en las coordenadas $[0, 5, 0]^T$, el cual representa los 5 centímetros que mide el cuadro de referencia que contienen las figuras. Busca digitalizar esta distancia en 55 píxeles en el espacio bidimensional de la imagen. Para lograrlo, toma como referencia las coordenadas en píxeles del centro de las imágenes que captura la cámara que utiliza, la mitad de 640 para el eje x y 480 para el eje y, es decir, u_0 igual a 320 y v_0 igual a 240. Luego, suma a la coordenada en y la cantidad de píxeles que queremos que se proyecte, es decir 55. De esta manera, las coordenadas finales de la proyección en la imagen del punto p queda en $[320, 295]^T$. Este proceso permitió convertir el tamaño de nuestras figuras a su representación digital en píxeles.

Entonces,

$$
\begin{aligned}
\lambda p &= K[R|t]P \\
&= KR[I, -c]P \\
&= K[I, -c]P \\
\lambda \begin{bmatrix} 320 \\ 295 \\ 1 \end{bmatrix} &= K[I, -c] \begin{bmatrix} 0 \\ 5 \\ 0 \\ 1 \end{bmatrix} \\
&= K \begin{bmatrix} I, \begin{bmatrix} 0 \\ 0 \\ -c_z \end{bmatrix} \end{bmatrix} \begin{bmatrix} 0 \\ 5 \\ 0 \\ 1 \end{bmatrix} \\
&= \begin{bmatrix} f & 0 & -u_0 \\ 0 & f & -v_0 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -c_z \end{bmatrix} \begin{bmatrix} 0 \\ 5 \\ 0 \\ 1 \end{bmatrix} \\
&= \begin{bmatrix} f & 0 & -u_0 \\ 0 & f & -v_0 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ 5 \\ -c_z \end{bmatrix} \\
&= \begin{bmatrix} u_0 c_z \\ 5f + v_0 c_z \\ c_z \end{bmatrix} \\
\lambda \begin{bmatrix} 320 \\ 295 \\ 1 \end{bmatrix} &= \begin{bmatrix} 320 c_z \\ 5f + 240 c_z \\ c_z \end{bmatrix}.
\end{aligned}
\tag{3.3}
$$

De lo anterior se deduce que

$$\lambda = c_z. \tag{3.4}$$

Y que

$$295\lambda = 5f + 240c_z. \tag{3.5}$$

Entonces

$$\begin{aligned} c_z &= \frac{295\lambda - 5f}{240} \\ c_z &= \frac{295c_z - 5f}{240} \\ (295 - 240)c_z &= 5f \\ c_z &= \frac{5f}{55} \\ c_z &= \frac{f}{11} \\ c_z &= \frac{525}{11} \\ \Rightarrow c_z &= 47.72 \text{ cm.} \end{aligned} \tag{3.6}$$

Para las ocho clases de figuras, se generaron seis instancias de cada una, se rotaron al azar y se tomaban fotografías con la cámara. Así, la base de datos quedó conformada por 480 imágenes, con 60 de cada clase.

3.3. Aumento de datos

Con el propósito de verificar la generalización del conocimiento de los algoritmos, se generó otra base de datos del mismo tamaño, aplicando las operaciones de traslación, rotación y escala a las imágenes de la BDO, a esta técnica se le denomina *aumento de datos*. Las variables y los rangos que se utilizaron para estas operaciones se seleccionaron de forma aleatoria y se presentan a continuación.

- t_x y t_y son la cantidad de píxeles que se trasladará la imagen en el eje x y y, respectivamente, ambas entre -7 y 7 píxeles.
- Ángulo son los grados que se rotará la imagen, entre 0 y 360 grados.
- Escala es la proporción de la imagen original que se reducirá, entre 0.6 y 1.

Para evitar que las operaciones aplicadas sobre la imagen produzcan zonas oscuras, se debe extraer la figura de interés y superponerla sobre un fondo aleatorio. Los fondos se generan de manera aleatoria para cada nueva imagen, a partir del modelo de una distribución gaussiana. Calcula los parámetros de media y desviación estándar (véase los valores en la tabla 3.1) de la imagen del fondo en la parte superior izquierda de la figura 3.6. Con estos valores y la función *random.normal* de la biblioteca Numpy, generamos una nueva imagen de fondo de manera aleatoria. Dado que el fondo aleatorio se genera a partir de la estadística de la imagen original, se conserva la apariencia del entorno. En la figura 3.6 se puede ver un ejemplo de este proceso.

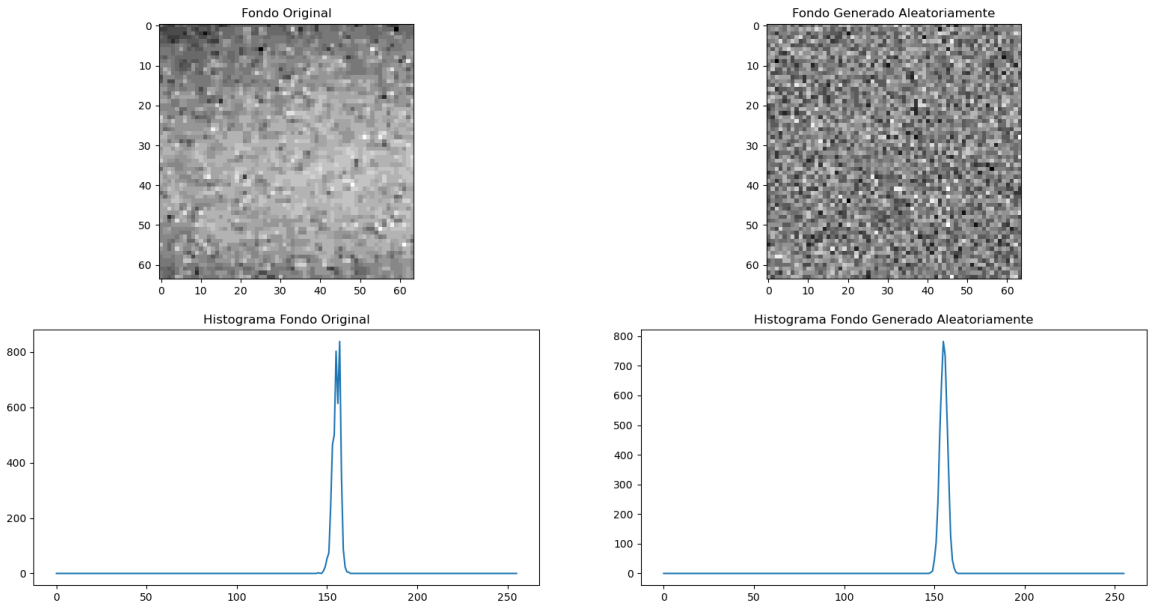


Figura 3.6: Estadísticas del fondo original y un ejemplo de los generados aleatoriamente.

Tabla 3.1: Parámetros del fondo utilizado para generar el modelo de una gaussiana que genera los fondos aleatorios.

Media	**Desviación estándar**
155.222412109375	2.093539085013309

La nueva base de datos aumentada (BDA) tiene la misma cantidad de imágenes y clases que la original, aunque con esta misma técnica se puede aumentar a cualquier número de imágenes que se requiera.

Capítulo 4

Algoritmos de aprendizaje automático tradicional

4.1. Métodos de extracción de características

4.1.1. Momentos de Hu

Para clasificar las imágenes de figuras de nuestra base de datos, se calcularon los momentos invariantes de Hu de cada una, haciendo uso de la función de la biblioteca OpenCV y se guardaron en un archivo junto con una etiqueta numérica para cada clase, la obtención de los siete momentos de Hu para cada imagen tomó aproximadamente de 20μs. Luego se observó que algunos momentos de Hu eran cercanos a cero, por lo que no aportaban información relevante que describiera a la forma, así que se eliminaron. Solo se conservaron las columnas 0, 1, 2, 3 y 6 como características para clasificar las figuras con los algoritmos de aprendizaje automático tradicional.

Los algoritmos con los que se realizó la clasificación de las imágenes de nuestra base de datos son: Bayes Ingenuo, MVS, Redes Neuronales y Redes Neuronales Recurrentes de Estado Eco.

4.1.2. Histograma de Gradientes Orientados (HOG)

También se usó el algoritmo HOG como descriptor de características. A diferencia de otros descriptores, el HOG tiene la singularidad de que el número de características que extrae de una imagen no es constante. En cambio, se define como el número total de bloques en la ventana de detección, multiplicado por el número de celdas por bloque, multiplicado por el número de contenedores de orientación.

El HOG se calcula mediante la siguiente ecuación:

$$HOG_{car} = \frac{I_w - B_w + S_w}{S_w} \cdot \frac{I_h - B_h + S_h}{S_h} \cdot \frac{B_w}{C_w} \cdot \frac{B_h}{C_h} \cdot N_{bins} \tag{4.1}$$

donde:

- I_w y I_h es el ancho y alto de la imagen en píxeles, respectivamente.
- B_w y B_h es el ancho y alto del bloque en píxeles, respectivamente.
- S_w y S_h es el ancho y alto del paso en píxeles, respectivamente.
- C_w y C_h es el ancho y alto de la celda en píxeles, respectivamente.
- N_{bins} es número de contenedores para los histogramas.

Este descriptor de características tiene las siguientes restricciones:

- Las dimensiones del paso deben ser divisibles del tamaño de la imagen.
- Las dimensiones de la celda deben ser divisibles del tamaño de la imagen.
- Las dimensiones de los pasos deben ser menor o igual que el tamaño del bloque.

Las principales desventajas de utilizar el descriptor HOG para extraer características de una imagen son:

- Se centra en la información local de los gradientes y no captura relaciones globales dentro de la imagen, lo que puede ser insuficiente para aplicaciones que requieren un entendimiento más profundo del contexto visual [17].
- Pede ser afectado por cambios en la iluminación, lo que altera los gradientes calculados y, por lo tanto, las características extraídas. Esto puede reducir la efectividad del algoritmo en condiciones de iluminación no controladas [7].

Búsqueda de los parámetros del extractor de características HOG

El uso del extractor de características HOG implica una gran complejidad, debido a las múltiples combinaciones posibles de los parámetros mencionados anteriormente, de los cuales depende el número de características que se extraerán. Por ello, se realizaron diversas pruebas de combinaciones aleatorias que extrajeran distintos números de características. Los parámetros del extractor y la cantidad de características extraídas de una imagen de 64×64 píxeles de nuestra base de datos se muestra en la tabla 4.1.

Tabla 4.1: Resultados de las pruebas variando los parámetros del extractor HOG y la cantidad de características que extrajo de la imagen.

Tamaño de bloque	**Tamaño de paso**	**Tamaño de celda**	**Número de contenedores**	**Número de características**
32×32	16×16	16×16	8	288
32×32	4×4	8×8	8	10,368
32×32	4×4	4×4	8	41,472
32×32	4×4	16×16	8	2,592
16×16	8×8	8×8	8	1,568
16×16	8×8	4×4	8	6,272
16×16	8×8	16×16	8	392
16×16	8×8	4×2	8	12,544
16×16	4×4	4×4	8	21,632
16×16	4×4	8×8	8	5,408
16×8	4×4	8×4	8	6,240
8×8	8×8	2×2	5	5,120
8×8	8×8	4×4	10	2,560
8×8	4×4	8×8	10	2,250
8×8	4×4	8×8	8	1,800
8×8	4×4	8×8	5	1,125
8×8	4×4	8×4	8	3,600

4.2. Bayes ingenuo

En el desarrollo de este estudio, también se usaron el modelo de Bayes Ingenuo, los cuales representan una técnica de clasificación estadística basada en el teorema de Bayes. Estos modelos fueron seleccionados por su eficiencia y simplicidad al manejar grandes conjuntos de datos. La implementación específica que se utilizó proviene de

la biblioteca Sklearn de Python.

Dentro de Sklearn, se empleó el modelo GaussianNB perteneciente al módulo naive_bayes. Este modelo asume que la probabilidad de las características es gaussiana. Se optó por mantener los parámetros del modelo en sus valores predeterminados para garantizar la reproducibilidad del experimento y facilitar la comparación con estudios similares. Esto implica que el parámetro priors se estableció en None, indicando que no se aplicó ninguna probabilidad a priori a las clases, y el parámetro var_smoothing se mantuvo en el valor de 1×10^{-9}, lo cual es una pequeña porción de la varianza máxima de todas las características que se añade a las varianzas para la estabilidad numérica durante el cálculo.

4.3. MVS

El preprocesamiento de los datos se realizó mediante la estandarización de las características, es decir, se transformaron los datos para que tuvieran una media cero y una desviación estándar igual a uno. Para esto se utilizó la clase StandardScaler de la biblioteca sklearn.preprocessing.

El modelo de MVS fue generado utilizando la clase SVC del módulo sklearn.svm, configurando el núcleo en rbf para permitir la clasificación no lineal. Se ajustaron los hiperparámetros gamma y C a 0.8 y 1000, respectivamente. El parámetro gamma define la influencia de un solo ejemplo de entrenamiento, con valores bajos que indican lejanía y valores altos que indican proximidad. El parámetro C actúa como un parámetro de regularización, proporcionando un equilibrio entre la correcta clasificación de ejemplos de entrenamiento y la maximización del margen de decisión.

4.4. Red neuronal directa

Se llevó a cabo la implementación de clasificadores basados en Redes Neuronales Directas (RND), reconocidas por su capacidad para modelar relaciones complejas y no lineales entre las características y las etiquetas de clasificación.

La implementación de la red neuronal se realizó utilizando la biblioteca scikit-learn de Python, aprovechando su módulo neural_network. Dentro de este módulo, se empleó la clase MLPClassifier, que permite la creación de redes neuronales multicapa con una arquitectura de propagación directa.

Para el preprocesamiento de los datos, se estandarizaron las características utilizando la clase StandardScaler del módulo sklearn.preprocessing. Este paso es funda-

mental para garantizar que las entradas de la red tengan una media de cero y una desviación estándar de uno, lo que contribuye a una convergencia más rápida y efectiva durante el entrenamiento.

El modelo de RND se configuró con una capa oculta de cuatro neuronas, utilizando el algoritmo lbfgs como solucionador. Este solucionador es una aproximación del método de optimización de BroydenFletcherGoldfarbShanno (BFGS), que es adecuado para modelos de tamaño pequeño a mediano. Se mantuvieron los valores predeterminados de los hiperparámetros para asegurar la reproducibilidad y facilitar la comparación con otros estudios similares. En particular, el parámetro max_iter se dejó en su valor predeterminado, lo que permite que el solucionador determine el número óptimo de iteraciones basado en la convergencia del modelo.

La elección de mantener una estructura simple y parámetros predeterminados se basó en la eficiencia y la simplicidad, permitiendo que el modelo sea robusto y menos propenso al sobreajuste. Además, la configuración de una semilla aleatoria fija, es decir, el parámetro random_state igual a 21, lo que garantiza que los resultados sean consistentes y verificables en futuras ejecuciones.

Capítulo 5

Algoritmos de aprendizaje profundo

5.1. Selección

Para los algoritmos de aprendizaje profundo se utilizaron como base las arquitecturas de las RNC disponibles en el sitio de la biblioteca Pytorch [20]. Esta biblioteca contiene arquitecturas de RNC de diferentes tamaños, es decir, con distintos números de parámetros. La selección de las arquitecturas de RNC que utilizamos se basó en el artículo [10], donde los autores proponen una arquitectura de una RNC que ellos llaman SimpleNet, la cual tiene 9.5 millones (M) de parámetros en su versión más grande y brinda una exactitud equiparable con la RNC AlexNet, que cuenta con 61.1 M de parámetros, en la clasificación de la base de datos ImageNet1K.

Por lo tanto, se determinó que el criterio para seleccionar las arquitecturas del sitio de Pytorch serían aquellas que tengan menos de 10 M de parámetros, en la tabla 5.1 se pueden observar los nombres de las arquitecturas, así como el valor de exactitud *Acc*@1 y *Acc*@5. El *Acc*@1 (Top-1 Accuracy) mide la exactitud del modelo al predecir la clase correcta como la opción más probable y *Acc*@5 (Top-5 Accuracy), por otro lado, evalúa si la clase correcta está entre las cinco opciones más probables, ambas respecto a la clasificación de la base de datos ImageNet1K. En la ultima columna se muestra la cantidad de parámetros entrenables de cada modelo.

Tabla 5.1: Modelos seleccionados del sitio web de Pytorch, las exactitudes que reportan para la clasificación de ImageNet1K y la cantidad de parámetros de estos[20].

Modelo	**Acc@1**	**Acc@5**	**Parámetros (M)**
DenseNet121	74.434	91.972	8.0
EfficientNet_B0	77.692	93.532	5.3
EfficientNet_B1 (V2)	79.838	94.934	7.8
EfficientNet_B2	80.608	95.31	9.1
GoogLeNet	69.778	89.53	6.6

MNASNet0_5	67.734	87.49	2.2
MNASNet0_75	71.18	90.496	3.2
MNASNet1_0	73.456	91.51	4.4
MNASNet1_3	76.506	93.522	6.4
MobileNet_V2 (V2)	72.154	90.822	3.5
MobileNet_V3_Large (V2)	75.274	92.566	5.5
MobileNet_V3_Small	67.668	87.402	2.5
RegNet_X_1_6GF (V2)	79.668	94.922	9.2
RegNet_X_400MF (V2)	74.864	92.322	5.5
RegNet_X_800MF (V2)	77.522	93.826	7.3
RegNet_X_400MF (V2)	75.804	92.742	4.3
RegNet_X_800MF (V2)	78.828	94.502	6.4
ShuffleNet_V2_X0_5	60.552	81.746	1.4
ShuffleNet_V2_X1_0	69.362	88.316	2.3

5.2. Modificaciones de las arquitecturas

Debido a que las imágenes de nuestra base de datos son de 64×64 píxeles fue necesario calcular el tamaño de los mapas de características a la salida de cada capa de los modelos, con el propósito de saber hasta que capa de cada arquitectura necesitábamos realizar la transferencia de aprendizaje, para esto se utilizaron las siguientes ecuaciones que calculan el tamaño de salida del mapa de características de una capa convolucional y una capa de agrupación.

El tamaño de salida de una capa Conv2d de Pytorch se calcula como:

$$n_s = \frac{n_e - k + 2p}{s} + 1$$

donde n_s es el tamaño de salida, n_e es el tamaño de entrada, k es el tamaño del núcleo, p es el relleno que se le aplica a la imagen y s es el paso.

El calculo del tamaño de salida de una capa MaxPool2d esta dado por la siguiente ecuación:

$$n_s = \frac{n_e + 2p - d(k-1) - 1}{s} + 1$$

donde n_s es el tamaño de salida, n_e es el tamaño de entrada, k es el tamaño del núcleo, p es el relleno que se le aplica a la imagen, s es el paso, d es la dilatación.

Una vez seleccionadas las capas de la sección de extracción de características de las RNC para nuestras arquitecturas, se procedió a modificar la última capa de la sección encargada de realizar la clasificación. La cantidad de neuronas en la última capa corresponde a la cantidad de clases de la base de datos imágenes que se clasificará.

Las salidas de estas neuronas forman el vector con los grados de pertenencia de la imagen de entrada a cada una de las clases.

5.3. Comparativa de tamaño entre las arquitecturas del sitio de Pytorch y nuestras propuestas

Posterior a la generación de nuestras arquitecturas, se realizó un resumen de cada una de ellas para comparar la cantidad de parámetros. En la tabla 5.2 se muestra el nombre original de las arquitecturas. La segunda columna presenta el número de parámetros entrenables de la arquitectura original. En la tercera columna se indica el número de parámetros entrenables de nuestras modificaciones a esas arquitecturas. Finalmente, en la última columna, se muestra el porcentaje que representan los parámetros de nuestras arquitecturas respecto al número de parámetros de la arquitectura original en el sitio de Pytorch.

Tabla 5.2: Número de parámetros de los modelos originales, el de nuestras modificaciones y el porcentaje que estas ultimas representas respecto al modelo original.

Modelo	**Hub**	**Propuesta**	**Porcentaje**
DenseNet121	7,978,856	1,296,392	16.25
EfficientNet_B0	5,288,548	73,162	1.38
GoogLeNet	6,624,904	276,392	4.17
MNASNet0_5	2,218,512	131,968	5.95
MobileNet_V3_Large (V2)	5,483,032	347,656	6.34
RegNet_X_1_6GF (V2)	9,190,136	53,208	0.58
ShuffleNet_V2_X0_5	1,366,792	62,064	4.54
SimpleNet_5m_m2	5,752,808	534,588	9.29

5.4. Transferencia de aprendizaje

Los modelos en Pytorch seleccionados cuentan con la opción de cargar el modelo con todos sus parámetros inicializados aleatoriamente o cargar los parámetros preentrenados con la base de datos ImageNet1K. Durante el entrenamiento con la base de datos ImageNet1K, los modelos aprendieron las características para clasificar 1000 clases, en las primeras capas de convolucionales estos modelos aprendieron las características generales de las imágenes y entre más profunda sea la capa, las características aprendidas son más abstractas. Podemos deducir con base en los valores de exactitud que reportan estos modelos en la clasificación de la base de datos ImageNet1K,

que los parámetros de las capas previas a la red completamente conectada son aquellos que brindaron la mejor extracción de características para cada uno de los modelos.

Con el propósito de minimizar el tiempo de entrenamiento de nuestras arquitecturas de RNC, se decidió cargar los modelos preentrenados de Pytorch y transcribir los parámetros de la etapa de extracción de características de estos modelos a los nuestros, es decir, realizar una transferencia de aprendizaje. Tomando como referencia el nombre de cada capa de la arquitecturas, se transfirieron los valores de los núcleos de convolución y los sesgos a nuestros modelos.

Capítulo 6

Desempeño de los modelos tradicionales

En este capítulo se presentan los resultados de la clasificación de la base de datos de imágenes original que nosotros creamos, la cual fue dividida en un 75 % para entrenamiento (360 imágenes) y un 25 % para prueba (120 imágenes) utilizando modelos de aprendizaje automático tradicionales. Estos resultados se organizan según el algoritmo de extracción de características utilizado. Finalmente, se muestra el desempeño de la generalización del conocimiento al clasificar la BDA desarrollada a partir de nuestra BDO.

6.1. Momentos de Hu

6.1.1. Bayes ingenuo

Se entrenó un modelo de clasificación de imágenes con el algoritmo de aprendizaje automático tradicional Bayes Ingenuo, usando los momentos invariantes de Hu de las figuras de nuestra base de datos como características. El entrenamiento del modelo duró 1.5 milisegundos (ms). El modelo presentó dificultades para clasificar las imágenes correspondientes a clases Óvalo, Rectángulo y Triángulo, lo que se refleja en una exactitud de 0.869 en el conjunto de entrenamiento, 0.891 en el de prueba y un valor de 0.86 en una validación cruzada con cinco dobleces. En la figura 6.1 se muestra la matriz de confusión de este modelo.

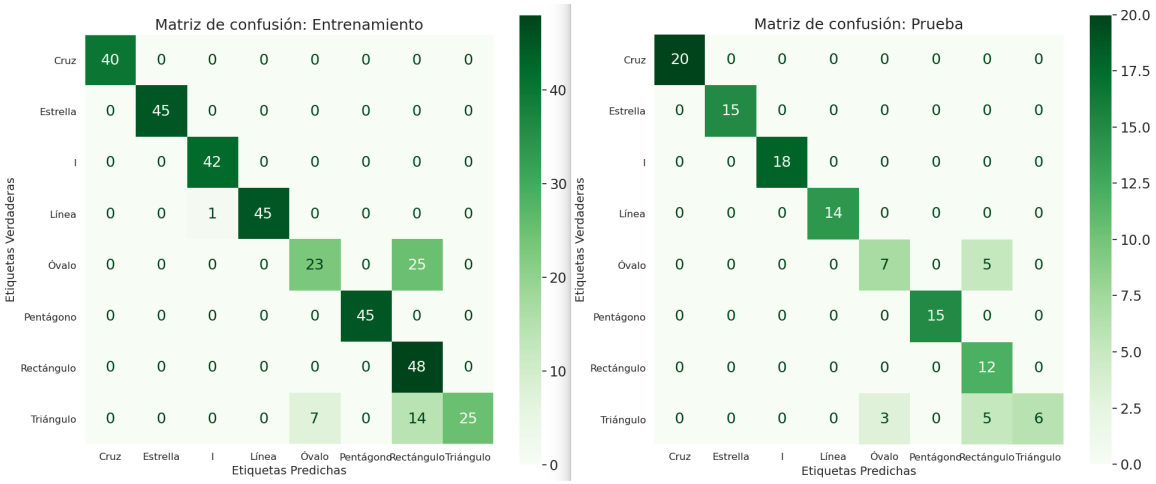


Figura 6.1: Matriz de confusión del modelo de Bayes Ingenuo utilizando como características los momentos invariantes de Hu.

Debido a que el modelo de Bayes Ingenuo entrenado con los momentos invariantes de Hu como características, no clasificaba correctamente todas las imágenes de nuestra base de datos, se optó por implementar dos modelos distintos con el mismo algoritmo de aprendizaje automático tradicional. El primero clasifica las imágenes con las etiquetas:

- Cruz.
- Estrella.
- I.
- Línea.
- Pentágono.
- ORT (Óvalo o Rectángulo o Triángulo).

El entrenamiento del primer modelo duró 1.4ms y logró una exactitud de 1 para ambos conjuntos de datos, el de entrenamiento y el de prueba. La matriz de confusión de este modelo se muestra en la figura 6.2.

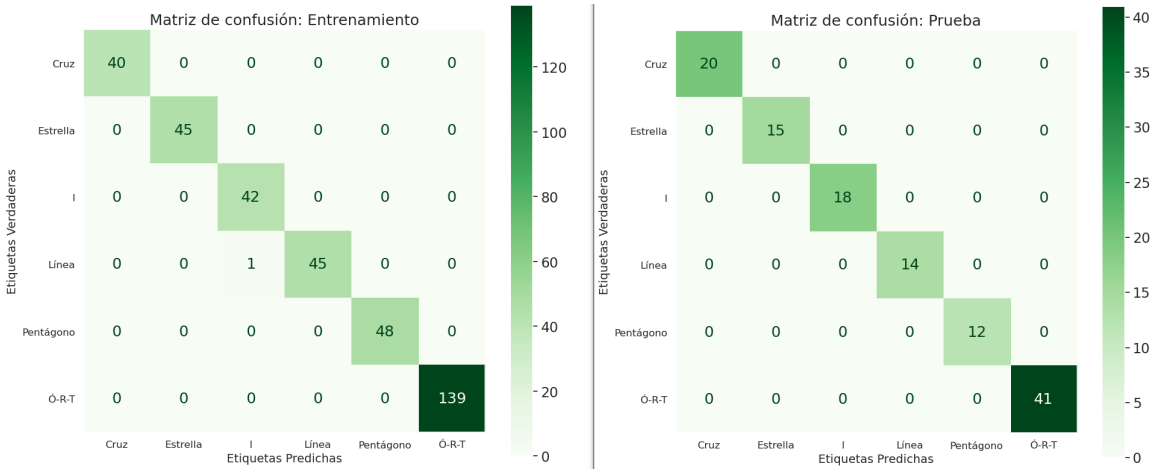


Figura 6.2: Matrices de confusión del primer clasificador de Bayes Ingenuo con los momentos de Hu. Ambas matrices muestran valores distintos a cero únicamente en la diagonal principal, lo que indica una correcta clasificación de todas las imágenes.

El segundo modelo de Bayes Ingenuo se encarga de clasificar las figuras de las clases ORT. Las características extraídas de las figuras de estas tres clases con las que se entrenó este segundo modelo son las siguientes:

- c_1 es la cantidad de píxeles del lado izquierdo del perímetro de la figura.
- c_2 es la cantidad de píxeles del lado derecho del perímetro de la figura.
- La relación entre en área de una elipse ajustada a los puntos del perímetro y el área de la figura.

Las características c_1 y c_2, que representan la cantidad de píxeles a la izquierda y a la derecha del perímetro de la figura, se calcularon buscando el primer píxel blanco del perímetro, etiquetado como v_1. A continuación, se identificó el píxel más alejado de v_1, que se etiquetó como v_2. Luego, se ordenaron los píxeles del perímetro a la derecha y a la izquierda de la línea que une los vértices v_1 y v_2. Finalmente, se sumó la cantidad de píxeles a la izquierda y se asignó este valor a la característica c_1, mientras que la suma de los píxeles a la derecha se asignó a la característica c_2.

La relación entre áreas se calcula ajustando una elipse al perímetro de la figura. Posteriormente, se calcula el área de la elipse y el área de la figura, y se obtiene la diferencia entre estas. Finalmente, esta diferencia se normaliza para obtener una relación entre ambas áreas y este valor es la tercera característica que utiliza el segundo modelo para clasificar las figuras ORT.

El tiempo de entrenamiento del segundo modelo de clasificación duró 0.9ms y obtuvo una exactitud de 1.0 tanto en el conjunto de entrenamiento como en el de prueba, véase la matriz de confusión de la figura 6.3.

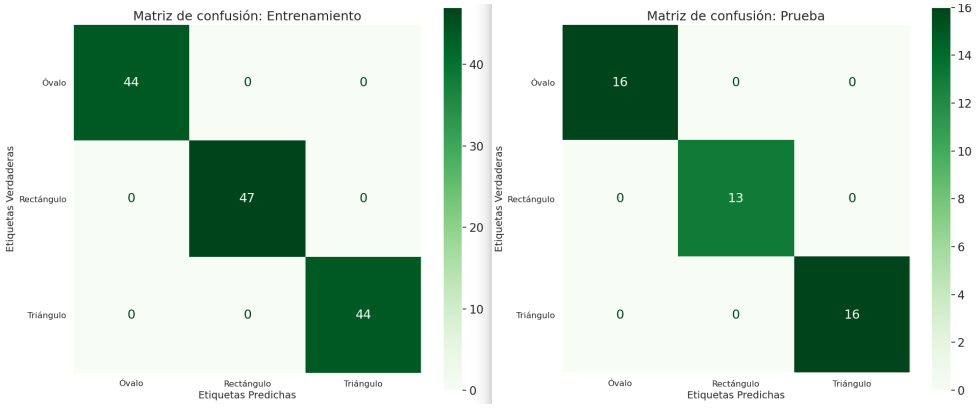


Figura 6.3: Matrices de confusión del segundo clasificador de Bayes Ingenuo con las características c_1, c_2 y relación de áreas. Ambas matrices muestran valores distintos a cero únicamente en la diagonal principal, lo que indica una correcta clasificación de todas las imágenes.

Se usaron ambos clasificadores en conjunto obteniendo un valor de 1 en la validación cruzada con cinco dobleces en la clasificación de nuestra base de datos y en la figura 6.4 se muestra la matriz de confusión.

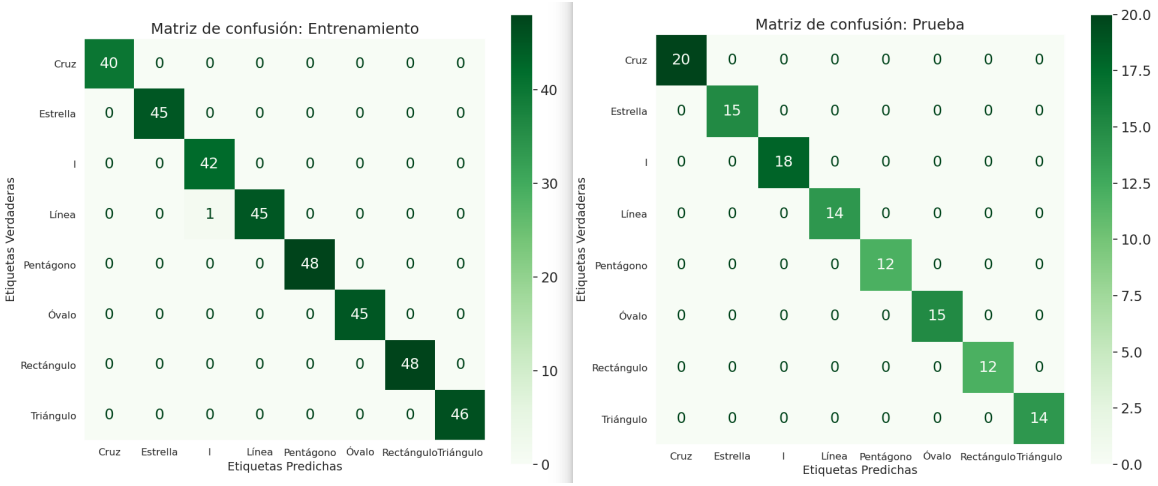


Figura 6.4: Matrices de confusión de los modelos de Bayes Ingenuo con características de los momentos invariantes de Hu y las características c1,c2 y área. En la matriz para el conjunto de entrenamiento se observa un único error en la clasificación de una imagen de la clase línea, etiquetado como de la clase i. La matriz para el conjunto de prueba muestra una correcta clasificación para todas las imágenes.

6.1.2. Máquina de Vectores de Soporte

Para clasificar las figuras de la BDO, se empleó el algoritmo de Máquinas de Vectores de Soporte SVC (se usó el módulo MVS.SVC de Sklearn) con los momentos de Hu como características. El modelo de MVS.SVC se ajustó con un núcleo rbf, un parámetro gamma de 0.8 y un parámetro C de 1000, el resto de parámetros se mantuvieron por defecto, es decir, degree igual a 3, coef0 igual a 0.0, shrinking igual a Verdadero, probabilidad igual a Falso, tamaño de cache igual a 200, peso de clase igual a Nulo, etc. El entrenamiento de este modelo duró 15μs y la evaluación mediante validación cruzada de cinco dobleces arrojó un valor de exactitud de 1, lo que significa que todas las imágenes de nuestra base de datos fueron clasificadas correctamente. La matriz de confusión de este modelo se presenta a continuación en la figura 6.5.

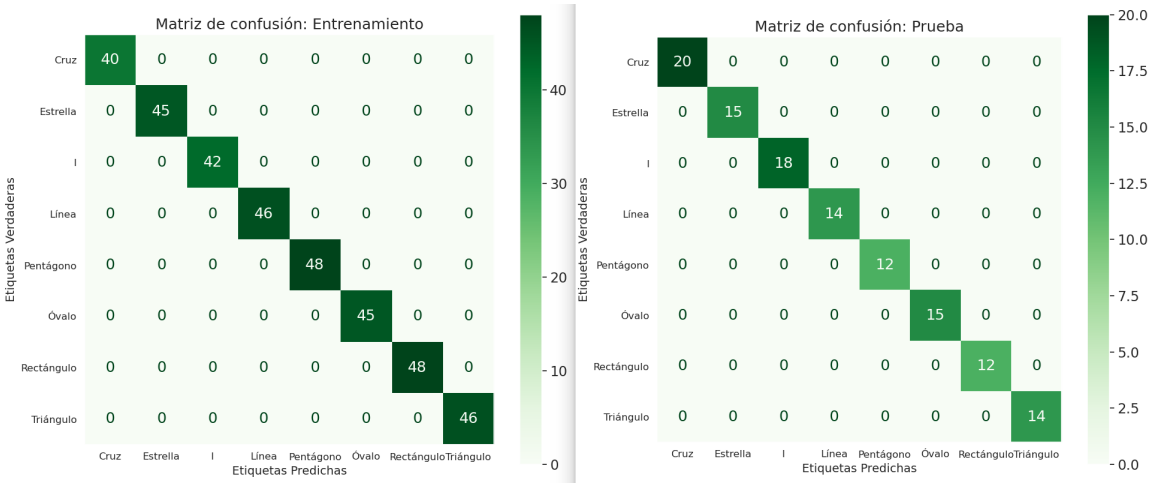


Figura 6.5: Matrices de confusión del modelo de la MVS.SVC con momentos de Hu. En ambas matrices se observa la presencia de números diferentes a cero únicamente en la diagonal principal, lo que indica una correcta clasificación de todas las imágenes.

6.1.3. Clasificador con redes neuronales directas

Para la clasificación de las imágenes de la BDO con redes neuronales se utilizó el clasificador de perceptrones multicapa (en inglés, Multi-layer Perceptron Classifier) de la biblioteca Sklearn.

El modelo consta de tres capas de neuronas: una capa de entrada, una capa oculta y una capa de salida. La arquitectura de la red neuronal se muestra en la figura 6.6.

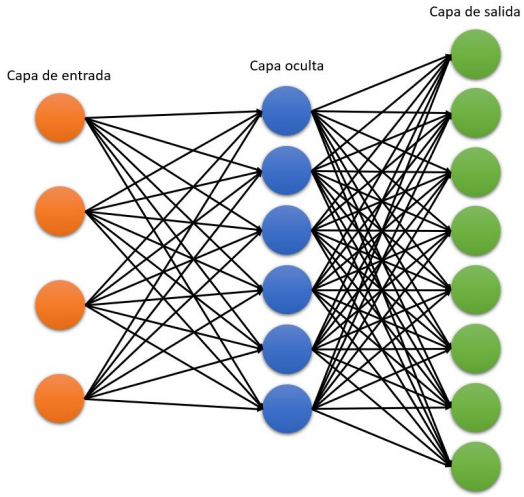


Figura 6.6: Arquitectura de la red neuronal.

Las neuronas de las capas de entrada y oculta utilizan la función de activación ReLU, que permite una mayor velocidad de aprendizaje y evita el problema de la desaparición del gradiente. La capa de salida utiliza la función softmax, que asigna las probabilidades de pertenencia a cada una de las clases posibles. El modelo fue entrenado durante 200 épocas, utilizando el algoritmo de optimización LBFGS con una tasa de aprendizaje de 0.001.

El entrenamiento de este modelo duró 0.7ms y clasificó correctamente todas las imágenes del conjunto de entrenamiento y del conjunto de prueba. Esto significa que tiene un valor de exactitud de 1. En la figura 6.7 se muestra la matriz de confusión de la red neuronal.

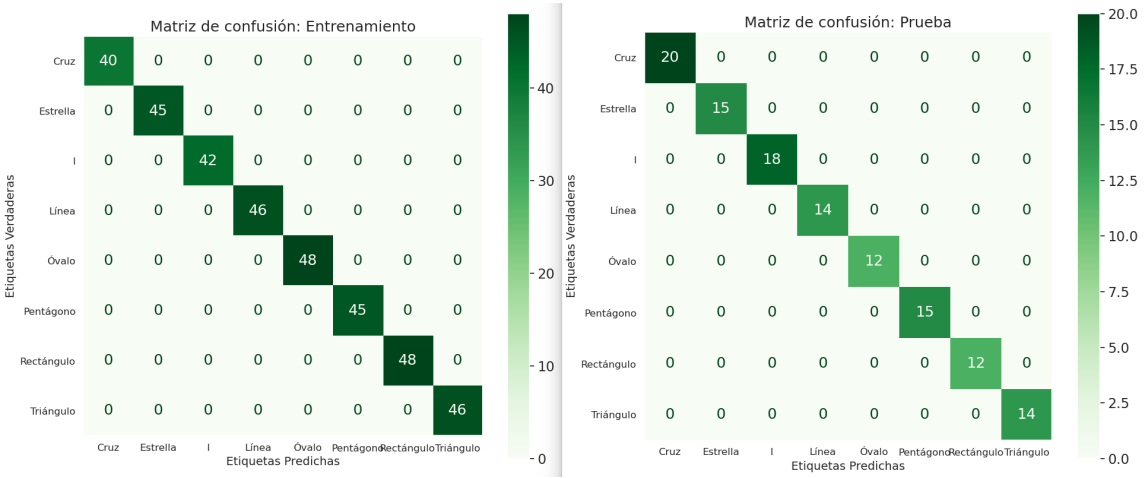


Figura 6.7: Matrices de confusión de la red neuronal directa. En ambas matrices se observa la presencia de números distintos a cero únicamente en la diagonal principal, lo que señala la correcta clasificación de todas las imágenes.

6.1.4. Clasificador con redes neuronales recurrentes de estado eco

Clasificador con una sola red

Para encontrar los parámetros de la red neuronal de estado eco se utilizó el algoritmo genético multi-objetivo NSGA-II, usando los momentos invariantes de Hu como características de las figuras. El valor máximo de exactitud fue de 0.7 para el conjunto de entrenamiento y de 0.8 para el conjunto de prueba. La matriz de confusión de este modelo se muestra en la figura 6.8. En ella se observa que las clases que se clasifican incorrectamente son: pentágono, óvalo y rectángulo. Por lo tanto, al igual que con el algoritmo de Bayes ingenuo, se optó por utilizar dos clasificadores distintos.

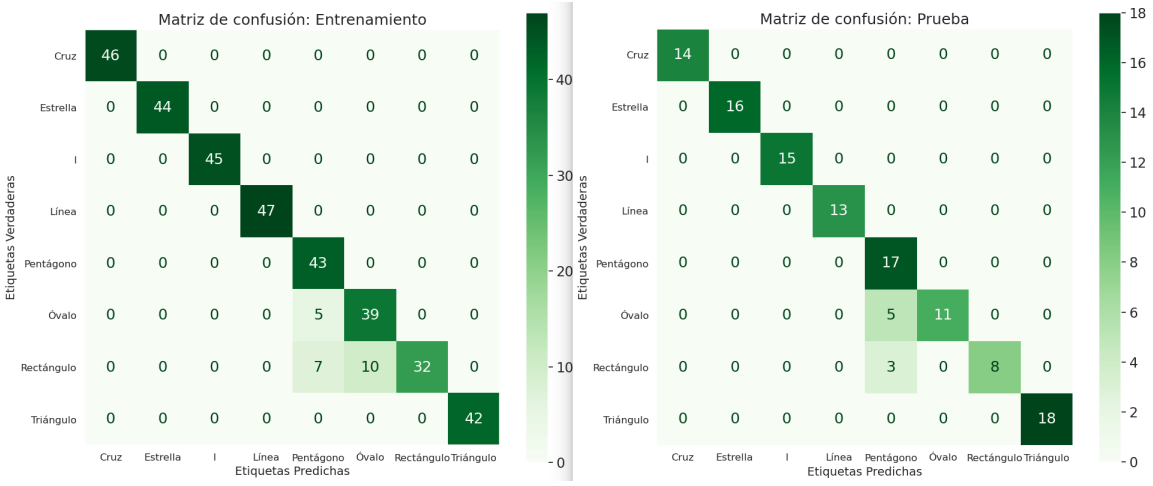


Figura 6.8: Matriz de confusión de la red neuronal de estado eco con los momentos de Hu como características. Se observa en ambas matrices que las únicas clases las etiquetadas son óvalo y rectángulo.

Clasificador con dos redes

Se empleó el algoritmo NSGA-II para optimizar los parámetros de la primera red de estado eco. El objetivo de esta red es clasificar figuras según las etiquetas que se describen a continuación:

- Cruz.
- Estrella.
- I.
- Línea.

- Triángulo.
- POR (Pentágono, Óvalo, Rectángulo).

Este nuevo modelo clasifica correctamente las imágenes según las etiquetas descritas previamente, alcanzando una exactitud de 1.0. La matriz de confusión de la figura 6.9 evidencia este resultado.

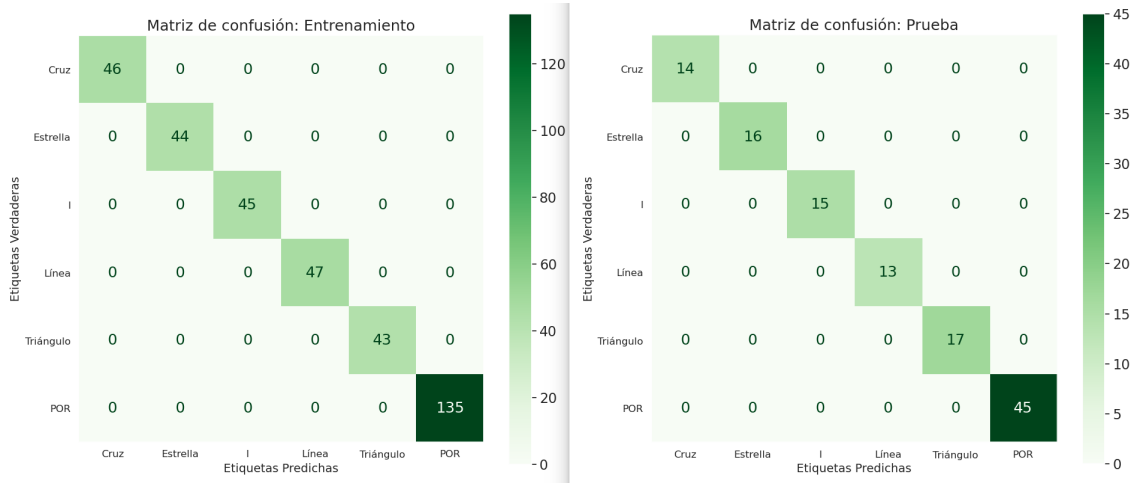


Figura 6.9: Matriz de confusión de la primera red neuronal de estado eco con los momentos de Hu como características (dos clasificadores en total). La presencia de valores distintos de cero únicamente en la diagonal principal denota la correcta clasificación de todas las imágenes.

Para clasificar las tres clases restantes, se diseñó un segundo modelo que también se sometió a una optimización de parámetros con el algoritmo NSGA-II. No obstante, este modelo solo logró una exactitud de 0.977 y 0.911 para el conjunto de entrenamiento y prueba, respectivamente. No logra clasificar correctamente las clases Pentágono y Óvalo, como se muestra en la matriz de confusión de la figura 6.10. Por esta razón, se decidió crear un nuevo modelo de red neuronal de estado eco que pudiera distinguir adecuadamente estas dos clases.

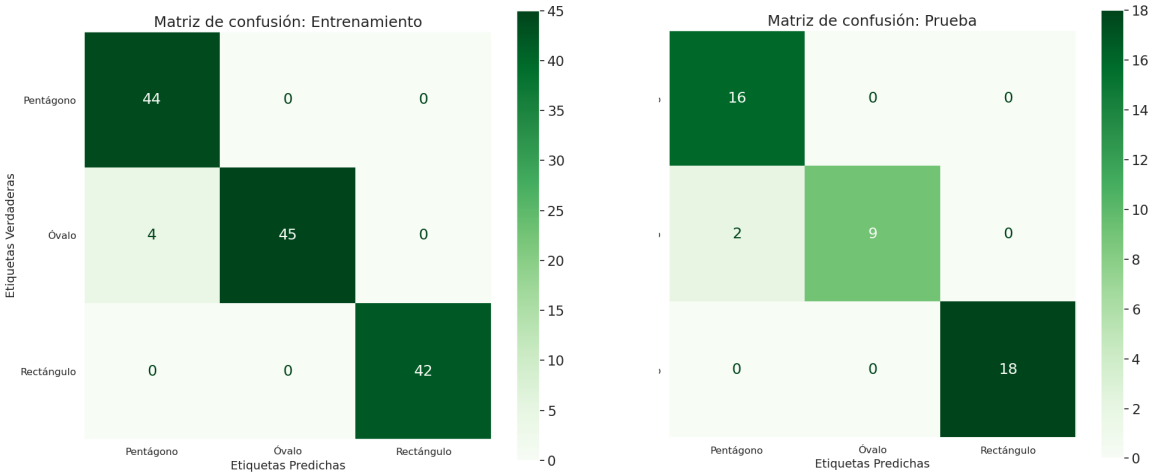


Figura 6.10: Matriz de confusión de la segunda red neuronal de estado eco con los momentos de Hu como características (dos clasificadores en total). Ambas matrices indican la clasificación incorrecta de algunas imágenes de las clases óvalo.

Clasificador con tres redes

Los parámetros de los tres modelos fueron optimizados con el algoritmo NSGA-II. Los parámetros del primer y segundo clasificador fueron los descritos anteriormente.

El tercer modelo tiene como objetivo clasificar las dos clases restantes: Pentágono y Óvalo. Con los siguientes parámetros se logró una exactitud igual a 1.0 (véase figura 6.11).

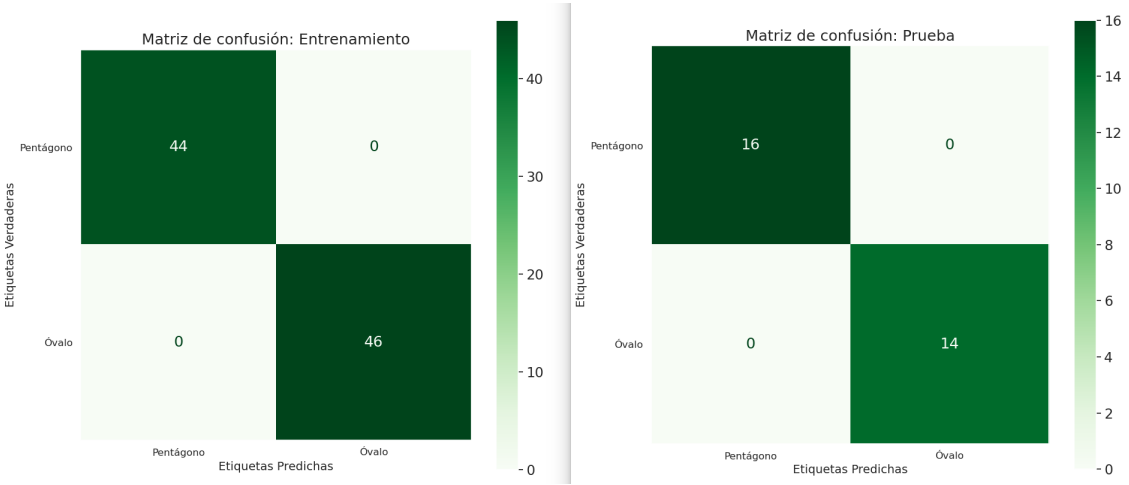


Figura 6.11: Matriz de confusión de la primera red neuronal de estado eco con los momentos de Hu como características (tres clasificadores en total). Las matrices indican la correcta clasificación de todas las imágenes.

Por lo que concatenando tres modelos de redes neuronales de estado eco fue posible la correcta clasificación de todas las imágenes de nuestra base de datos de figuras, como se muestra en la matriz de confusión de la figura 6.12.

Figura 6.12: Matriz de confusión de los tres clasificadores con redes neuronal de estado eco con los momentos de Hu como características.

En la tabla 6.1 se muestran los parámetros que describen la red neuronal de estado eco de cada clasificador y su valor de exactitud.

Tabla 6.1: Valores de los parámetros de los clasificadores obtenidos con las redes neuronales de estado eco.

	Primero (6 clases)	**Segundo (3 clases)**	**Tercero (2 clases)**
Ruido	0.28240013587590496	0.8348566825734045	0.5255812097067702
Filtración	0.9199889162622901	0.8092210958789994	0.2392000751906882
Contenedor	36	50	8
SemillaW	73	73	94
SemillaP	78	83	25
Exactitud	1.0	1.0	1.0
Tiempo de entrenamiento	66.6 ms	54.3 ms	32.9 ms

6.2. HOG

6.2.1. Bayes ingenuo

Se entrenó un modelo de Bayes Ingenuo con las características extraídas y se evaluó su exactitud mediante validación cruzada con cinco pliegues en la clasificación de nuestra base de datos. Además, se aplicó el algoritmo de ACP para reducir la dimensionalidad a 128, 64, 32 y 16, y se empleó la misma metodología de evaluación para cada caso.

A continuación se muestra los resultados en las tablas 6.2 y 6.3, el valor de la exactitud y una letra B o M y entre paréntesis las iniciales de las únicas clases que fueron bien o mal clasificadas, respectivamente; sino se especifica significa que clasificó correctamente todas las imágenes.

Tabla 6.2: Exactitud del algoritmo Bayes Ingenuo en la clasificación de nuestra base de datos de imágenes utilizando las características HOG sin y con reducción a 128 y 64 dimensiones mediante ACP.

Car. HOG	**Sin ACP**	**ACP(128)**	**ACP(64)**
288	0.9 B(C,E)	0.9895 M(O,T)	0.9875 M(O,T)
10368	0.9166 B(C,E)	0.9875	0.9895
41472	0.9520 B(C,E,L)	0.9645 B(C,E,I,L)	0.9520 M(R,O,T)
2592	0.8562 B(C,E)	0.9979 M(O,R)	0.9937 M(O,R)
1568	0.8833 B(C,E,I,L)	0.9916	0.9916 M(O,R)
6272	0.9375 B(C,E)	0.977 M(O,R,T)	0.9812 M(O,R,T)
392	0.8625 B(C)	0.9875	0.9916 M(R,O)
12544	0.9395 B(C,E)	0.9666 B(C,E,L,P)	0.9604 M(O,R,T)
21632	0.9395 B(C,E)	0.975 M(O,T,R)	0.9729 M(O,T,R)
5408	0.87 B(C,E,L)	0.9979 M(O,T)	0.9937 M(O,I,T)
6240	0.8708 B(C,E,I)	0.9895 M(O,I,R)	0.9895
5120	0.902 B(C,E)	0.9416 B(C,E,P)	0.9375 B(C,E,P)
2560	0.85 B(E)	0.9625 M(O,I,R)	0.9645 B(C,E,L,P)
2250	0.8645 B(C,E,I)	0.9875 B(O,I)	0.9916
1800	0.8645 B(C,E,I)	0.9895 M(O,T)	0.9937 M(O,R,T)
1125	0.8687 B(C,E,I)	0.9791 M(O,R)	0.9812 M(O,R)
3600	0.85 B(C,E)	0.9895 M(O,R,T)	0.9937 M(O,I,T)

Tabla 6.3: Exactitud del algoritmo Bayes Ingenuo en la clasificación de nuestra base de datos de imágenes utilizando las características HOG con reducción a 32 y 16 dimensiones mediante ACP.

Car. HOG	**ACP(32)**	**ACP(16)**
288	0.9937 M(O,I,T)	0.975 B(E,P,R)
10368	0.9875 M(T,I)	0.9687 M(O,T,I)
41472	0.95 B(O,R,T)	0.925 B(C,E,I,L)
2592	0.9875 M(O,I,R)	0.9770 M(O,I,R)
1568	0.9854 M(I,O,T,R)	0.9625 M(I,O,T,R)
6272	0.9729 M(O,I,T)	0.9541 B(E,P)
392	0.9791 M(O,I,R,T)	0.9812 M(O,I,R,T)
12544	0.9645 B(C,E,L,P)	0.9020 B(C,E,P)
21632	0.987 B(C,E,P)	0.9312 B(E)
5408	0.9708 M(I,T,O,R)	0.9625 B(C,E,P)
6240	0.975 M(O,I,T)	0.9708 B(E,P)
5120	0.9125 B(E,P)	0.877 B(P)
2560	0.9520 B(C,E,P)	0.9229 B(C,E,P)
2250	0.9833 M(O,I,T)	0.9708 M(T,I,O,R)
1800	0.9875 M(O,I,R)	0.9708 M(O,R,T)
1125	0.9854 M(O,R,T)	0.9770 M(O,R,I)
3600	0.9895 B(C,E,P)	0.975 M(T,I,O,R)

Con base en los resultados mostrados en la tabla anterior, se optó por aplicar el algoritmo ACP para reducir la dimensionalidad de las características a 64. Esta elección permite conservar una buena descripción de las imágenes con un menor número de características. Al clasificar estos datos con el algoritmo de Bayes Ingenuo, se logran exactitudes de 0.99 mediante validación cruzada de cinco dobleces.

6.2.2. Redes neuronales directas

Para clasificar nuestra base de datos con una red neuronal, se seleccionaron los archivos con el número de características reducidas a 64 que presentaron una exactitud igual o mayor a 0.9895 con validación cruzada, estos se listan a continuación:

- 392
- 1568
- 1800
- 2250

- 2592
- 3600
- 5408
- 6240
- 10368

Se realizó una búsqueda iterativa de la arquitectura de la red neuronal, variando el número de neuronas por capa entre 8 y 21, y el número de capas ocultas entre 1 y 3; se seleccionaron estos rango con el propósito de mantener simple la arquitectura de la red. Se entrenó y evaluó la red neuronal con validación cruzada de cinco dobleces, buscando el ajuste de parámetros que maximizara la exactitud en la clasificación de la base de datos, idealmente cercana a 1. En la tabla 6.4 se muestran los resultados de esta prueba.

Tabla 6.4: Parámetros de la red neuronal para clasificar la base de datos con características HOG (ACP 64).

Núm. de Carac. HOG	**Neuronas por capa oculta**	**Exactitud**
392	16,18,17	0.9813
1568	20,17,12	0.9771
1800	17,10,14	0.9896
2250	18,19	0.9646
2592	20	0.9875
3600	14	0.9646
5408	19,16	0.9792
6240	17,20	0.9729
10368	19,18	0.9813

La tabla 6.4 muestra que ninguna de las combinaciones de parámetros evaluadas logra clasificar correctamente todas las imágenes de la base de datos. Por esta razón, se requiere el uso de una red neuronal adicional. Inicialmente, se pensó en elegir la combinación de parámetros que tuviera la mayor exactitud, que según la tabla sería la red neuronal con una exactitud de 0.9896, compuesta por tres capas ocultas con 17, 10 y 14 neuronas cada una. No obstante, se optó por la segunda combinación de parámetros con mayor exactitud, 0.9875, que tiene una sola capa oculta con 20 neuronas, ya que este modelo es más simple y ligero que el anterior, en la figura 6.13 se muestra la matriz de confusión de este modelo.

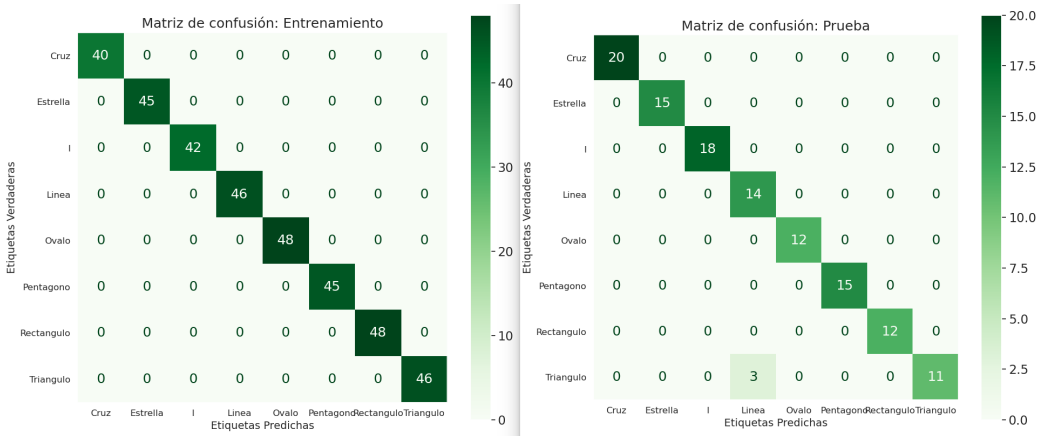


Figura 6.13: Matrices de confusión de la red neuronal con 20 neuronas en su única capa oculta. Se observa en la matriz del conjunto de entrenamiento una correcta clasificación de todas las imágenes. En la matriz del conjunto de prueba se muestra la incorrectas clasificación de únicamente tres imágenes de la clase triángulo.

Las únicas figuras mal clasificadas fueron Línea y Triángulo, por lo que se realizó una búsqueda de los parámetros de cantidad de neuronas en cada capa oculta y la cantidad de capas ocultas de una red neuronal que clasifique correctamente ambas clases, es decir, una exactitud en validación cruzada con cinco dobleces igual a uno. Esto se logró con una red neuronal con tres capas ocultas, 8, 8 y 12 neuronas por capa, respectivamente.

A continuación en la figura 6.14 se muestra la matriz de confusión de ambas redes neuronales concatenadas, donde se muestra una correcta clasificación de nuestra base de datos. El tiempo de entrenamiento fue de 30.1 ms para el primer modelo con una capa oculta y 23.3 ms para el segundo modelo con tres capas ocultas.

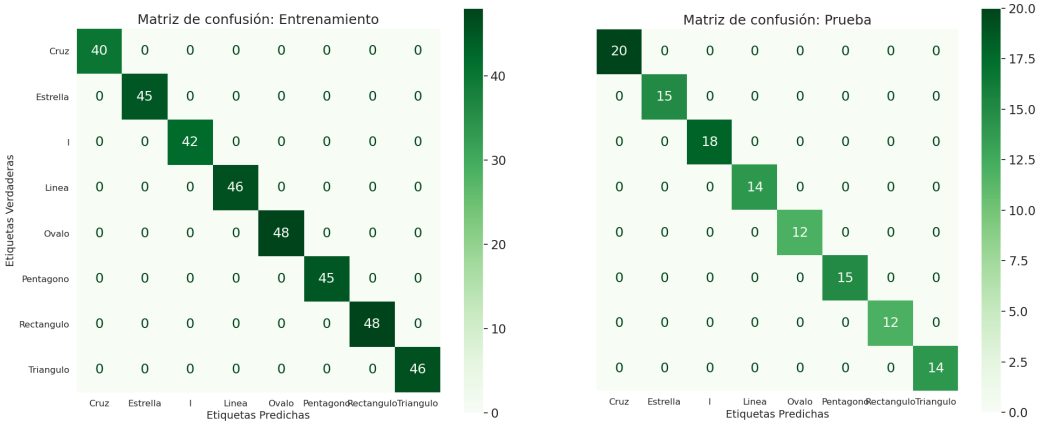


Figura 6.14: Matrices de confusión del modelo con un par de redes neuronales concatenadas. Ambas matrices muestran valores distintos a cero únicamente en su diagonal principal, representando la correcta clasificación de todas las imágenes.

6.2.3. Máquina de Vectores de Soporte

Para clasificar las figuras de nuestra base de datos, se utilizó el algoritmo MVS SVC con las 2592 características extraídas mediante el algoritmo HOG y reducidas a 64 dimensiones mediante ACP. El modelo de MVS utilizado fue Uno contra el Resto (one-vs-rest, en inglés), el resto de parámetros se dejaron por defecto. El entrenamiento del modelo duró 20.4 ms. Se evaluó el rendimiento del modelo usando validación cruzada de cinco dobleces, y se obtuvo una exactitud de 0.9895. A continuación, se muestra la matriz de confusión del modelo en la figura 6.15.

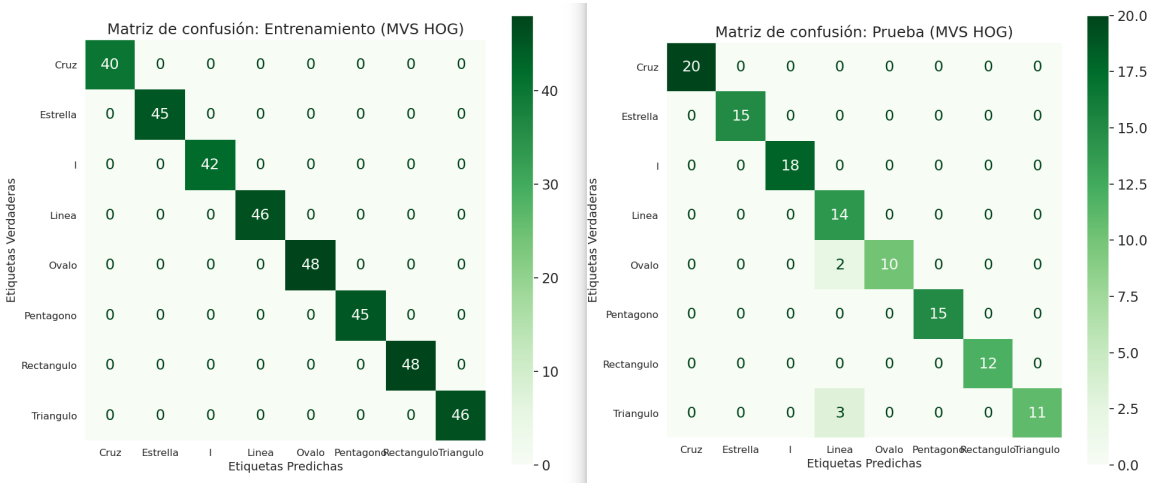


Figura 6.15: Matrices de confusión del modelo de MVS con características HOG. Para el conjunto de entrenamiento todas las imágenes fueron clasificadas correctamente. La matriz de confusión del conjunto del prueba muestra la incorrecta clasificación de algunas imágenes de las clases óvalo y triángulo etiquetándolas como pertenecientes a la clase línea.

6.3. Aumento de datos

6.3.1. Bayes Ingenuo (Hu)

Al modelo con el algoritmo Bayes Ingenuo entrenado anteriormente con la BDO, se le cargó la BDA y se evaluó mediante validación cruzada con cinco dobleces y se obtuvo una exactitud de 0.8792. La matriz de confusión para estos nuevos datos se muestra en la figura 6.16.

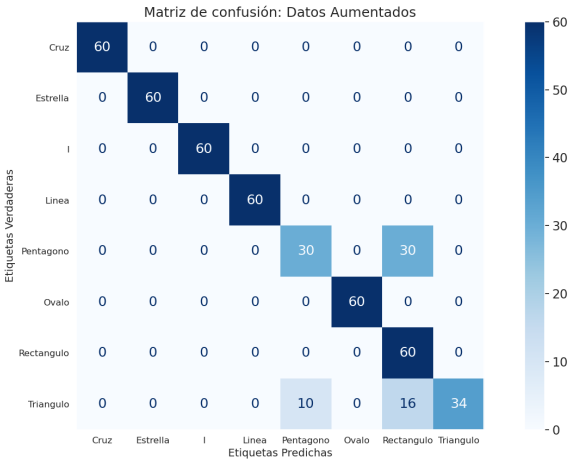


Figura 6.16: Matriz de confusión del modelo con Bayes Ingenuo para la BDA, se observa la incorrecta clasificación de imágenes de las clases pentágono etiquetándolas como rectángulo y de la clase triángulo como pentágono y rectángulo.

A la nueva base de datos también se le aplicó la extracción de características c_1, c_2 y relación de áreas (ver la sección 6.1, en la página 41), para observar el desempeño de la concatenación del clasificador con momentos de Hu y con las características c_1, c_2 y relación de área, los cuales fueron entrenados con la BDO. Estos modelos presentan una exactitud de 0.9563 en la clasificación de la BDA. En la figura 6.17 se muestra la matriz de confusión.

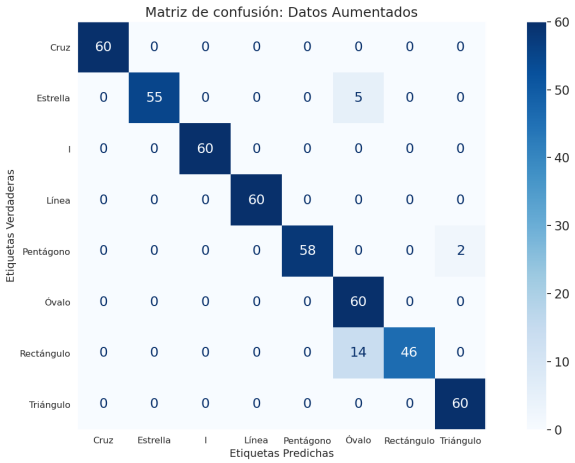


Figura 6.17: Matriz de confusión de los modelos de Bayes Ingenuo utilizando como características los momentos invariantes de Hu y las características c_1, c_2 y área. Se observa una clasificación errónea de imágenes de las clases estrella, pentágono y rectángulo.

6.3.2. Máquina de Vectores de Soporte (Hu)

Se evaluó el desempeño del algoritmo MVS, previamente entrenado con los momentos de Hu de la BDO, usando los momentos de Hu de una nueva base de datos. El resultado fue una clasificación correcta, con una exactitud igual a 1, como se puede observar en la matriz de confusión que se muestra en la figura 6.18.

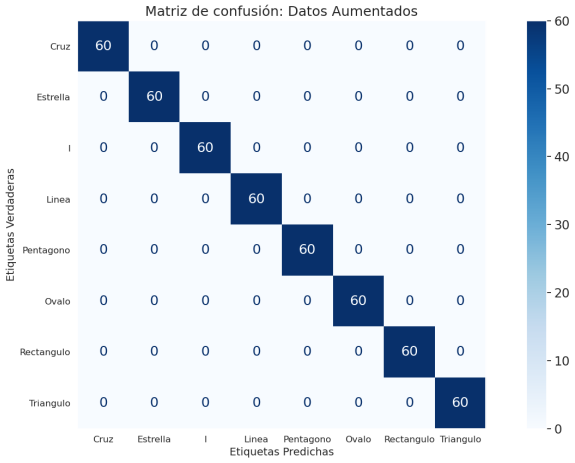


Figura 6.18: Matriz de confusión del modelo con MVS utilizando como características los momentos invariantes de Hu de la nueva base de datos.

6.3.3. Redes neuronales (Hu)

Para evaluar el desempeño del algoritmo de Redes Neuronales que se entrenó con los momentos de Hu de la BDO, se ingresaron los momentos de Hu de la nueva base de datos. El algoritmo obtuvo una exactitud en validación cruzada con cinco dobleces de 1. La matriz de confusión de la figura 6.19 muestra el desempeño del algoritmo para cada clase.

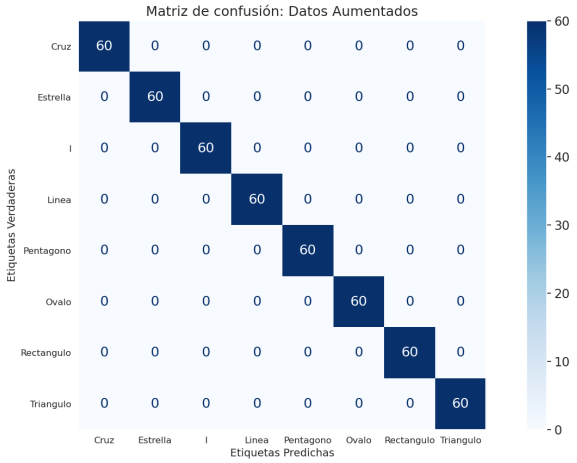


Figura 6.19: Matriz de confusión del modelo con redes neuronales utilizando como características los momentos invariantes de Hu de la BDA. La presencia de valores distintos de cero únicamente en la diagonal principal representa la exactitud igual a 1 del modelo.

6.3.4. Clasificador con redes neuronales recurrentes de estado eco (Hu)

Como anteriormente mostramos, un único modelo o la concatenación de dos modelos de RNEE no lograron proporcionar una clasificación correcta de la BDO, se evaluó directamente el desempeño de la concatenación de las tres redes de estado eco entrenadas anteriormente en la clasificación de la BDA. Esta red tuvo dificultades para clasificar siete muestras de la clase Estrella, asignándoles la clase Rectángulo. Por lo tanto, la exactitud de este modelo fue de 0.9854. La matriz de confusión se presenta a continuación en la figura 6.20.

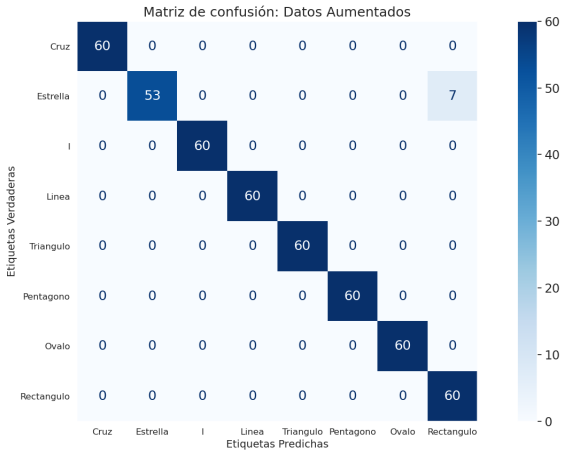


Figura 6.20: Matriz de confusión del modelo con tres redes neuronales de estado eco utilizando como características los momentos invariantes de Hu de la DBA. Se observa que siete imágenes de la clase estrella fueron etiquetados incorrectamente como pertenecientes a la clase rectángulo.

6.3.5. Bayes Ingenuo (HOG)

Se evaluó también el desempeño del modelo de Bayes Ingenuo, que se había preentrenado con las características extraídas por el descriptor HOG y luego reducidas a 64 dimensiones mediante el algoritmo de ACP. El clasificador obtuvo una exactitud de 0.1521 en esta prueba, como se puede observar en la siguiente matriz de confusión en la figura 6.21.

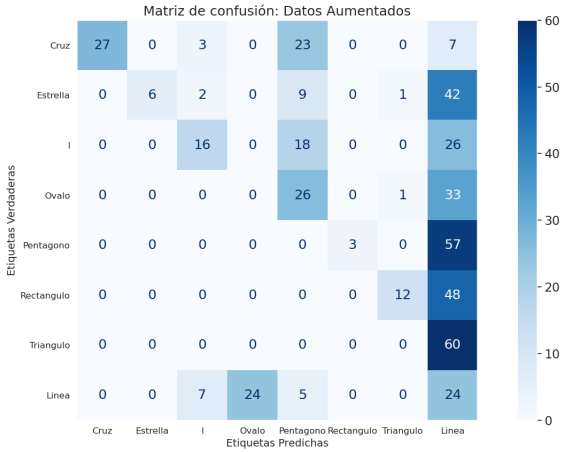


Figura 6.21: Matriz de confusión del modelo de Bayes Ingenuo utilizando las características HOG extraídas de la BDA con reducción a 64 dimensiones mediante ACP. La distribución de valores alejados de la diagonal principal indican la incapacidad del modelo de clasificar correctamente la mayoría de las imágenes.

6.3.6. Máquina de Vectores de Soporte (HOG)

Se probó también el desempeño del modelo pre-entrenado de MVS, con las características HOG reducidas a 64 dimensiones mediante el algoritmo ACP, usando la nueva BDA. El modelo logró una exactitud de 0.40 en la clasificación, como se muestra en la matriz de confusión que se presenta a continuación en la figura 6.22.

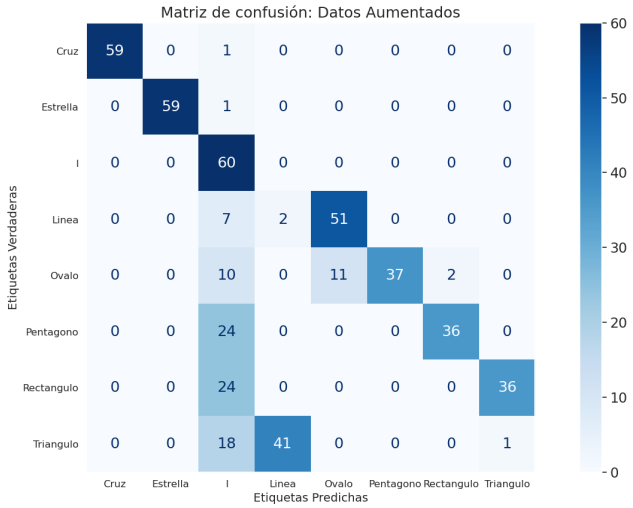


Figura 6.22: Matriz de confusión del modelo de MVS utilizando las características HOG extraídas de la BDA con reducción a 64 dimensiones mediante ACP. Debido a la distribución de valores distintos a cero fuera de la diagonal principal se observa una incorrecta clasificación para la mayoría de las imágenes.

6.3.7. Redes neuronales (HOG)

El desempeño del modelo con dos redes neuronales, que se habían entrenado previamente, se evaluó también con la BDA, para observar su capacidad de clasificación. El modelo alcanzó una exactitud de 0.3583, como se observa en la matriz de confusión que se adjunta a continuación en la figura 6.23.

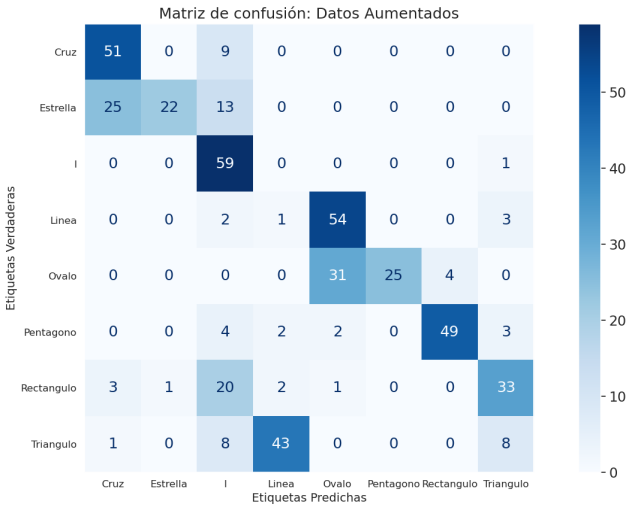


Figura 6.23: Matriz de confusión del modelo de dos redes neuronales utilizando las características HOG extraídas de la BDA con reducción a 64 dimensiones mediante ACP. Se observa una incorrecta clasificación para la mayoria de las imágenes.

En la clasificación de la BDA, los modelos mostrados que fueron entrenados previamente con la BDO tuvieron un desempeño deficiente. Por lo tanto, para lograr una clasificación correcta, necesitan entrenarse de nuevo con los nuevos datos. Esto revela la incapacidad de generalización del extractor de características HOG cuando las imágenes están trasladadas, rotadas y escaladas.

6.4. Sumario

En el campo del aprendizaje automático tradicional, los únicos algoritmos que lograron clasificar correctamente nuestra base de datos de imágenes, utilizando un solo clasificador y como características los momentos de Hu, fueron la MVS y la red neuronal directa. El entrenamiento de estos modelos fue de 15 μs y 0.7 ms, respectivamente. Sin embargo, cuando se extrajeron 2592 características HOG y se redujeron a 64 dimensiones mediante el ACP, la mayor exactitud que se alcanzó al implementar un único clasificador fue 0.9895 con la MVS y un tiempo de entrenamiento de 20.4 ms.

Capítulo 7

Desempeño de los modelos de aprendizaje profundo

Inicialmente en nuestro proceso de experimentación, se creó un conjunto de datos de imágenes personalizadas, denominado CustomImageDataset, para el reentrenamiento de las RNC de Pytorch. Se organizó la base de datos en dos directorios principales: uno para las imágenes de entrenamiento y otro para las de prueba. El directorio de entrenamiento incluye 45 imágenes por clase, sumando un total de 360 imágenes, mientras que el directorio de prueba contiene 15 imágenes por clase, alcanzando un total de 120 imágenes.

7.1. Sobreajuste y aumento de datos

Utilizando el CustomImageDataset, la primera RNC reentrenada fue nuestra propuesta basada en la arquitectura ImageNet_5m_m2. Se empleó la función de pérdida utilizada fue CrossEntropyLoss y el optimizador Adadelta, configurado con una tasa de aprendizaje de 0.01, un valor de epsilon igual a 10^{-6} y un peso de decaimiento de cero. Inicialmente se seleccionó la CPU del ordenador para entrenar la RNC, sin embargo, se presentó un error computacional debido las limitaciones del hardware en el cálculo de la pérdida durante el entrenamiento originando un Not a number (NaN), este error fue solucionado utilizando la UPG para entrenar la red. El resto de redes tambien fueron entrenadas con la UPG. El proceso de entrenamiento para la modificación de la RNC ImageNet_5m_m2 se definió con 200 épocas. En la figura 7.1, se presentan las gráficas de la exactitud y el error durante el entrenamiento de la RNC. Se observa un sobreajuste significativo, caracterizado por una alta exactitud en la clasificación del conjunto de entrenamiento y un estancamiento en la exactitud del conjunto de prueba, lo cual indica que no hay mejoras significativas a medida que aumenta el número de épocas.

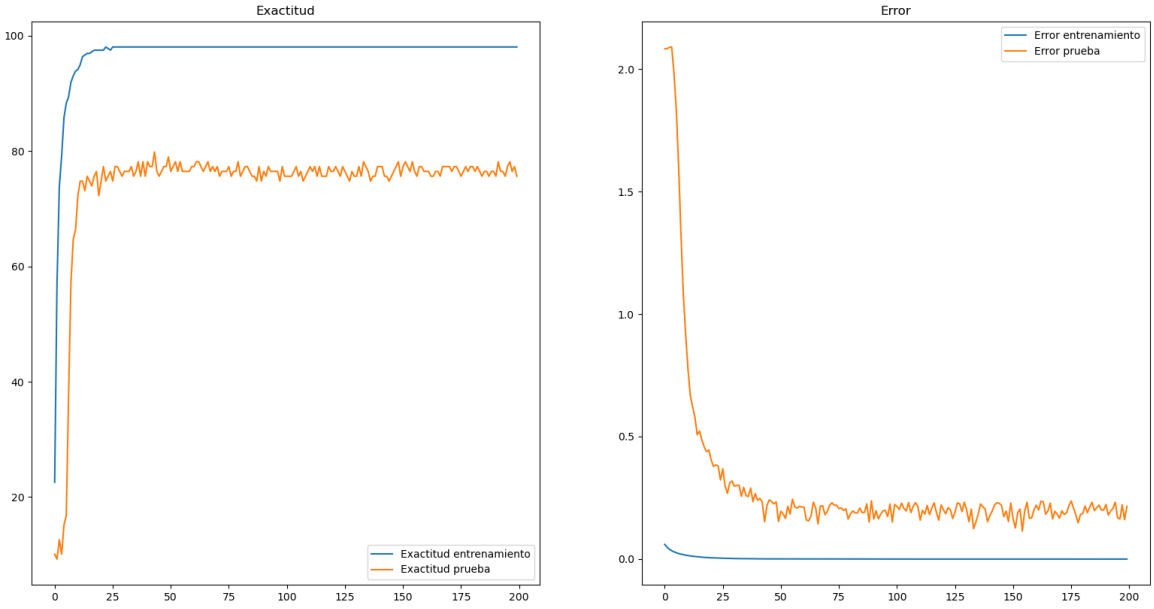


Figura 7.1: Gráficas de exactitud y error durante el reentrenamiento de la SimpleNet con nuestra base de datos de imágenes. A la izq. la gráfica azul representa la exactitud para el conjunto de entrenamiento y la naranja para el conjunto de prueba, es evidente un sobreajuste del modelo. A la der. se muestra la grafica de error del modelo, en azul para el conjunto de entrenamiento y en naranja para el conjunto de prueba.

Para abordar el problema de sobreajuste observado, se decidió ampliar el conjunto de datos. Esto se logró aplicando operaciones de escalado, traslación y rotación, técnicas ya utilizadas previamente, lo que resultó en triplicar el volumen de nuestra base de datos hasta alcanzar un total de 1440 imágenes. De estas, 1080 imágenes se asignaron al conjunto de entrenamiento y 360 imágenes al conjunto de prueba. Adicionalmente, se experimentó con la inclusión de una capa oculta en la red neuronal completamente conectada, probando con 20 y 32 neuronas. Se aumentó el número de épocas de entrenamiento a 50, manteniendo constantes los demás parámetros. Las figuras 7.2 y 7.3 muestran las gráficas de precisión y pérdida de la RNC correspondientes a este experimento, con las configuraciones de 20 y 32 neuronas en la capa oculta, respectivamente.

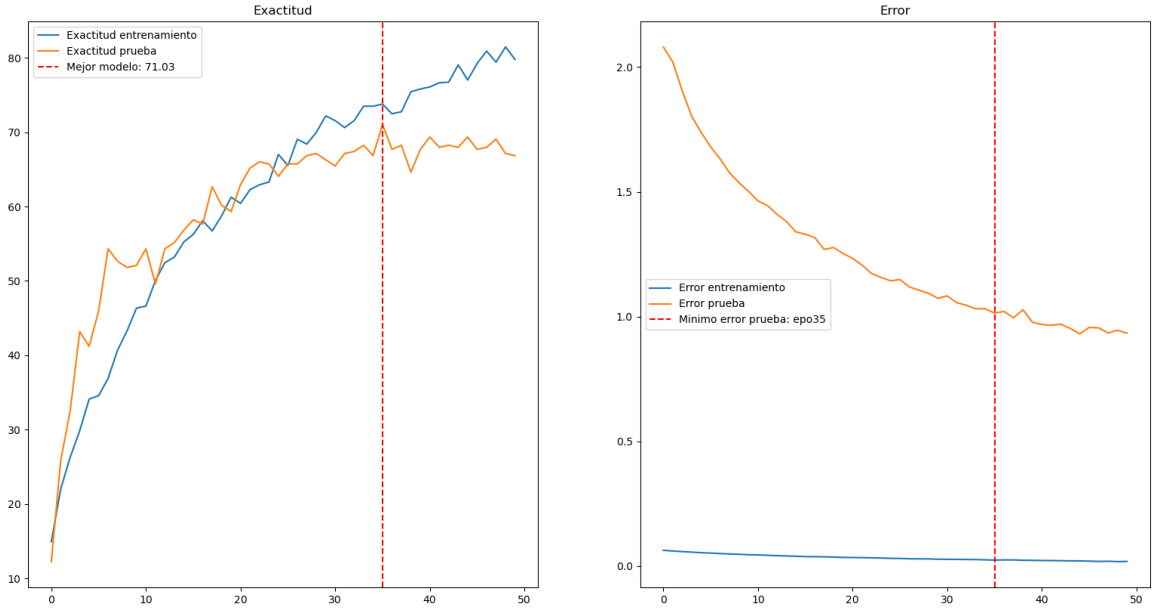


Figura 7.2: A la izq. las gráficas de exactitud de entrenamiento (azul) y prueba (naranja) y a la der. las gráficas de error de entrenamiento (azul) y prueba (naranja) durante el reentrenamiento de la SimpleNet con una capa oculta de 20 neuronas. La línea punteada vertical representa la obtención del mejor modelo para esa prueba.

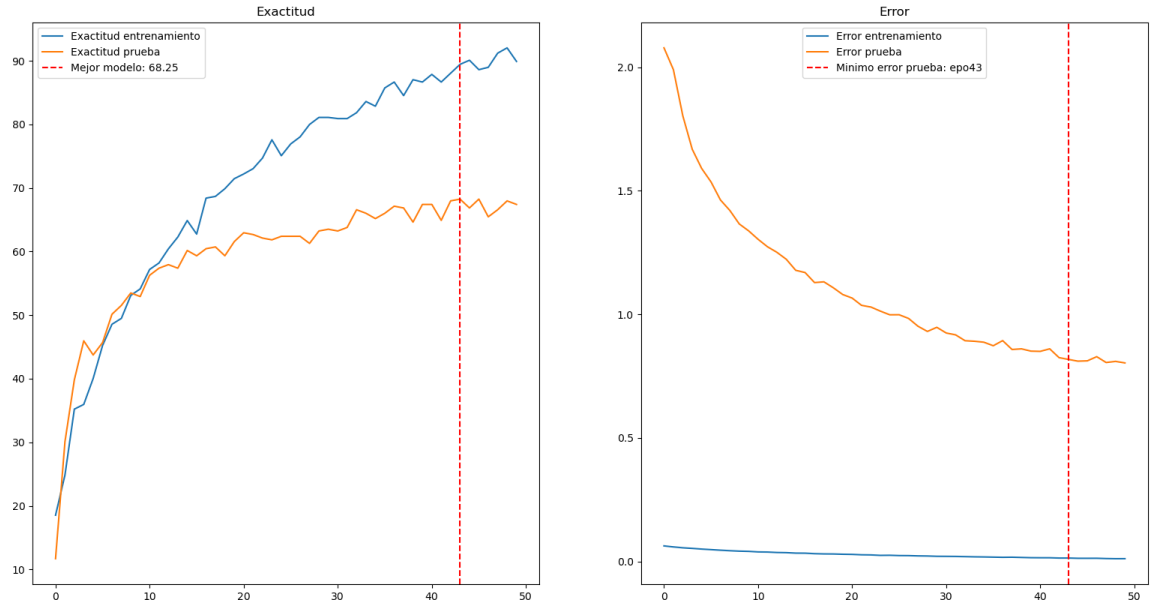
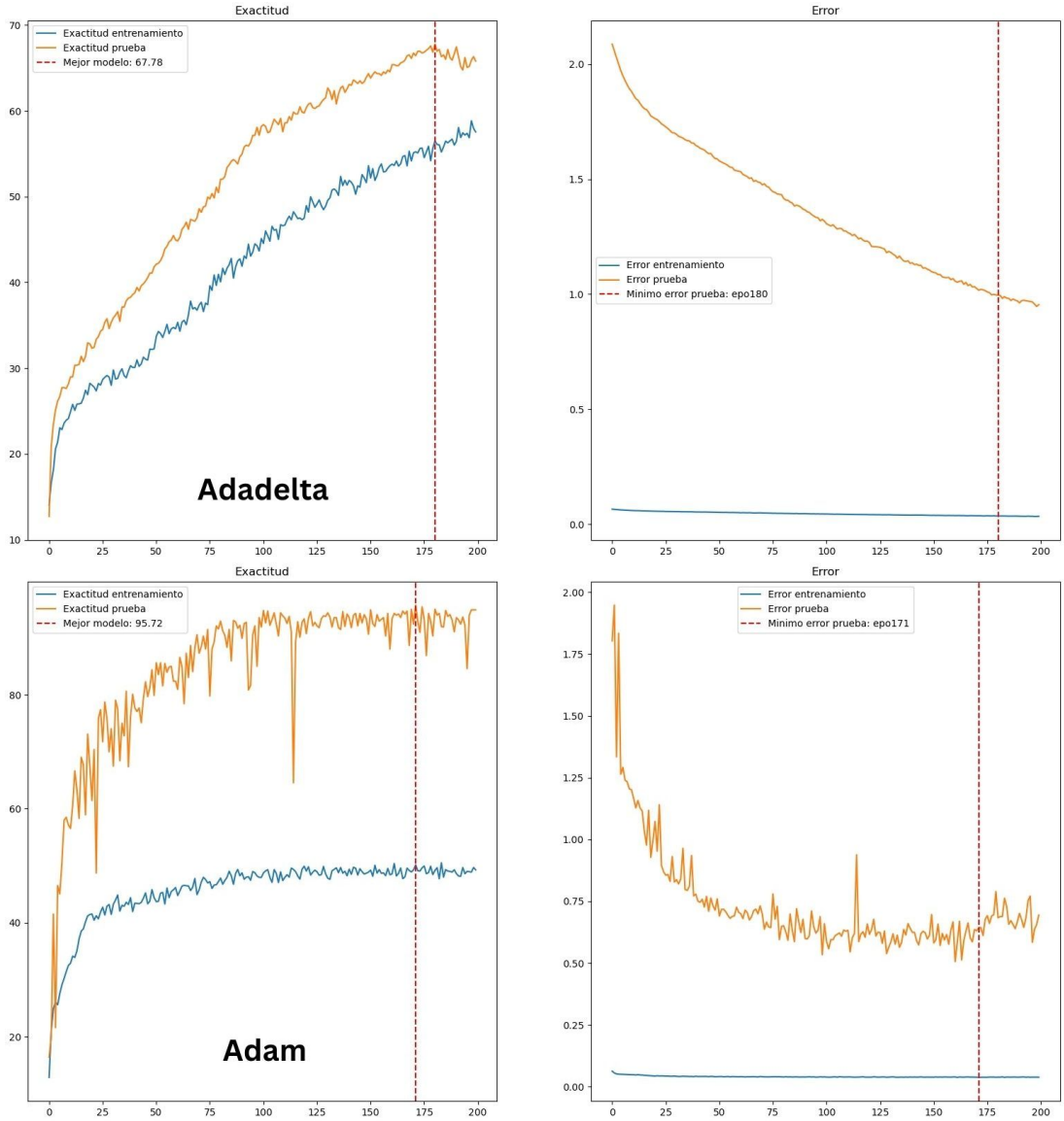


Figura 7.3: A la izq. las gráficas de exactitud de entrenamiento (azul) y prueba (naranja) y a la der. las gráficas de error de entrenamiento (azul) y prueba (naranja) durante el reentrenamiento de la SimpleNet con una capa oculta de 32 neuronas. La línea punteada vertical representa la obtención del mejor modelo para esa prueba.

7.2. Experimentación con los optimizadores

En la siguiente etapa de experimentación, se centró la atención en evaluar el desempeño del optimizador, con el objetivo de determinar si había diferencias notables en su desempeño durante el entrenamiento de la RNC para la clasificación de nuestra base de datos. Se seleccionaron los optimizadores Adadelta, Adam y Descenso de Gradiente Estocástico (DGE), todos configurados con una tasa de aprendizaje de 0.001. En las figuras 7.4 se ilustra el rendimiento de los optimizadores Adadelta, Adam y DGE, respectivamente.



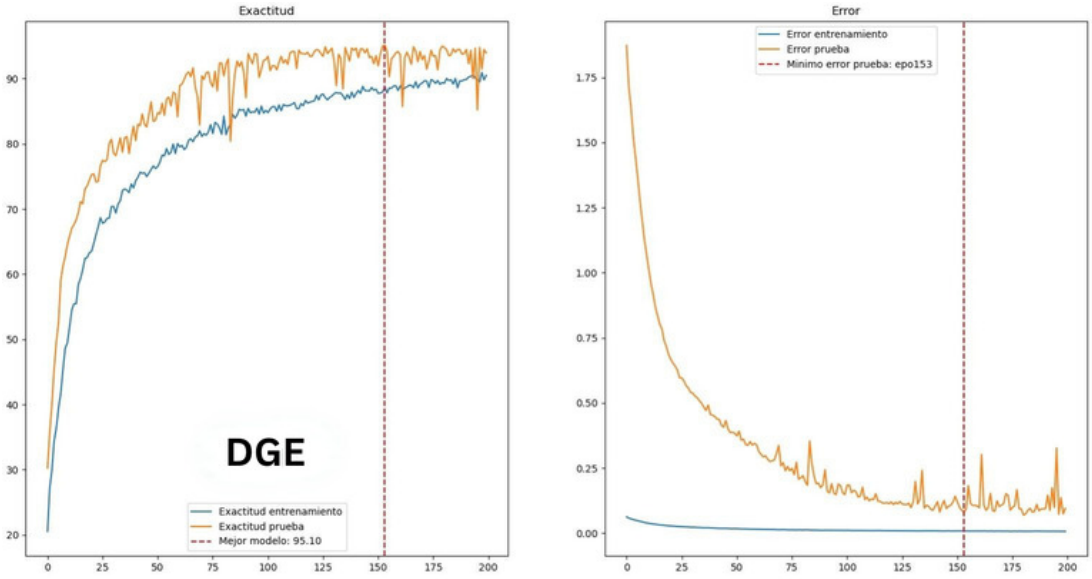


Figura 7.4: En la columna izq. las gráficas de exactitud de entrenamiento (azul) y prueba (naranja) y en la columna der. las gráficas de error de entrenamiento (azul) y prueba (naranja) durante el reentrenamiento de la SimpleNet con los optimizadores Adadelta, Adam y DGE, respectivamente. La línea punteada vertical representa la obtención del mejor modelo para esa prueba.

Los resultados experimentales indicaron que el optimizador DGE sobresalió en términos de rendimiento, logrando una precisión superior al 80 % en ambos conjuntos de imágenes. Basándonos en esta evidencia, se tomó la decisión de emplear el optimizador DGE para los experimentos futuros, con la expectativa de mantener o mejorar este nivel de exactitud.

7.3. Experimentación con 96,000 imágenes

Ante la continua presencia de sobreajuste, se decidió implementar un incremento significativo en el volumen de datos, rotando, escalando y trasladando aleatoriamente las imágenes. Tomando como referencia el artículo [29], que cataloga a la base de datos MNIST como una base de datos grande, la cual está conformada por 70,000 imágenes, optamos por expandir nuestra BDO en un factor de 200. Como resultado, la nueva base de datos consta de 96,000 imágenes: 72,000 destinadas al conjunto de entrenamiento y 24,000 al conjunto de prueba.

Con esta base de datos ampliada, se procedió a reentrenar nuestra versión modificada de la RNC SimpleNet. Utilizamos el optimizador DGE con una tasa de aprendizaje de 0.1. La RNC alcanza una exactitud de 97.56 % para el conjunto de entrenamiento y del 97.37 % en el conjunto de prueba, resultado superior en comparación con el 95.10 % obtenido en el experimento anterior. Además, se logró una reducción significativa en el número de épocas necesarias para alcanzar la máxima exactitud durante el entrenamiento, disminuyendo de 153 a 36 épocas, las gráficas que muestran la exactitud y el error durante el entrenamiento se presentan en la figura 7.5 y la matriz de confusión correspondiente en la figura 7.6. Con base en estos hallazgos, se decidió emplear la base de datos de 96,000 imágenes para la experimentación en las demás arquitecturas de nuestras RNC.

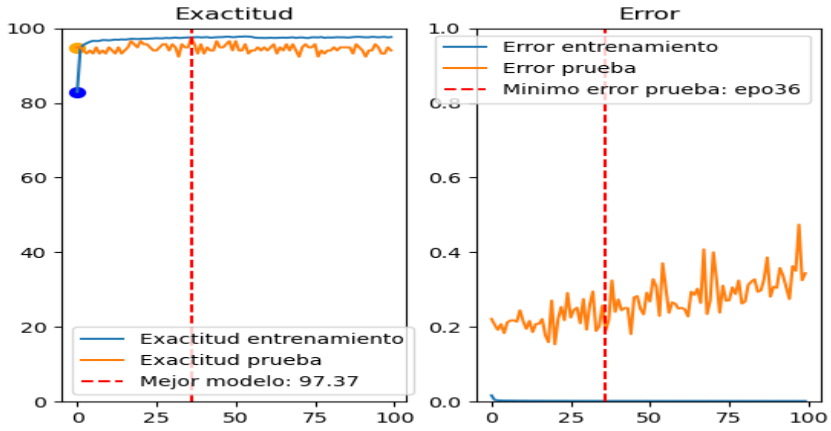


Figura 7.5: A la izq. las gráficas de exactitud de entrenamiento (azul) y prueba (naranja) y a la der. el error de entrenamiento (azul) y prueba (naranja) durante el reentrenamiento de nuestra modificación de la RNC SimpleNet y nuestra base de datos con 96,000 imágenes. La línea punteada vertical representa la obtención del mejor modelo para esa prueba, el cual muestra valores de exactitud sin sobreajuste.

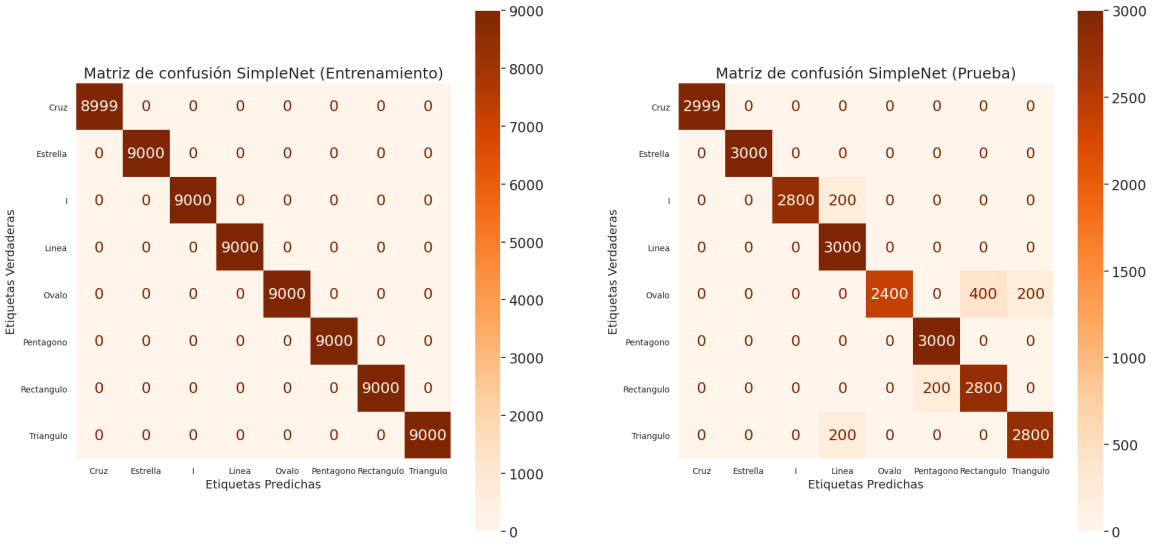


Figura 7.6: Matrices de confusión de nuestra modificación de nuestra modificación de la RNC SimpleNet_5m_m2 para la clasificación de la base de datos de 96,000 imágenes. La matriz de la izq. muestra una correcta clasificación de todas las imágenes del conjunto de entrenamiento, en la matriz de la der. se observa que solo las imágenes del conjunto de prueba de las clases cruz, estrella, línea y pentágono fueron clasificadas correctamente por este modelo.

7.4. DenseNet121

Utilizando la base de datos de 96,000 imágenes, entrenamos nuestra versión modificada de la RNC DenseNet121 con el optimizador DGE y una tasa de aprendizaje de 0.1 a lo largo de 100 épocas. El proceso de entrenamiento tomó 5 horas, 8 minutos y 30 segundos. La mejor exactitud se logró ya en la primera época, alcanzando un 99.87 % en el conjunto de prueba y un 100 % en el conjunto de entrenamiento. Las gráficas de exactitud y error durante el entrenamiento se pueden consultar en la figura 7.7, mientras que la matriz de confusión del modelo se presenta en la figura 7.8.

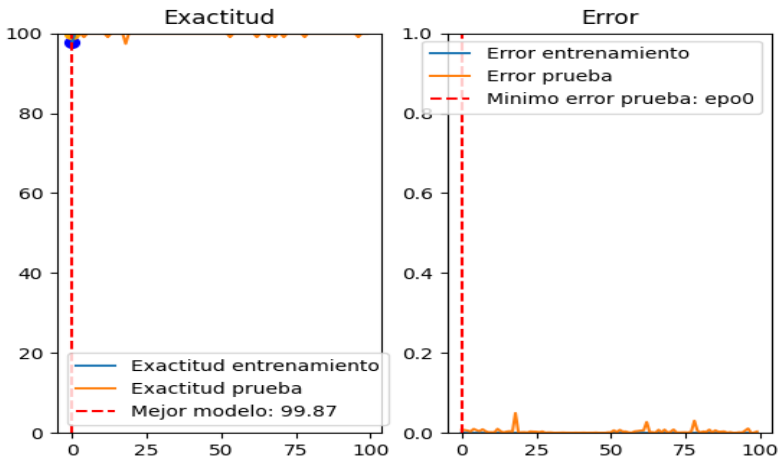


Figura 7.7: A la izq. las gráficas de exactitud de entrenamiento (azul) y prueba (naranja) y a la der. el error de entrenamiento (azul) y prueba (naranja) durante el reentrenamiento de nuestra modificación de la RNC DenseNet121 y nuestra base de datos con 96,000 imágenes. La línea punteada vertical representa la obtención del mejor modelo para esa prueba, el cual muestra valores de exactitud sin sobreajuste.

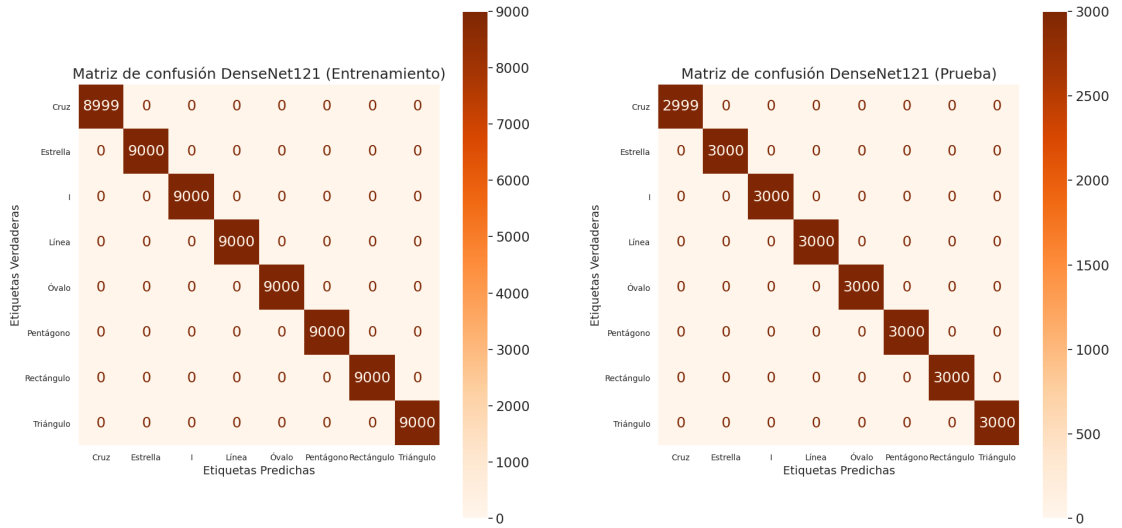


Figura 7.8: Matrices de confusión de nuestra modificación de nuestra modificación de la RNC DenseNet121 para la clasificación de la base de datos de 96,000 imágenes. Ambas matrices muestran una correcta clasificación de las imágenes por este modelo.

7.5. EfficientNet_B0

Siguiendo los parámetros previamente establecidos, se reentrenó nuestra modificación de la RNC EfficientNet_B0 durante 100 epocas. El proceso de entrenamiento tomó 3 horas, 44 minutos y 20 segundos. El modelo más eficaz se obtuvo en la primera época, logrando una exactitud del 99.87 % en el conjunto de prueba y un perfecto 100 % en el conjunto de entrenamiento. Las gráficas que detallan la exactitud y el error a lo largo del proceso de entrenamiento se ilustran en la Figura 7.9, y la matriz de confusión del modelo se expone en la Figura 7.10.

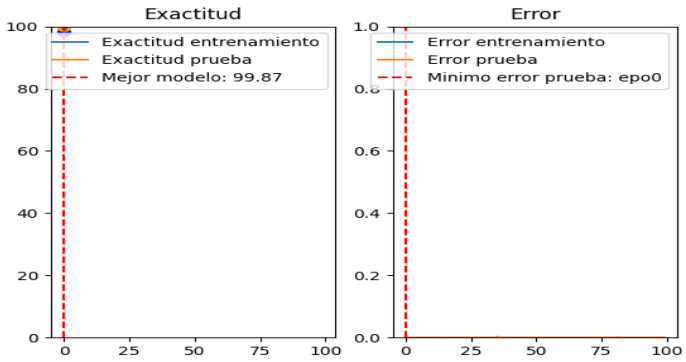


Figura 7.9: A la izq. las gráficas de exactitud de entrenamiento (azul) y prueba (naranja) y a la der. el error de entrenamiento (azul) y prueba (naranja) durante el reentrenamiento de nuestra modificación de la RNC EfficientNet_B0 y nuestra base de datos con 96,000 imágenes. La línea punteada vertical representa la obtención del mejor modelo para esa prueba, el cual muestra valores de exactitud sin sobreajuste.

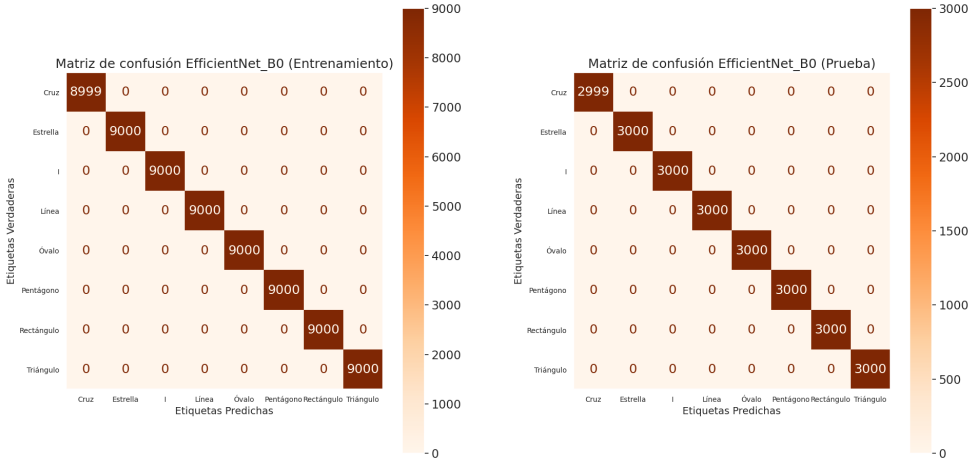


Figura 7.10: Matrices de confusión de nuestra modificación de nuestra modificación de la RNC EfficientNet_B0 para la clasificación de la base de datos de 96,000 imágenes. Ambas matrices muestran una correcta clasificación de las imágenes por este modelo.

7.6. GoogLeNet

Se reentrenó nuestra modificación de GoogLeNet durante 100 epocas. El proceso de entrenamiento tomó 2 horas, 39 minutos y 3 segundos. También se obtuvo el mejor modelo en la primera época, logrando un 99.87 % de exactitud en el conjunto de entrenamiento y un 100 % en el conjunto de entrenamiento. Las gráficas que exhiben la exactitud y el error durante el entrenamiento se pueden apreciar en la figura 7.11, y la matriz de confusión está detallada en la figura 7.12.

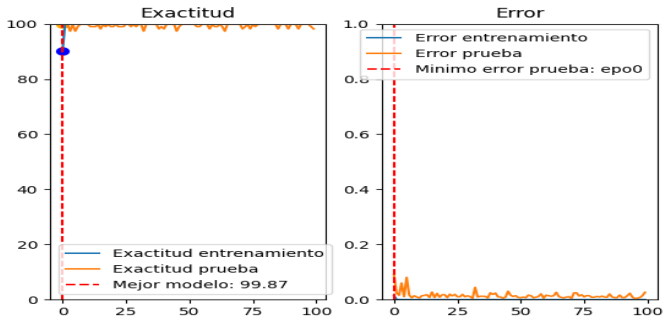


Figura 7.11: A la izq. las gráficas de exactitud de entrenamiento (azul) y prueba (naranja) y a la der. el error de entrenamiento (azul) y prueba (naranja) durante el reentrenamiento de nuestra modificación de la RNC GoogLeNet y nuestra base de datos con 96,000 imágenes. La línea punteada vertical representa la obtención del mejor modelo para esa prueba, el cual muestra valores de exactitud sin sobreajuste.

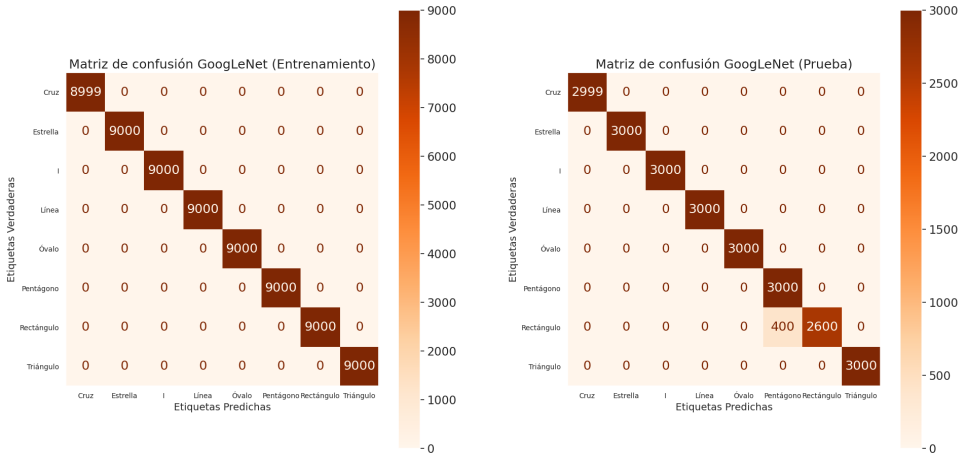


Figura 7.12: Matrices de confusión de nuestra modificación de nuestra modificación de la RNC GoogLeNet para la clasificación de la base de datos de 96,000 imágenes. La matriz de la izq. muestra una correcta clasificación de todas las imágenes del conjunto de entrenamiento, en la matriz de la der. se observa que solo las imágenes del conjunto de prueba de la clase rectángulo fueron mal clasificada por este modelo.

7.7. MNASNet0_5

En la continuación de nuestro proyecto, se reentrenó nuestra modificación de la RNC MNASNet0_5 durante 100 épocas. El proceso de entrenamiento tomó 4 horas, 43 minutos y 30 segundos. El mejor modelo que obtuvimos se alcanzó en la época 31, donde el modelo reporta una exactitud de 99.87 % para el conjunto de prueba y un 100 % para el conjunto de entrenamiento. Las gráficas de exactitud y error durante el entrenamiento se pueden observar en la figura 7.13, y la matriz de confusión está exhibida en la figura 7.14.

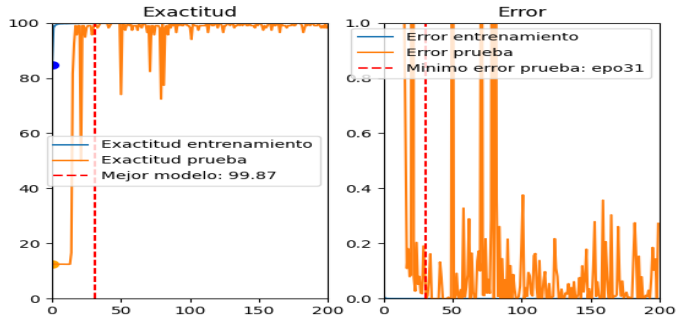


Figura 7.13: A la izq. las gráficas de exactitud de entrenamiento (azul) y prueba (naranja) y a la der. el error de entrenamiento (azul) y prueba (naranja) durante el reentrenamiento de nuestra modificación de la RNC MNASNet0_5 y nuestra base de datos con 96,000 imágenes. La línea punteada vertical representa la obtención del mejor modelo para esa prueba, el cual muestra valores de exactitud sin sobreajuste.

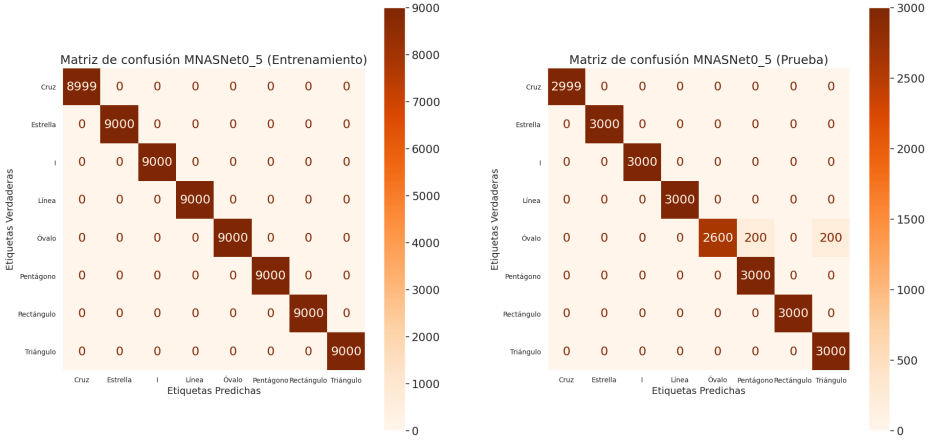


Figura 7.14: Matrices de confusión de nuestra modificación de nuestra modificación de la RNC MNASNet0_5 para la clasificación de la base de datos de 96,000 imágenes. La matriz de la izq. muestra una correcta clasificación de todas las imágenes del conjunto de entrenamiento, en la matriz de la der. se observa que solo las imágenes del conjunto de prueba de la clase óvalo fueron mal clasificada por este modelo.

7.8. MobileNet_V3_Large (V2)

Como antepenúltimo modelo se reentrenó nuestra modificación de la RNC MobileNet_V3_Large (V2) durante 50 épocas. El proceso de entrenamiento tomó 1 hora, 25 minutos y 29 segundos. El mejor modelo se alcanzó en la segunda época de entrenamiento, reportando una exactitud de 99.87 % para el conjunto de prueba y un 100 % para el conjunto de entrenamiento. En la figura 7.15 se muestran las gráficas de exactitud y error durante el entrenamiento de la red y en la figura 7.16 se muestra la matriz de confusión.

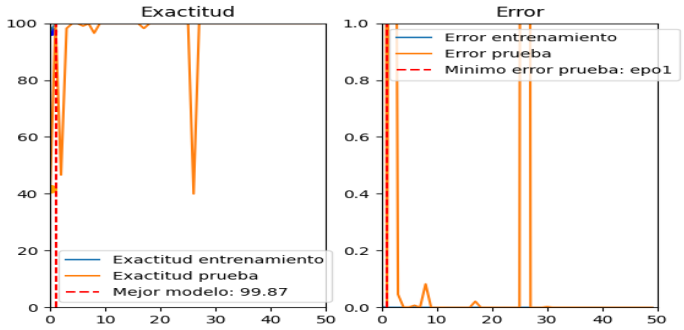


Figura 7.15: A la izq. las gráficas de exactitud de entrenamiento (azul) y prueba (naranja) y a la der. el error de entrenamiento (azul) y prueba (naranja) durante el reentrenamiento de nuestra modificación de la RNC MobileNet_V3_Large (V2) y nuestra base de datos con 96,000 imágenes. La línea punteada vertical representa la obtención del mejor modelo para esa prueba, el cual muestra valores de exactitud sin sobreajuste.

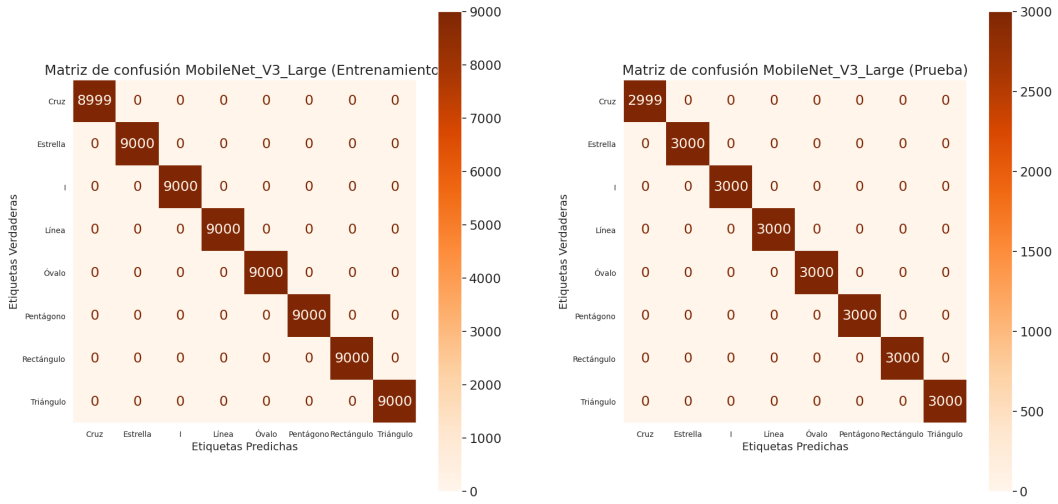


Figura 7.16: Matrices de confusión de nuestra modificación de nuestra modificación de la RNC MobileNet_V3_Large (V2) para la clasificación de la base de datos de 96,000 imágenes. Ambas matrices muestran una correcta clasificación de las imágenes por este modelo.

7.9. RegNet_X_1_6GF

En una de las fases finales del proyecto, se llevó a cabo un reentrenamiento utilizando nuestra adaptación de la Red Neuronal Convolucional RegNet_X_1_6GF durante 20 épocas. El proceso de entrenamiento tomó 1 hora, 53 minutos y 2 segundos. El desempeño más destacado se logró durante la primera época de entrenamiento, alcanzando una exactitud del 99.04 % en el conjunto de prueba y manteniendo un 100 % en el conjunto de entrenamiento. Las gráficas que ilustran la exactitud y el error a lo largo del proceso de entrenamiento de la red se pueden observar en la figura 7.17, mientras que la matriz de confusión se presenta en la figura 7.18.

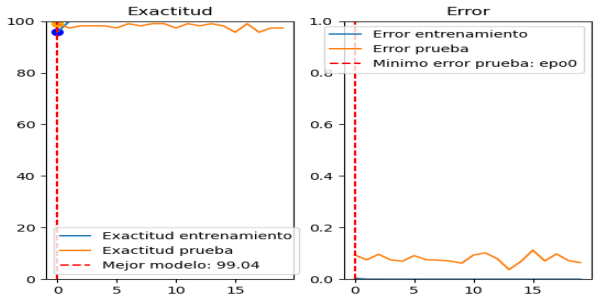


Figura 7.17: A la izq. las gráficas de exactitud de entrenamiento (azul) y prueba (naranja) y a la der. el error de entrenamiento (azul) y prueba (naranja) durante el reentrenamiento de nuestra modificación de la RNC RegNet_X_1_6GF y nuestra base de datos con 96,000 imágenes. La línea punteada vertical representa la obtención del mejor modelo para esa prueba, el cual muestra valores de exactitud sin sobreajuste.

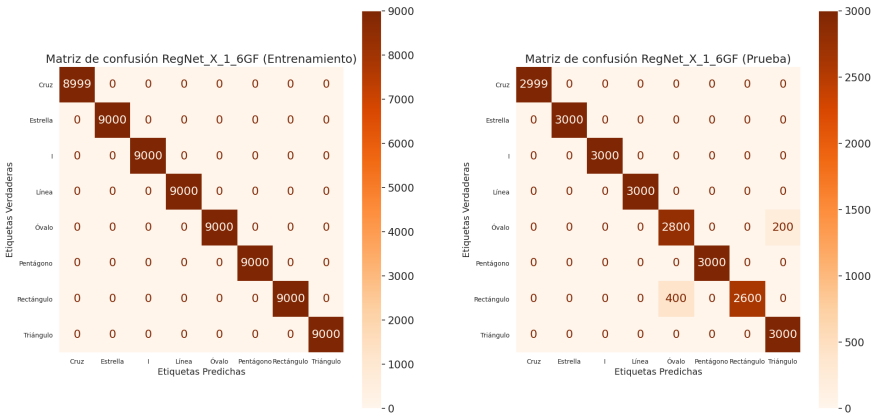


Figura 7.18: Matrices de confusión de nuestra modificación de nuestra modificación de la RNC RegNet_X_1_6GF para la clasificación de la base de datos de 96,000 imágenes. La matriz de la izq. muestra una correcta clasificación de todas las imágenes del conjunto de entrenamiento, en la matriz de la der. se observa que solo las imágenes del conjunto de prueba de las clases óvalo y rectángulo fueron mal clasificada por este modelo.

7.10. ShuffleNet_V2_X0_5

Finalmente la última arquitectura de RNC fue nuestra modificación de ShuffleNet_V2_X0_5 durante 100 épocas. El proceso de entrenamiento tomó 1 hora, 53 minutos y 21 segundos. Con este modelo se obtuvo una exactitud de 99.4 % para el conjunto de prueba y 100 % para el conjunto de entrenamiento en la época 23. Se puede observar las gráficas de exactitud y error durante el entrenamiento en la figura 7.19 y la matriz de confusión en la figura 7.20.

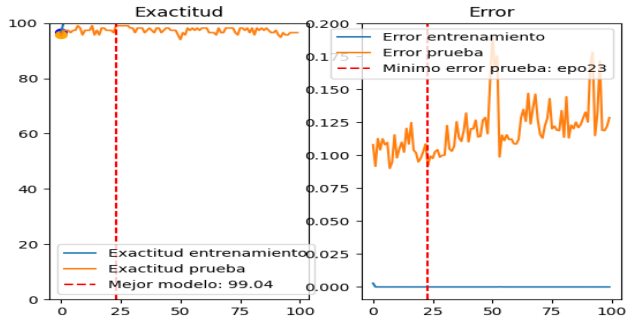


Figura 7.19: A la izq. las gráficas de exactitud de entrenamiento (azul) y prueba (naranja) y a la der. el error de entrenamiento (azul) y prueba (naranja) durante el reentrenamiento de nuestra modificación de la RNC ShuffleNet_V2_X0_5 y nuestra base de datos con 96,000 imágenes. La línea punteada vertical representa la obtención del mejor modelo para esa prueba, el cual muestra valores de exactitud sin sobreajuste.

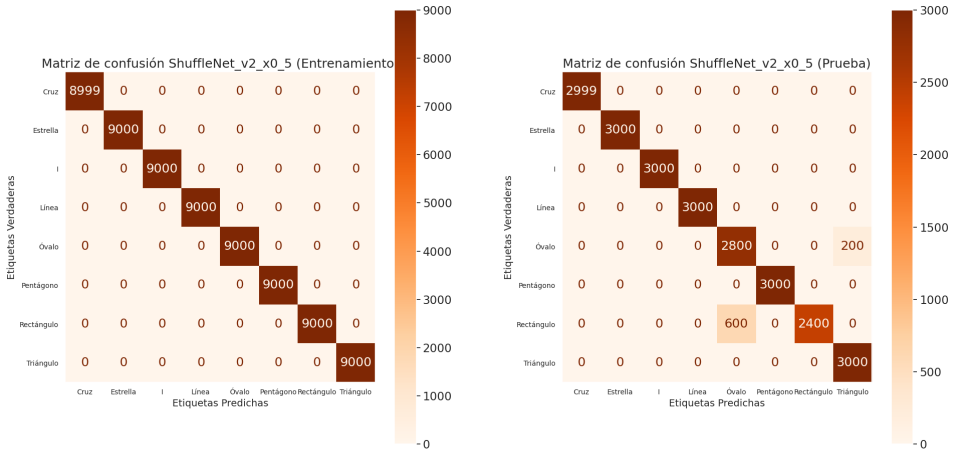


Figura 7.20: Matrices de confusión de nuestra modificación de nuestra modificación de la RNC ShuffleNet_V2_X0_5 para la clasificación de la base de datos de 96,000 imágenes. La matriz de la izq. muestra una correcta clasificación de todas las imágenes del conjunto de entrenamiento, en la matriz de la der. se observa que solo las imágenes del conjunto de prueba de las clases óvalo y rectángulo fueron mal clasificada por este modelo.

7.11. Mejores modelos de RNC para clasificar nuestra base de datos de imágenes

Después de reentrenar nuestras modificaciones de las RNC con nuestra base de datos de imágenes y visualizar el desempeño en las respectivas matrices de confusión, se identificaron las arquitecturas capaces de clasificar correctamente todas las imágenes, estas son la modificación de DenseNet121, EfficientNet_B0 y MobileNet_V3_Large. Con base en estos resultados, se probó únicamente estas tres arquitecturas para la clasificación de la base de datos de números manuscritos MNIST.

7.12. MNIST

Con base en los resultados obtenidos anteriormente, se reentrenaron las tres arquitecturas que presentaron el mejor desempeño en la clasificación de nuestra base de datos con la base de datos MNIST (véase figura 7.21) y la distribución de conjuntos de entrenamiento y prueba por defecto de torchvision. A continuación, se muestran las respectivas gráficas de exactitud y error durante el entrenamiento, además de las matrices de confusión.

Figura 7.21: Imágenes de la base de datos MNIST [15].

7.12.1. DenseNet121

Se reentrenó nuestra modificación de DenseNet121 con los mismos parámetros definidos anteriormente, durante 20 épocas. El proceso de entrenamiento tomó 39 minutos y 23 segundos. El mejor modelo se obtuvo en la época 16 con una exactitud de 99.32 % en el set de prueba. Las gráficas de la exactitud y el error a lo largo del proceso de entrenamiento de la red se pueden observar en la figura 7.22, mientras que la matriz de confusión se presenta en la figura 7.23.

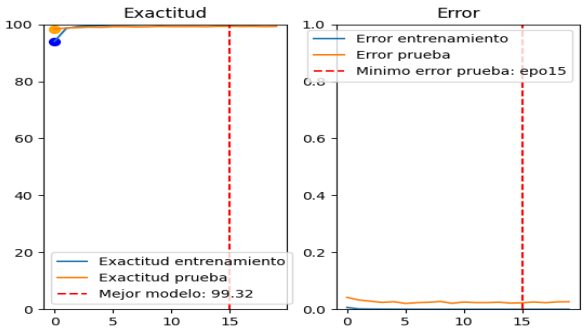


Figura 7.22: A la izq. las gráficas de exactitud de entrenamiento (azul) y prueba (naranja) y a la der. el error de entrenamiento (azul) y prueba (naranja) durante el reentrenamiento de nuestra modificación de la RNC DenseNet121 con la base de datos MNIST. La línea punteada vertical representa la obtención del mejor modelo para esa prueba, el cual muestra valores de exactitud sin sobreajuste.

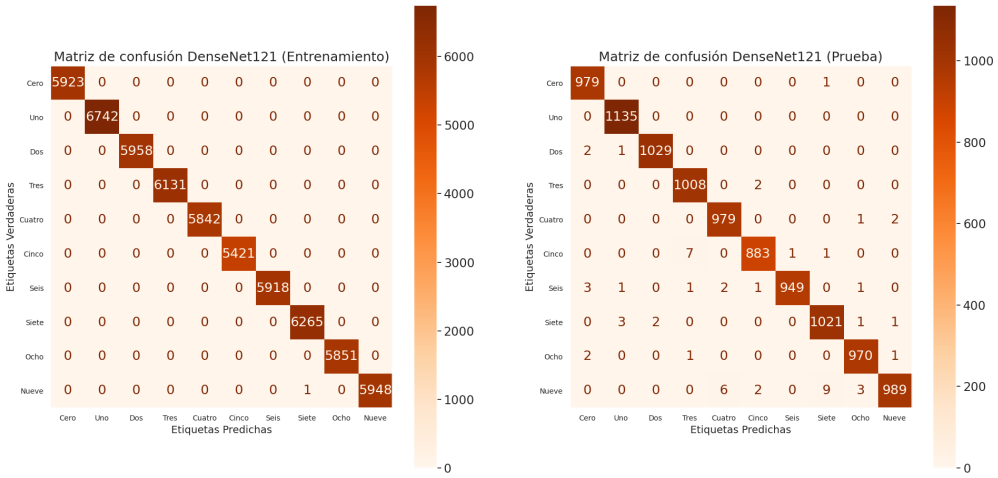


Figura 7.23: Matrices de confusión de nuestra modificación de nuestra modificación de la RNC DenseNet121 para la clasificación de la base de datos MNIST. Ambas matrices muestran una definida coloración en la diagonal principal, señal de una correcta clasificación para la mayoría de las imágenes.

7.12.2. EfficientNet_B0

Del mismo modo se reentrenó nuestra modificación de EfficientNet_B0 durante 20 épocas. El proceso de entrenamiento tomó 26 minutos y 49 segundos. El mejor modelo se obtuvo en la época 14 con una exactitud de 99.25 % en el set de prueba. Las gráficas de la exactitud y el error a lo largo del proceso de entrenamiento de la red se pueden observar en la figura 7.24, mientras que la matriz de confusión se presenta en la figura 7.25.

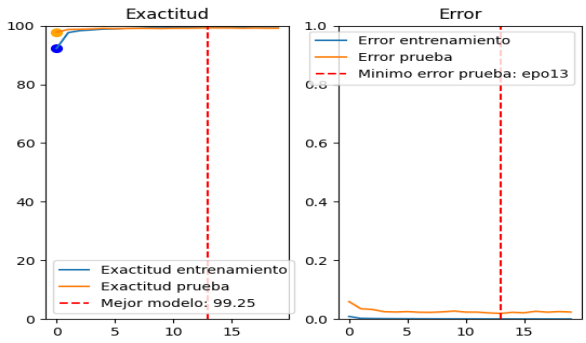


Figura 7.24: A la izq. las gráficas de exactitud de entrenamiento (azul) y prueba (naranja) y a la der. el error de entrenamiento (azul) y prueba (naranja) durante el reentrenamiento de nuestra modificación de la RNC EfficientNet_B0 con la base de datos MNIST. La línea punteada vertical representa la obtención del mejor modelo para esa prueba, el cual muestra valores de exactitud sin sobreajuste.

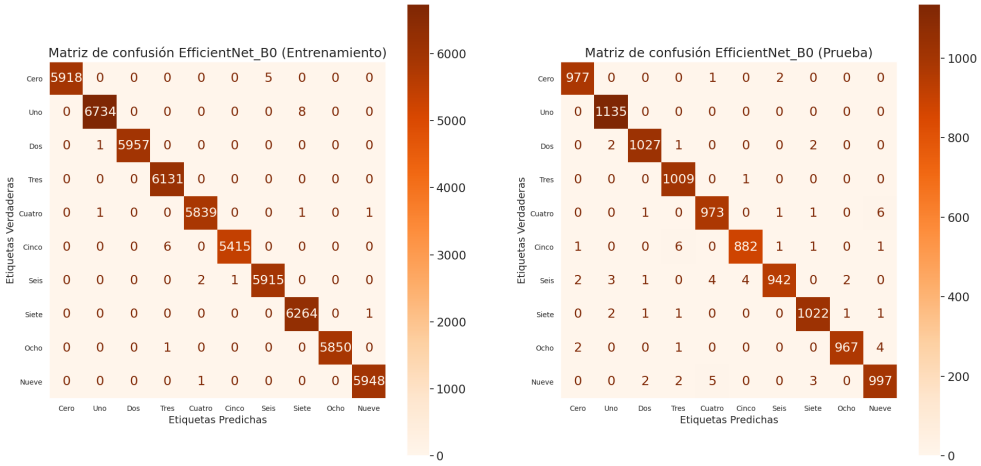


Figura 7.25: Matrices de confusión de nuestra modificación de nuestra modificación de la RNC EfficientNet_B0 para la clasificación de la base de datos MNIST. Ambas matrices muestran una definida coloración en la diagonal principal, señal de una correcta clasificación para la mayoría de las imágenes.

7.12.3. MobileNet_V3_Large

Finalmente, para clasificar la base de datos MNIST reentrenó nuestra modificación de MobileNet_V3_Large con los mismos parámetros definidos anteriormente, durante 20 épocas. El proceso de entrenamiento tomó 18 minutos y 55 segundos. El mejor modelo se obtuvo en la época 15 con una exactitud de 98.99 % en el conjunto de prueba. Las gráficas de la exactitud y el error a lo largo del proceso de entrenamiento de la red se pueden observar en la figura 7.26, mientras que la matriz de confusión se presenta en la figura 7.27.

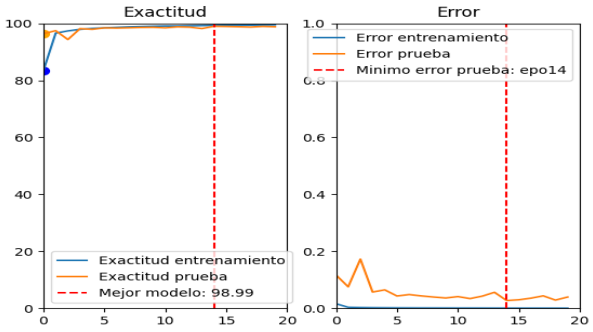


Figura 7.26: A la izq. las gráficas de exactitud de entrenamiento (azul) y prueba (naranja) y a la der. el error de entrenamiento (azul) y prueba (naranja) durante el reentrenamiento de nuestra modificación de la RNC MobileNet_V3_Large con la base de datos MNIST. La línea punteada vertical representa la obtención del mejor modelo para esa prueba, el cual muestra valores de exactitud sin sobreajuste.

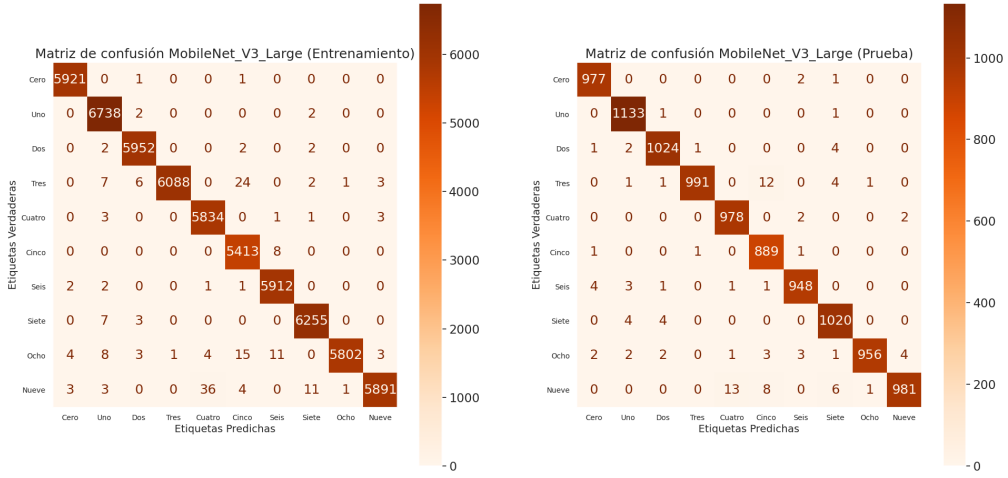


Figura 7.27: Matrices de confusión de nuestra modificación de nuestra modificación de la RNC MobileNet_V3_Large para la clasificación de la base de datos MNIST. Ambas matrices muestran una definida coloración en la diagonal principal, señal de una correcta clasificación para la mayoría de las imágenes.

7.13. International Skin Imaging Collaboration (ISIC)

Basándonos en el informe de Aljohani et al. [1], donde los autores llevaron a cabo un análisis experimental del rendimiento de diversas arquitecturas de RNC preentrenadas implementadas en Keras. El objetivo era construir un clasificador binario entre imágenes de las clases de lesión benigna de queratosis y melanomas utilizando la base de datos ISIC 2019 (véase figura 7.28).

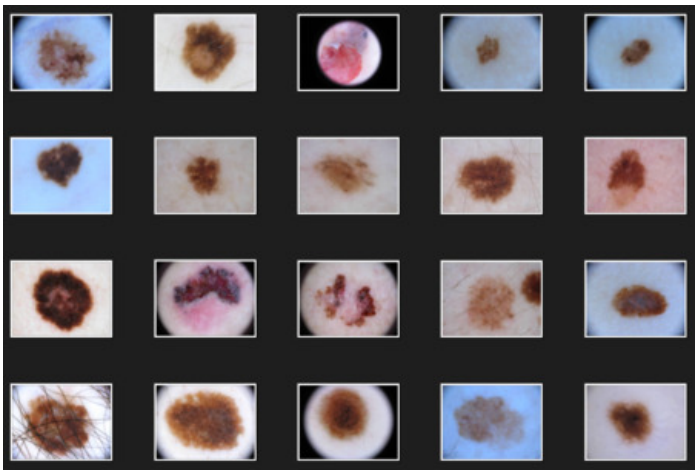


Figura 7.28: Imágenes de la base de datos ISIC 2019 [5].

Entre las arquitecturas que utilizaron se encuentran DenseNet201, MobileNetV2 y GoogLeNet. Destacan a GoogLeNet como la arquitectura que obtuvo el mejor desempeño en la clasificación, con una exactitud del 74.91 % en el conjunto de entrenamiento y 76.08 % en el conjunto de prueba. Para su experimentación, utilizaron un total de 4,522 imágenes de cáncer de piel de tipo melanoma y 7,146 imágenes de lesiones benignas de queratosis, divididos en 80 % de las imágenes para entrenamiento y 20 % para la prueba.

A continuación, presentamos el desempeño de nuestras modificaciones de las RNC: DenseNet121, EfficientNet_B0, GoogLeNet y MobileNet_V3_Large, en esta clasificación.

7.13.1. DenseNet121

Se reentrenó nuestra modificación de la RNC DenseNet121 durante 100 épocas. El proceso de entrenamiento tomó 53 minutos y 29 segundos. El mejor modelo se obtuvo en la época 56 con una exactitud de 97.0 % para el conjunto de entrenamiento y 80.27 % para el conjunto de prueba. En la figura 7.29 se muestra la gráfica de la exactitud y el error del modelo durante el entrenamiento y la matriz de confusión puede visualizarse en la figura 7.30.

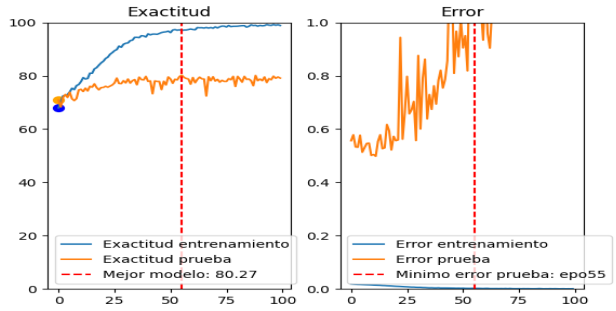


Figura 7.29: Gráficas de exactitud y error durante el reentrenamiento de nuestra modificación de la RNC DenseNet121 con la base de datos de imágenes de melanomas ISIC 2019. A la izq. podemos observar un sobreajuste del modelo al incrementar únicamente la exactitud de modelo para el conjunto de entrenamiento (azul) pero no para el conjunto de prueba (naranja). A la der. observamos un aumento en la gráfica del error para el conjunto de prueba. La línea punteada vertical representa la obtención del mejor modelo para esa prueba, el cual muestra valores de exactitud sin sobreajuste.

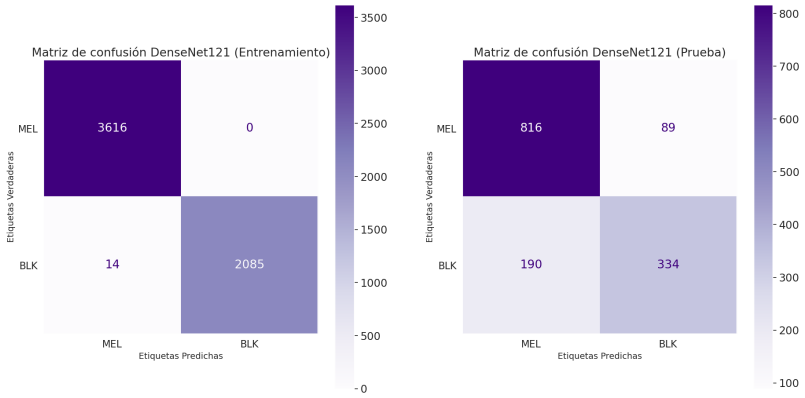


Figura 7.30: Matriz de confusión de nuestra modificación de la RNC DenseNet121 para la clasificación de la base de datos de imágenes de melanomas ISIC 2019. Se pueden observar falsos positivos debido a la clasificación errónea de imágenes de lesiones benignas de queratosis como melanomas, así como falsos negativos por la clasificación errónea de imágenes de melanomas como lesiones benignas de queratosis.

7.13.2. EfficientNet_B0

Para nuestra modificación de la RNC EfficientNet_B0, realizamos un reentrenamiento de 100 épocas. El proceso de entrenamiento tuvo una duración de 53 minutos y 29 segundos. El modelo óptimo se alcanzó en la época 56, logrando una exactitud del 91.10 % en el conjunto de entrenamiento y 78.66 % en el conjunto de prueba. En la figura 7.31, se presenta la gráfica de la exactitud y el error durante el entrenamiento, mientras que la gráfica específica del error se muestra en la figura 7.32.

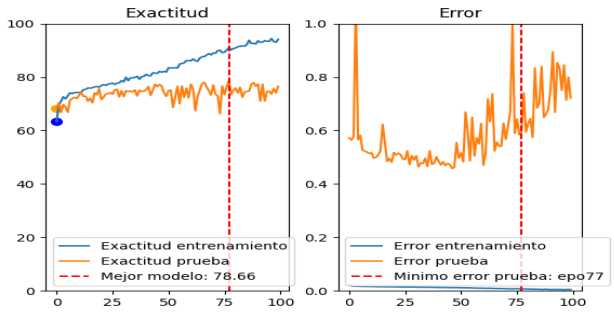


Figura 7.31: Gráficas de exactitud y error durante el reentrenamiento de nuestra modificación de la RNC EfficientNet_B0 con la base de datos de imágenes de melanomas ISIC 2019. A la izq. podemos observar un sobreajuste del modelo al incrementar únicamente la exactitud de modelo para el conjunto de entrenamiento (azul) pero no para el conjunto de prueba (naranja). A la der. observamos un aumento en la gráfica del error para el conjunto de prueba. La línea punteada vertical representa la obtención del mejor modelo para esa prueba, el cual muestra valores de exactitud sin sobreajuste.

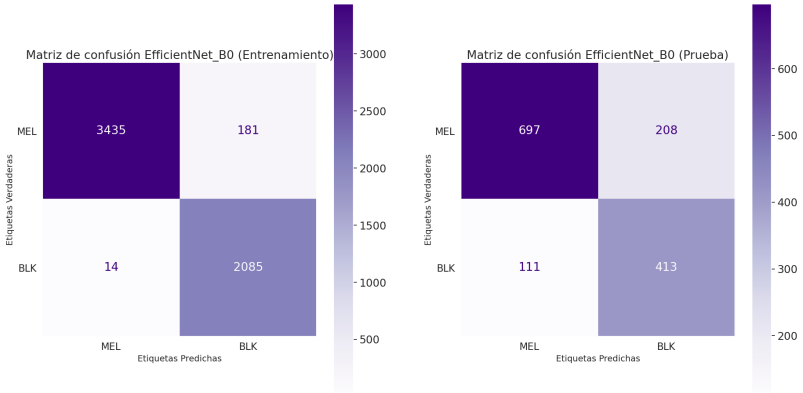


Figura 7.32: Matriz de confusión de nuestra modificación de la RNC EfficientNet_B0 para la clasificación de la base de datos de imágenes de melanomas ISIC 2019. Se pueden observar falsos positivos debido a la clasificación errónea de imágenes de lesiones benignas de queratosis como melanomas, así como falsos negativos por la clasificación errónea de imágenes de melanomas como lesiones benignas de queratosis.

7.13.3. GoogLeNet

Nuestro reentrenamiento de la RNC GoogLeNet se llevó a cabo durante 100 épocas. El proceso de entrenamiento tuvo una duración de 50 minutos y 41 segundos. El mejor modelo se obtuvo en la época 56, con una exactitud del 97.0 % en el conjunto de entrenamiento y 80.27 % en el conjunto de prueba. La figura 7.33 presenta la gráfica de la exactitud y el error del modelo durante el entrenamiento, mientras que la gráfica específica del error está representada en la figura 7.34.

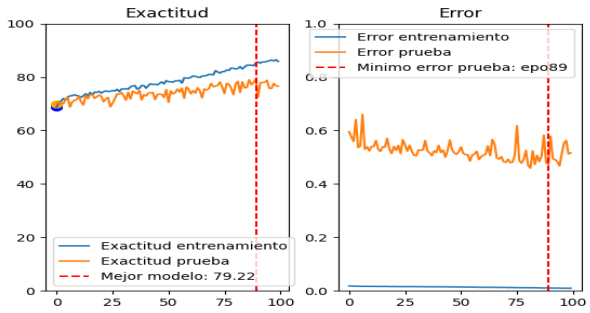


Figura 7.33: Gráficas de exactitud y error durante el reentrenamiento de nuestra modificación de la RNC GoogLeNet con la base de datos de imágenes de melanomas ISIC. A la izq. podemos observar un sobreajuste del modelo al incrementar únicamente la exactitud de modelo para el conjunto de entrenamiento (azul) pero no para el conjunto de prueba (naranja). A la der. observamos un aumento en la gráfica del error para el conjunto de prueba. La línea punteada vertical representa la obtención del mejor modelo para esa prueba, el cual muestra valores de exactitud sin sobreajuste.

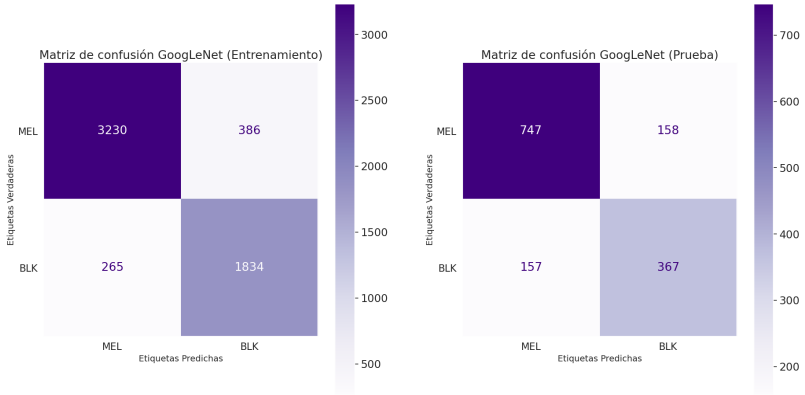


Figura 7.34: Matriz de confusión de nuestra modificación de la RNC GoogLeNet para la clasificación de la base de datos de imágenes de melanomas ISIC 2019. Se pueden observar falsos positivos debido a la clasificación errónea de imágenes de lesiones benignas de queratosis como melanomas, así como falsos negativos por la clasificación errónea de imágenes de melanomas como lesiones benignas de queratosis.

7.13.4. MobileNet_V3_Large

Reentrenamos nuestra modificación de la RNC MobileNet_V3_Large durante 100 épocas. El proceso de entrenamiento tomó 53 minutos y 29 segundos. El mejor modelo se obtuvo en la época 56 con una exactitud del 82.70 % para el conjunto de entrenamiento y 76.98 % para el conjunto de prueba. La figura 7.35 muestra la gráfica de la exactitud y el error del modelo durante el entrenamiento, y la gráfica del error puede visualizarse en la figura 7.36.

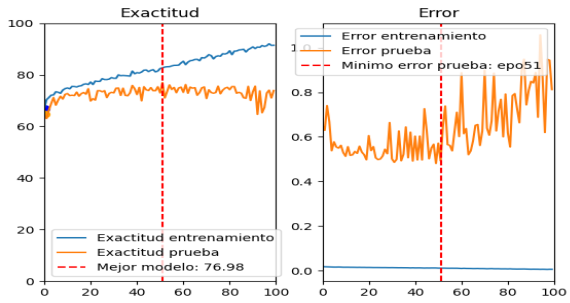


Figura 7.35: Gráficas de exactitud y error durante el reentrenamiento de nuestra modificación de la RNC MobileNet_V3_Large con la base de datos de imágenes de melanomas ISIC 2019. A la izq. podemos observar un sobreajuste del modelo al incrementar únicamente la exactitud de modelo para el conjunto de entrenamiento (azul) pero no para el conjunto de prueba (naranja). A la der. observamos un aumento en la gráfica del error para el conjunto de prueba. La línea punteada vertical representa la obtención del mejor modelo para esa prueba, el cual muestra valores de exactitud sin sobreajuste.

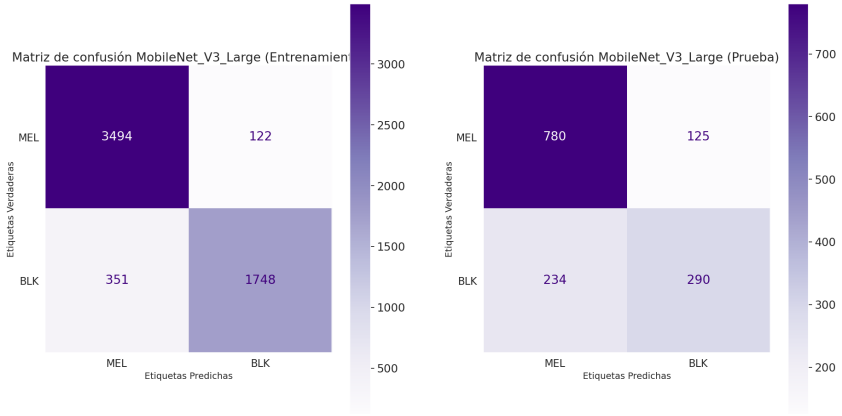


Figura 7.36: Matriz de confusión de nuestra modificación de la RNC MobileNet_V3_Large para la clasificación de la base de datos de imágenes de melanomas ISIC 2019. Se pueden observar falsos positivos debido a la clasificación errónea de imágenes de lesiones benignas de queratosis como melanomas, así como falsos negativos por la clasificación errónea de imágenes de melanomas como lesiones benignas de queratosis.

7.14. Sumario

Nuestras modificaciones a cada una de las RNC utilizadas en este estudio llevaron a un sobreajuste durante el entrenamiento para clasificar nuestra BDO de figuras con 480 imágenes. Por lo tanto, fue necesario aumentar esta base de datos 200 veces, resultando en 72,000 imágenes para entrenamiento y 24,000 para evaluación, con el fin de proporcionar suficientes imágenes a las RNC, evitar el sobreajuste y permitir la generalización del conocimiento para clasificar correctamente las imágenes. En la tabla 7.1 se muestra los resultados que obtuvimos con nuestras ocho modificaciones de RNC en la clasificación de nuestra base de datos aumentada 200 veces.

Tabla 7.1: Resultados de nuestras modificaciones de RNC en la clasificación de nuestra base de datos aumentada 200 veces.

Modelo	**Tiempo de entrenamiento**	**Exactitud entrenamiento**	**Exactitud prueba**
DenseNet121	**5 hrs. 8 min. y 30 seg.**	**100 %**	**99.87 %**
EfficientNet_B0	**3 hrs. 44 min. y 20 seg.**	**100 %**	**99.87 %**
GoogLeNet	2 hrs. 39 min. y 3 seg.	100 %	99.87 %
MNASNet0_5	4 hrs. 43 min. y 30 seg.	100 %	99.87 %
MobileNet_V3_Large	**1 hr. 25 min. y 29 seg.**	**100 %**	**99.87 %**
RegNet_X_1_6GF	1 hr. 53 min. y 2 seg.	100 %	99.04 %
ShuffleNet_V2_X0_5	1 hr. 53 min. y 21 seg.	100 %	99.4 %
SimpleNet_5m_m2	3 hrs. 31 min. y 5 seg.	97.56 %	97.37 %

De las ocho arquitecturas que modificamos, solo las siguientes fueron capaces de clasificar correctamente todas las imágenes de nuestra base de datos:

- DenseNet121
- EfficienNet_B0
- MobileNet_V3_Large

Con estas tres arquitecturas, se procedió a probar su desempeño en la clasificación de la base de datos MNIST. En la tabla 7.2 se muestran los resultados obtenidos, siendo nuestra modificación de la RNC DenseNet121 la que alcanzó la mejor exactitud en comparación con las otras dos.

Tabla 7.2: Resultados de nuestras modificaciones de RNC en la clasificación de la base de datos MNIST.

Modelo	**Tiempo de entrenamiento**	**Exactitud entrenamiento**	**Exactitud prueba**
DenseNet121	**39 min. 23 seg.**	**99.9 %**	**99.32 %**
EfficientNet_B0	26 min. 49 seg.	99.3 %	99.25 %
MobileNet_V3_Large	18 min. 55 seg.	99.3 %	98.99 %

La tabla 7.3 presenta un resumen de los resultados alcanzados mediante nuestra modificación de RNC para la clasificación binaria de imágenes de melanoma frente a lesiones benignas de queratosis, utilizando la base de datos ISIC. En la parte inferior de esta tabla, con letras itálicas, se detallan los resultados que Aljohani et al. reportaron. Estos sirven como referencia comparativa con los resultados de nuestras propuestas, las cuales demostraron ser superiores en términos de exactitud. A diferencia de la metodología de Aljohani et al., que empleó la arquitectura completa de las RNC, nuestras modificaciones se caracterizan por tener una menor cantidad de parámetros entrenables, lo que optimiza el proceso. Además, es importante mencionar que en su informe no se especifica el tiempo de entrenamiento para cada uno de los modelos utilizados

Tabla 7.3: Resultados de nuestras modificaciones de RNC y los resultados reportados por Aljohani et al. [1] en la clasificación de la base de datos de melanomas ISIC.

Modelo	**Tiempo de entrenamiento**	**Exactitud entrenamiento**	**Exactitud prueba**
DenseNet121	**53 min. 29 seg.**	**97.0 %**	**80.27 %**
EfficientNet_B0	50 min. 41 seg.	91.1 %	78.66 %
GoogLeNet	3 hrs. 44 min. 07 seg.	85.2 %	79.21 %
MobileNet_V3_Large	50 min. 49 seg.	82.7 %	76.98 %
DenseNet201	*No reportado*	*73.96 %*	*74.68 %*
GoogLeNet	*No reportado*	*74.91 %*	*76.08 %*
MobileNet_V3_Large	*No reportado*	*71.88 %*	*73.98 %*

Capítulo 8

Conclusiones

Se creó una base de datos con 480 imágenes divididas en ocho clases, lo que permitió entrenar y evaluar el desempeño de algoritmos de aprendizaje automático tradicional y de aprendizaje profundo. Esta base de datos de imágenes también se utilizó en la realización del artículo de conferencia "Image Classification with Recurrent Spiking Neural Networks".

Los algoritmos de MVS y red neuronal directa, entrenados con los momentos de Hu, clasificaron correctamente la BDO y lograron generalizar el conocimiento para clasificar correctamente las imágenes de la BDA. El tiempo de entrenamiento de estos modelos con los momentos de Hu no fue mayor a 1 ms.

Para las 2,592 características HOG, reducidas mediante ACP a 64 dimensiones, el algoritmo MVS mostró el mejor desempeño, alcanzando una exactitud de 0.9895. Sin embargo, el tiempo de entrenamiento fue mayor respecto a los modelos entrenados con los momentos de Hu, registrando una duración de 20.4 ms.

Este análisis resalta la diferencia en el tiempo de entrenamiento de los algoritmos según las características utilizadas a pesar de ser las mismas imágenes. Además, un vector de características de mayor tamaño aumenta el tiempo de entrenamiento, pero no garantiza una mayor exactitud. En general los algoritmos de aprendizaje automático tradicional no presentaron un tiempo de entrenamiento superior a medio segundo.

Respecto a los algoritmos de aprendizaje profundo, el criterio de selección basado en el número de parámetros de las RNC, limitado a un máximo de 10 M, permite utilizar modelos preentrenados más pequeños en comparación con redes como AlexNet y lograr una correcta clasificación de las imágenes. También, la modificación de las capas de la etapa de extracción de características de las RNC, adaptándolas al tamaño de las imágenes de la base de datos que se pretende clasificar, como la BDO, reduce significativamente el número de parámetros de la RNC.

En la clasificación de imágenes con RNC, un mayor número de parámetros entre-

nables no garantiza un mejor rendimiento para una misma tarea. Es posible reducir en más de un 80 % el tamaño de una RNC y aún así alcanzar una exactitud similar o superior en la clasificación de imágenes en comparación con la RNC original. Esta reducción disminuye el tiempo de entrenamiento de la modificación de RNC respecto a la arquitectura original. También la transferencia de aprendizaje contribuye a disminuir los tiempos de entrenamiento, ya que los valores transferidos están optimizados para ser un extractor de características eficaz. Además, la técnica de aumento de datos ayuda a reducir el tiempo necesario para alcanzar la máxima exactitud que la RNC puede lograr.

Entrenar una RNC con la CPU del ordenador puede generar errores durante este proceso, por lo que se requiere utilizar una UPG para entrenar estos modelos. No fue posible entrenar las modificaciones de RNC con la BDO sin que estas se sobreajustaran, debido a esto fue necesario aumentar 200 veces el tamaño, resultando en un total de 96,000 imágenes.

Podemos determinar que los algoritmos de aprendizaje automático tradicional, utilizando algoritmos de extracción de características como Hu y/o HOG, son los adecuados para clasificar una base de datos con un aproximado de 500 imágenes. Sin embargo, los algoritmos de aprendizaje profundo, es decir, las RNC requieren de un gran conjunto de imágenes, en el orden de los 96,000 imágenes o el equivalente a aumentar en el orden de las 200 veces una base de datos de imágenes con aproximadamente 500 imágenes, para obtener una clasificación exacta de la base de datos.

Es posible aplicar técnicas de aumento de datos a bases de datos de imágenes para incrementar su tamaño y habilitar el uso de RNC. Sin embargo, es necesario contar con una UPG para el entrenamiento, ya que, sin este recurso, aunque se disponga de un gran conjunto de imágenes, el entrenamiento del modelo podría no ser computacionalmente viable. La falta de acceso a una UPG limita el uso de RNC, lo que obliga a recurrir a algoritmos de aprendizaje automático tradicionales para la clasificación de imágenes.

También presentamos las conclusiones derivadas de la implementación y evaluación de nuestra modificación de RNC, específicamente la basada en la DenseNet121. Esta red fue sometida a una serie de pruebas en tiempo real para determinar su eficacia en la clasificación de objetos de nuestra base de datos. Para ello, se dispuso una cámara fija en un trípode y se procedió a realizar las pruebas con impresiones de las figuras a una distancia inicial de 50 cm. Posteriormente, se incrementó la distancia en intervalos de 10 cm, evaluando la capacidad del modelo para mantener una clasificación precisa a medida que las figuras se alejaban.

Además, se rotaron las figuras alrededor de su eje vertical central en cada intervalo para determinar el ángulo máximo de rotación en el cual el modelo aún era capaz

de clasificar correctamente. Este proceso no solo probó la robustez del modelo ante cambios en la orientación de los objetos, sino también su habilidad para identificarlos fuera del área de clasificación predefinida, es decir, fuera del cuadro verde central de 64×64 píxeles.

Las pruebas se llevaron a cabo para cada figura, siguiendo un orden alfabético. Los resultados de estas pruebas se documentan y pueden ser consultados en la figura 8.1. Estos experimentos no solo confirmaron la precisión de nuestro modelo en condiciones ideales y estáticas, sino que también demostraron su eficiencia en tiempo real, con las figuras alejadas y rotadas respecto a la cámara, como se reportó anteriormente, subrayando la viabilidad de nuestra propuesta basada en la RNC DenseNet121 como una herramienta confiable para la clasificación en tiempo real.

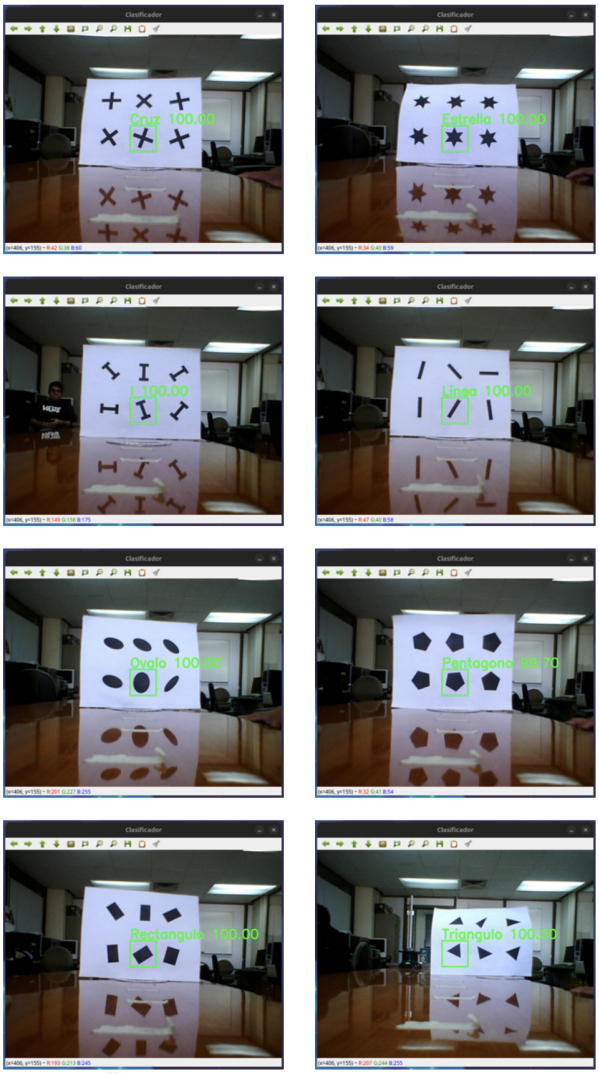


Figura 8.1: Ejecución en tiempo real de nuestra modificación de DenseNet121 para la clasificación de las figuras de la BDO.

En la tabla 8.1, se muestran los parámetros de distancia y ángulo para cada una de las figuras.

Tabla 8.1: Resultados de la clasificación en tiempo real con nuestra modificación la RNC DenseNet121.

Figura	**Distancia máxima**	**Ángulo máximo**
Cruz	60 cm	10^{o}
Estrella	50 cm	35^{o}
I	50 cm	40^{o}
Línea	60 cm	30^{o}
Óvalo	50 cm	10^{o}
Pentágono	50 cm	10^{o}
Rectángulo	50 cm	10^{o}
Triángulo	50 cm	55^{o}

Respecto a la área que debe estar dentro de la zona captura de la cámara, las figuras Cruz, Rectángulo y Pentágono necesitan estar completamente dentro de está área para ser clasificados correctamente.

Nuestra modificación de la RNC DenseNet121 también fue la que mejor desempeño tuvo en la clasificación de la base de datos MNIST, el tiempo de entrenamiento fue de 53 minutos y 29 segundo, logrando un 99.9 % de exactitud en el conjunto de entrenamiento y un 99.32 % en el conjunto de prueba, superando al otro par de RNC que modificamos y probamos con esta base de datos.

Las cuatro modificaciones de nuestras RNC, DenseNet121, EfficientNet_B0, GoogLeNet y MobileNet, utilizadas para la clasificación binaria entre imágenes de cáncer de piel tipo melanoma e imágenes de lesiones benignas de queratosis, mostraron un mejor desempeño que lo reportado en la literatura [1]. Es relevante mencionar que, mientras en el estudio de referencia se utilizó la arquitectura original, nosotros empleamos nuestras modificaciones con una considerable disminución en el tamaño de la RNC, logrando obtener mejores resultados. El mejor resultado en la clasificación lo reporta nuestra modificación de la red DenseNet121, con un 97.0 % de exactitud en el conjunto de entrenamiento y un 80.27 % en el conjunto de prueba, superando al mejor resultado reportado por Aljohani et al., quienes, utilizando la RNC GoogLeNet sin modificaciones, alcanzaron una exactitud del 74.91 % en el conjunto de entrenamiento y del 76.08 % en el conjunto de prueba.

8.1. Trabajo futuro

En esta sección se presentan propuestas para investigaciones futuras basada en los hallazgos y metodologías desarrolladas en este estudio.

- Modificar la base de datos de imágenes, aumentando el número de clases y la diversidad de las imágenes.
- Para la clasificación de imágenes médicas se podría extender la metodología a la clasificación de otras enfermedades dermatológicas.
- Evaluar la efectividad de los modelos en la detección de enfermedades en imágenes médicas de diferentes modalidades (radiografías, resonancias magnéticas, etc).
- Realizar estudios clínicos para validar la efectividad del modelo en entornos reales.
- Crear una aplicación o plataforma que permita a los profesionales de la salud utilizar los modelos de manera intuitiva.
- Se podría investigar métodos para aumentar la cantidad de imágenes en las aplicaciones médicas.

Bibliografía

[1] Khalil Aljohani y Turki Turki. "Automatic classification of melanoma skin cancer with deep convolutional neural networks". En: *Ai* 3.2 (2022), págs. 512-525.

[2] Daniel Berrar. "Bayes' theorem and naive Bayes classifier". En: *Elsevier* (2019).

[3] Christopher M Bishop y Nasser M Nasrabadi. *Pattern recognition and machine learning*. Vol. 4. 4. Springer, 2006.

[4] Paolo Carli et al. "Pattern analysis, not simplified algorithms, is the most reliable method for teaching dermoscopy for melanoma diagnosis to residents in dermatology". En: *British Journal of Dermatology* 148.5 (2003), págs. 981-984.

[5] Noel Codella, Veronika Rotemberg, Philipp Tschandl et al. *ISIC 2019: Skin Lesion Analysis Towards Melanoma Detection*. Accessed: 2024-07-19. 2019. URL: https://challenge.isic-archive.com/data/#2019.

[6] Shuang Cong y Yang Zhou. "A review of convolutional neural network architectures and their optimizations". En: *Artificial Intelligence Review* 56.3 (2023), págs. 1905-1969.

[7] Navneet Dalal y Bill Triggs. "Histograms of oriented gradients for human detection". En: *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*. Vol. 1. Ieee. 2005, págs. 886-893.

[8] Reza Ebrahimzadeh y Mahdi Jampour. "Efficient handwritten digit recognition based on histogram of oriented gradients and SVM". En: *International Journal of Computer Applications* 104.9 (2014).

[9] Andre Esteva et al. "Dermatologist-level classification of skin cancer with deep neural networks". En: *nature* 542.7639 (2017), págs. 115-118.

[10] Seyyed Hossein Hasanpour et al. "Lets keep it simple, using simple architectures to outperform deeper and more complex architectures". En: *arXiv preprint arXiv:1608.06037* (2016).

[11] Herbert Jaeger. "The "echo state" approach to analysing and training recurrent neural networks-with an erratum note". En: *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report* 148.34 (2001), pág. 13.

[12] Kanika Kalra, Anil Kumar Goswami y Rhythm Gupta. "A comparative study of supervised image classification algorithms for satellite images". En: *International journal of electrical, electronics and data communication* 1.10 (2013), págs. 10-16.

[13] Alex Krizhevsky, Ilya Sutskever y Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". En: *Advances in neural information processing systems* 25 (2012).

[14] Yann LeCun, Yoshua Bengio y Geoffrey Hinton. "Deep learning". En: *nature* 521.7553 (2015), págs. 436-444.

[15] Yann LeCun, Corinna Cortes y CJ Burges. *The MNIST Database of Handwritten Digits*. Accessed: 2024-07-19. 1998. URL: http://yann.lecun.com/exdb/mnist/.

[16] Peng Liu et al. "SVM or deep learning? A comparative study on remote sensing image classification". En: *Soft Computing* 21 (2017), págs. 7053-7065.

[17] David G Low. "Distinctive image features from scale-invariant keypoints". En: *Journal of Computer Vision* 60.2 (2004), págs. 91-110.

[18] David G Lowe. "Distinctive image features from scale-invariant keypoints". En: *International journal of computer vision* 60 (2004), págs. 91-110.

[19] Dengsheng Lu y Qihao Weng. "A survey of image classification methods and techniques for improving classification performance". En: *International journal of Remote sensing* 28.5 (2007), págs. 823-870.

[20] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". En: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, págs. 8024-8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[21] Marius-Constantin Popescu et al. "Multilayer perceptron and neural networks". En: *WSEAS Transactions on Circuits and Systems* 8.7 (2009), págs. 579-588.

[22] F Rosenblatt y S Papert. *Perceptron*. Vol. 9. April, 2021.

[23] Ashkan Shakarami y Hadis Tarrah. "An efficient image descriptor for image classification and CBIR". En: *Optik* 214 (2020), pág. 164833.

[24] Sho Sonoda y Noboru Murata. "Neural network with unbounded activation functions is universal approximator". En: *Applied and Computational Harmonic Analysis* 43.2 (2017), págs. 233-268. ISSN: 1063-5203. DOI: https://doi.org/10.1016/j.acha.2015.12.005. URL: https://www.sciencedirect.com/science/article/pii/S1063520315001748.

[25] Shan Suthaharan y Shan Suthaharan. "Support vector machine". En: *Machine learning models and algorithms for big data classification: thinking with examples for effective learning* (2016), págs. 207-235.

[26] Tomasz Szandała. "Review and comparison of commonly used activation functions for deep neural networks". En: *Bio-inspired neurocomputing* (2021), págs. 203-224.

[27] Cesar H Valencia, Marley MBR Vellasco y Karla Figueiredo. "Echo state networks: novel reservoir selection and hyperparameter optimization model for time series forecasting". En: *Neurocomputing* 545 (2023), pág. 126317.

[28] ME Vestergaard et al. "Dermoscopy compared with naked eye examination for the diagnosis of primary melanoma: a meta-analysis of studies performed in a clinical setting". En: *British Journal of Dermatology* 159.3 (2008), págs. 669-676.

[29] Pin Wang, En Fan y Peng Wang. "Comparative analysis of image classification algorithms based on traditional machine learning and deep learning". En: *Pattern Recognition Letters* 141 (2021), págs. 61-67.

[30] Ting-Wei Wu et al. "Applications of convolutional neural networks for intelligent waste identification and recycling: A review". En: *Resources, Conservation and Recycling* 190 (2023), pág. 106813.

[31] Jure Zupan. "Introduction to artificial neural network (ANN) methods: what they are and how to use them". En: *Acta Chimica Slovenica* 41.3 (1994), pág. 327.