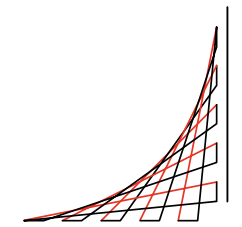
Sintonizador de hiperparámetros de redes neuronales usando computación natural

(PSO: Optimización de enjambre de partículas)

Trabajo Dirigido
Ingeniería de Sistemas

Autores:
Santiago Andrés Rocha Cristancho
Sebastián Rojas Bueno
Directora:
María Irma Díaz Rozo



Decanatura de Ingeniería de Sistemas
Escuela Colombiana de Ingeniería Julio Garavito
Bogotá, Colombia
Julio 2024

Resumen

Al diseñar modelos de redes neuronales, la selección de hiperparámetros puede ser compleja si se desean obtener resultados favorables en problemas difíciles. Resolver este problema requiere ya sea una alta experiencia o una metodología acertada de sintonización. Nuestro objetivo ha sido crear una solución que, utilizando una gama predefinida de hiperparámetros, sintonice automáticamente modelos de redes neuronales convolucionales (CNN) eficaces. Esta solución está pensada para ser útil tanto para usuarios con poca experiencia en redes neuronales y sus hiperparámetros, como para aquellos más inmersos en esta rama de la inteligencia artificial.

A pesar de los avances en automatizar la selección de hiperparámetros con herramientas especializadas, aún hay áreas de mejora para lograr la eficacia de las técnicas utilizadas. En este contexto, el algoritmo de optimización de enjambres de partículas (PSO) es una opción atractiva para la sintonización. PSO emula las interacciones de agentes en enjambres naturales con el propósito de encontrar buenas soluciones a problemas con espacios de búsqueda vastos, destacándose por su capacidad para explorar eficazmente dicho espacio y abordar problemáticas como los óptimos locales. Por otro lado, el problema de reconocimiento facial se postula como un reto pertinente para comprender el proceso de sintonización y evaluar la automatización propuesta. Este problema ha enfrentado múltiples desafíos desde los años 60 y tiene diversas aplicaciones relevantes en la actualidad.

Nuestra solución tiene tres focos principales:

- Crear un *framework* de sintonización de hiperparámetros utilizando PSO. Este *framework* demostrará su funcionamiento y eficacia frente a otros sintonizadores usando los conjuntos de datos MNIST y CIFAR10.
- Generar un modelo de reconocimiento facial de referencia utilizando el modelo VGG16 con datos de VGGFace2. Este modelo se entrenará aplicando métodos de sintonización manual.
- Comparar y aplicar los resultados de ambos enfoques (PSOTuner y sintonización manual) para evaluar la efectividad del PSOTuner en problemas complejos como el reconocimiento facial.

En términos de resultados, PSOT (PSO Tuner) ha superado los resultados obtenidos por los sintonizadores de Keras Tuner bajo las mismas condiciones y datasets MNIST y CIFAR10, a partir de hipermodelos de CNN que implementan diferentes heurísticos. Las diferencias en los resultados de exactitud con MNIST son de entre 0.2 % y 0.5 % en validación y pruebas, mientras que con CIFAR10 se han logrado mejoras de entre 10 % y 15 %.

Esto sienta una base sólida para su aplicación en desafíos más complejos, como el reconocimiento facial, donde se alcanzó una exactitud del 77.91 % en pruebas utilizando la arquitectura VGG sintonizada manualmente, frente a una exactitud del 88.09 % lograda con sintonización manual, demostrando que aunque otorgue buenos resultados, aún es importante la mano experta quien sintonice con mayor conocimiento.

De esta forma, nuestra contribución principal es el *framework* de sintonización que aprovecha la experiencia lograda en la sintonización manual, cumpliendo con nuestro objetivo.

Índice general

Introducción

Antecedentes y justificación

Antecedentes

La sintonización de hiperparámetros juega un papel crucial en la mejora del rendimiento de los modelos de aprendizaje profundo. La elección adecuada de estos hiperparámetros influye directamente en la capacidad del modelo para generalizar patrones a partir de datos de entrada, lo que resulta fundamental en tareas complejas como el reconocimiento facial. En este contexto, el algoritmo de optimización por enjambre de partículas (PSO, por sus siglas en inglés) es una herramienta bioinspirada que puede ser efectiva para abordar este desafío.

Importancia de la sintonización de hiper-parámetros La sintonización de hiperparámetros implica ajustar factores de diseño del modelo y de configuración para los algoritmos de aprendizaje, como número de capas, funciones de activación y tasas de aprendizaje, entre otros. Estos hiperparámetros impactan en la capacidad de la red neuronal para aprender de los datos y generalizar patrones complejos. La sintonización exitosa puede conducir a modelos más efectivos, robustos y adaptables a diferentes escenarios.

Optimización por enjambre de partículas (PSO) PSO se inspira en el comportamiento social de las partículas en un enjambre, como las aves migratorias o cardúmenes de peces. Cada partícula representa una solución en el espacio de búsqueda de soluciones. Estas partículas se mueven a través del espacio en busca de la mejor solución, actualizando su posición y velocidad en función de su experiencia pasada y la información de sus vecinos. Este enfoque colaborativo y adaptativo permite una exploración eficiente del espacio de búsqueda.

PSO en la optimización de hiperparámetros En el contexto de la sintonización de hiperparámetros, cada partícula en PSO representa un conjunto específico de hiperparámetros y su calidad está determinada por la función objetivo que mide la eficacia del modelo. Las partículas se mueven hacia regiones del espacio de búsqueda que han demostrado ser más prometedoras, lo que permite una convergencia hacia buenas soluciones.

Justificación

La sintonización de hiperparámetros es un área de investigación activa en el aprendizaje automático. Los algoritmos bioinspirado se han utilizado para buscar conjuntos efectivos en espacios de búsqueda complejos. Investigaciones recientes destacan la eficacia de PSO en la búsqueda de hiperparámetros para redes neuronales [10]. La idea central es desarrollar un sintonizador bioinspirado, específicamente PSO, para encontrar de manera eficiente combinaciones eficaces de hiperparámetros en el espacio de búsqueda de las CNN (redes neuronales convolucionales, por sus siglas en inglés) e implementarlo en el Keras. El reto seleccionado para este sintonizador es el reconocimiento facial.

PSO La elección de PSO como sintonizador se basa en su adaptabilidad dinámica, eficiencia computacional y capacidad para evitar mínimos locales, como se detalla en la sección anterior. [8].

CNN La decisión de sintonizar hiperparámetros específicamente en CNN se fundamenta en el interés por abordar la complejidad inherente de estos modelos y la influencia significativa que tienen los hiperparámetros en su rendimiento [16]. La complejidad y el gran número de hiperparámetros en CNN hacen que la sintonización manual sea poco práctica y propensa a errores. La aplicación de técnicas de optimización, como PSO, ofrece una solución automatizada para encontrar buenas configuraciones, mejorando así el rendimiento y la eficiencia de las CNN en tareas de clasificación de imágenes.

Keras La elección de Keras para nuestro proyecto se basó en su facilidad para el desarrollo de redes neuronales y su oferta de sintonizadores de hiperparámetros. Keras ofrece una interfaz intuitiva y de alto nivel espe cíficamente diseñada para trabajar con arquitecturas de redes, simplificando la implementación y experimentación.

Reconocimiento Facial La congestión en las entradas de las universidades, especialmente durante la verificación de la carnetización en el ingreso, es un problema común en muchas instituciones educativas. Este inconveniente no solo afecta la eficiencia del proceso de acceso, sino que también puede generar frustración entre los estudiantes, el personal y visitantes. Las soluciones tradicionales, como el uso de tarjetas de identificación físicas, a menudo resultan en retrasos considerables, especialmente en momentos de alto flujo de personas.

La tecnología de reconocimiento facial y las redes neuronales ofrecen una oportunidad para abordar estos desafíos de manera innovadora. Integrar estas soluciones en el contexto universitario podría tener un impacto positivo en la experiencia general de la comunidad académica, aunque no está exenta de problemas relacionados a la exactitud del reconocimiento.

El reconocimiento facial es una rama de la visión por computadora que se centra en identificar y verificar caras humanas a partir de imágenes o videos. Este problema se ha vuelto crucial en diversas aplicaciones, desde la seguridad y la vigilancia hasta la autenticación biométrica. En especial, las redes neuronales convolucionales (CNN) han demostrado ser efectivas en el reconocimiento facial debido a su capacidad para aprender patrones complejos y representaciones características. Modelos preentrenados, como VGGFace y FaceNet, han demostrado ser excepcionales para extraer características representativas de rostros, lo que permite la identificación precisa incluso en condiciones variables de iluminación y pose.

Objetivos

El objetivo general del proyecto es el desarrollo de un sintonizador de hiperparámetros basado en PSO para optimizar CNN. El problema seleccionado como reto para el sintonizador es el reconocimiento facial.

Los objetivos específicos son:

- Desarrollar un sintonizador de hiperparámetros para redes, enfocados principalmente a CNN
- Entrenar CNNs que solucionen problemas clásicos de clasificación de imágenes usando el sintonizador
- Desarrollar una CNN para el reconocimiento facial a modo de reto de experimentación y de aplicación

Metodología

La propuesta es un desarrollo en paralelo de un sintonizador de hiperparámetros a partir de PSO y el desarrollo de una CNN para el reconocimiento facial. Adoptaremos una metodología ágil basada en el marco de trabajo Scrum, facilitando la entrega incremental de funcionalidades, la revisión periódica de avances y la capacidad de responder a cambios inesperados en nuestro desarrollo.

Las fases para el desarrollo del proyecto son:

1. *Revisión del estado del arte.* Revisión de documentos científicos, trabajos relacionados, investigaciones previas y tecnologías relevantes.
2. *Implementación del sintonizador para CNN.* Desarrollo del sintonizador como un*framework* bioinspirado, basado en PSO.
3. *Uso experimental y demostración de funcionamiento del sintonizador.* Hacer uso del sintonizador desarrollado en diferentes redes neuronales convolucionales, permitiendo analizar su funcionamiento e identificar potenciales mejoras.
4. *Desarrollo de una CNN para el reconocimiento facial.*
    - *Selección de dataset.* Búsqueda, evaluación y selección de *datasets* públicos.
    - *Enriquecimiento del dataset.* Adición al dataset de los datos del subgrupo de estudiantes.
    - *Entrenamiento de la red.* Entrenamiento de la red configurando los hiperparámetros de manera manual.
5. *Implementación de funcionalidades adicionales.* Desarrollo de características secundarias adicionales según los requisitos. Perfeccionamiento continuo del sintonizador para asegurar su adaptabilidad y funcionalidad.
6. *Pruebas de funcionamiento.* Evaluación del sistema, verificación de las pruebas implementadas y la introducción de pruebas adicionales. Corrección de errores identificados durante las pruebas para garantizar el funcionamiento robusto y eficaz del sistema.
7. *Elaboración de la documentación.* Creación simultánea de documentación detallada que abarque los objetivos, diseño, implementación y resultados del proyecto. Incluye descripciones de decisiones clave y métodos utilizados.

Plan de trabajo

El plan de trabajo se lista en las siguientes actividades a desarrollar, y se ilustran las fechas estimadas en las que se plantea desarrollarlas en cuadro 1.

1. Elaboración del marco teórico
2. Recopilación de datos necesarios para la CNN y el sintonizador PSO.
3. Consolidación de *datasets* para uso en sintonizador
4. Desarrollo del sintonizador PSO
5. Implementación de funcionalidades adicionales / secundarias.
6. Consolidación de *datasets* para reconocimiento facial
7. Desarrollo de una CNN para el reconocimiento facial.
8. Sintonización de la CNN de reconocimiento facial para consolidación del sintonizador.

9. Perfeccionamiento del proyecto.
10. Estimación de esfuerzo y asignación de tareas.
11. Configuración de entornos de desarrollo.
12. Implementación de pruebas unitarias y de integración.
13. Pruebas de funcionamiento
14. Evaluación del proyecto.
15. Elaboración de la documentación.
16. Preparación de avances, entrega y presentación.

Actividad	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	x	x	x	x													
2			x	x													
3				x	x							x					
4					x	x	x	x	x	x							
5													x	x	x		
6				x	x	x					x						
7				x	x	x	x										
8											x	x	x				
9														x	x		
10	x																
11		x	x														
12					x	x	x	x	x	x	x						
13											x	x					
14															x	x	
15		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
16						x					x					x	x
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17

Cuadro 1: Cronograma

Contenido

El documento está estructurado en las siguientes partes

1. Introducción
2. Estado de arte
3. Soluciones
4. Conclusiones y trabajo futuro

Parte I

Estado de Arte

Capítulo 1

Sintonización de hiperparámetros

Resumen

La sintonización de hiperparámetros es crucial para el rendimiento de las redes neuronales convolucionales (CNN). Este artículo revisa el uso de algoritmos bioinspirados, especialmente el de optimización por enjambre de partículas (PSO), en la optimización de hiperparámetros de CNN. Se analizan enfoques como los algoritmos genéticos y la optimización bayesiana, destacando sus ventajas y desafíos. Los resultados muestran que PSO es particularmente efectivo para esta tarea, gracias a su adaptabilidad y capacidad para evitar mínimos locales. Este trabajo resalta la importancia de seguir investigando estas técnicas para mejorar el rendimiento de las CNN en diversas aplicaciones.

1.1. Introducción

La sintonización de hiperparámetros desempeña un papel crucial en el rendimiento de las redes neuronales convolucionales (CNN, por sus siglas en inglés). Los hiperparámetros adecuadamente ajustados impactan en la capacidad de aprendizaje, la generalización y la eficiencia computacional de estos modelos. Este documento revisa la aplicación de sintonizadores bioinspirados, en particular el algoritmo de optimización por enjambre de partículas (PSO, por sus siglas en inglés), para abordar el desafío de encontrar configuraciones óptimas en el espacio de hiperparámetros de las CNN.

1.2. Metodología

La investigación se enfocó en abordar la relevancia y la importancia de la sintonización de hiperparámetros en las redes neuronales convolucionales (CNN), centrándose especialmente en la aplicación de sintonizadores bioinspirados, con énfasis en optimización por enjambre de partículas (PSO). El proceso de investigación se llevó a cabo en varias etapas cruciales:

Búsqueda de artículos por palabras clave

- Se realizó una búsqueda de artículos utilizando focos clave relevantes en bases de datos académicas como IEEE Xplore, Google Scholar y ScienceDirect.
- La ecuación de búsqueda fue $(neuralnetwork)((hyperparametertuning|naturalcomputing|geneticalgorit$
- La fecha límite fue 2015

- Los temas claves incluyeron 'definición de hiperparámetros claves', 'propuesta de representación' y 'búsqueda de solución'.

El resultado fue un conjunto de artículos que después de su debida revisión de resúmenes, introducciones y conclusiones se seleccionaron once artículos relacionados con la temática.

Filtrado de artículos Para refinar la selección, se llevó a cabo una revisión inicial de los resúmenes, introducciones y conclusiones de los 11 artículos identificados, reduciendo la lista a 7 artículos que presentaban enfoques relevantes sobre la sintonización de hiperparámetros en CNNs. Posteriormente, se procedió a una lectura completa de los 7 artículos restantes, seleccionando finalmente 4 artículos por su validez y relevancia.

1.3. Resultados

En este apartado se exploran los hiperparámetros clave, algoritmos y herramientas relevantes para la sintonización de redes neuronales convolucionales (CNN).

1.3.1. Hiperparámetros relevantes

Los hiperparámetros son variables que controlan el comportamiento y la estructura de una red neuronal convolucional. A continuación, se destacan algunos de los más relevantes:

- *Número de capas y neuronas*: Define la profundidad y la complejidad de la arquitectura de la CNN. Un mayor número de capas y neuronas puede aumentar la capacidad del modelo para aprender características complejas, pero también aumenta el riesgo de sobreajuste. En el caso de CNN se debe definir el número de capas convolucionales y el número de capas densas. [15].
- *Número de Filtros*: En cada capa convolucional, el número de filtros determina cuántas características se extraerán de la imagen[15].
- *Kernel Size*: Tamaño del filtro utilizado en las capas convolucionales para la extracción de características[15].
- *Stride y Padding*: Parámetros que afectan cómo los filtros convolucionales se mueven sobre la imagen y cómo manejan los bordes de la imagen.
- *Función de Activación*: Define la salida de una neurona dada una entrada, facilitando la no linealidad en el modelo. Funciones comunes incluyen ReLU, Sigmoid y Tanh [16].
- *Tasa de aprendizaje*: Determina la rapidez con la que se actualizan los pesos de la red durante el entrenamiento. Una tasa de aprendizaje alta puede acelerar el entrenamiento, pero puede llevar a que el modelo no converja, mientras que una tasa de aprendizaje baja puede resultar en un entrenamiento extremadamente lento [16].
- *Tamaño del lote (Batch size)*: Especifica el número de ejemplos de entrenamiento utilizados en cada iteración de optimización. Los tamaños de lote pequeños pueden proporcionar estimaciones más precisas del gradiente, pero aumentan la variabilidad de la actualización del modelo [16].
- *Optimizador*: Algoritmo utilizado para minimizar la función de pérdida ajustando los pesos de la red. Ejemplos comunes incluyen SGD, Adam y RMSprop [16].
- *Momentum*: Parámetro del optimizador que ayuda a acelerar el entrenamiento al considerar la dirección previa del gradiente.

- *Dropout Rate*: Proporción de neuronas que se omiten durante el entrenamiento para prevenir el sobreajuste.

1.3.2. Algoritmos para la optimización de hiperparámetros

Existen varios enfoques para la optimización de hiperparámetros, cada uno con sus ventajas y desafíos:

- *Grid Search y Random Search*:
  - *Grid Search* explora exhaustivamente el espacio de hiperparámetros probando todas las combinaciones posibles dentro de un rango predefinido. Aunque garantiza encontrar la mejor combinación en el rango definido, puede ser computacionalmente costoso [16].
  - *Random Search* selecciona combinaciones aleatorias de hiperparámetros. Aunque menos exhaustivo, puede ser más eficiente computacionalmente y, en muchos casos, encontrar buenas combinaciones más rápidamente [16].
- *Optimización Bayesiana*: Utiliza modelos probabilísticos para dirigir la búsqueda hacia áreas prometedoras del espacio de hiperparámetros. Se basa en el teorema de Bayes para actualizar la probabilidad de una hipótesis a medida que se obtiene nueva información. Métodos como BOHB (*Bayesian Optimization and Hyperband*) combinan la optimización bayesiana con técnicas de reducción de recursos [15].
- *Algoritmos Genéticos*: Inspirados en la selección natural, estos algoritmos evolucionan una población de soluciones para encontrar la configuración óptima. Utilizan operadores genéticos como la selección, cruce y mutación para explorar el espacio de búsqueda. Son efectivos en problemas con espacios de búsqueda grandes y no lineales [6].
- *Optimización por Enjambre de Partículas (PSO)*: Modela la búsqueda de soluciones como el comportamiento de una bandada de pájaros. Cada "partícula" en el enjambre representa una solución potencial y se mueve en el espacio de búsqueda influenciada por su mejor posición conocida y la mejor posición conocida del enjambre. Es especialmente útil para evitar mínimos locales y ajustar hiperparámetros dinámicamente durante el entrenamiento [8].

1.3.3. Herramientas y *frameworks*

Para implementar y evaluar la sintonización de hiperparámetros en CNNs, se utilizan diversas herramientas y *frameworks*:

- *TensorFlow*: Un *framework* de código abierto desarrollado por `Google`, ampliamente utilizado para el desarrollo y entrenamiento de modelos de aprendizaje profundo. `TensorFlow` proporciona APIs de alto nivel como `Keras`, que simplifican la construcción de modelos y la experimentación con hiperparámetros. Además, TensorFlow incluye herramientas como `TensorBoard` para la visualización del proceso de entrenamiento [16].
- *Keras Tuner*: Una biblioteca de optimización de hiperparámetros para `Keras`. `Keras Tuner` facilita la búsqueda de los mejores hiperparámetros mediante la implementación de técnicas como la búsqueda aleatoria, la búsqueda en hiperbandas y la optimización bayesiana. Ofrece una integración sencilla con modelos construidos en Keras, permitiendo experimentar con diferentes configuraciones de manera eficiente.

- *PyTorch*: Desarrollado por *Facebook*, es otro *framework* popular para el aprendizaje profundo. `PyTorch` es conocido por su flexibilidad y facilidad de uso, permitiendo una integración sencilla con herramientas de optimización de hiperparámetros. Su diseño dinámico de gráficos computacionales facilita la experimentación y el ajuste fino de modelos [16].
- *Scikit-learn*: Biblioteca en `Python` que ofrece herramientas simples y eficientes para la minería de datos y el análisis de datos, incluyendo funcionalidades para la evaluación de modelos. Aunque es más conocida por sus aplicaciones en aprendizaje automático tradicional, `Scikit-learn` incluye módulos para la optimización de hiperparámetros que pueden integrarse con *frameworks* de aprendizaje profundo [16].
- *Hyperopt*: Una biblioteca de optimización de hiperparámetros en `Python` que utiliza la búsqueda aleatoria y la optimización bayesiana. `Hyperopt` permite definir espacios de búsqueda complejos y aplicar técnicas avanzadas de optimización para encontrar combinaciones óptimas de hiperparámetros. Es especialmente útil para problemas con grandes espacios de búsqueda y múltiples hiperparámetros [16].

Dejé únicamente los artículos de computación natural. La información de los otros ya estaba incluída en el resumen.

1.3.4. Sintonización con computación natural

En esta sección se sintetiza el análisis de los artículos enfocados en el desarrollo de un sintonizador de hiperparámetros para redes neuronales (NN) basado en los enfoques de computación natural.

Automating Configuration of CNN Hyperparameters Using Genetic Algorithm [6] Este artículo propone el uso de algoritmos genéticos para la optimización automática de arquitecturas de CNN en tareas de clasificación de imágenes. La metodología incluye la representación de cromosomas para codificar la estructura de la red y el uso de operadores genéticos adaptativos y estrategias de selección que favorecen la diversidad en las primeras generaciones y la convergencia en las últimas. Los puntos clave de este trabajo son

- *Representación de cromosomas*: Utilizar una codificación eficiente para representar diferentes arquitecturas de CNN.
- *Operadores genéticos*: Implementar operadores de cruce y mutación adaptativos que permitan explorar y explotar eficientemente el espacio de búsqueda.
- *Selección y evolución*: Adoptar estrategias de selección que mantengan la diversidad en las generaciones iniciales y promuevan la convergencia en las generaciones finales.

Particle Swarm Optimization for Automatically Evolving CNNs [8] Este artículo utiliza la optimización por enjambre de partículas (PSO) para la evolución automática de arquitecturas de CNNs. La metodología combina la optimización y el entrenamiento del modelo en paralelo, permitiendo ajustar la estructura de la red durante el proceso de aprendizaje. Los puntos claves de este trabajo son:

- *Adaptabilidad dinámica*: Implementar PSO para ajustar los hiperparámetros dinámicamente durante el entrenamiento.

- *Paralelismo en optimización*: Ejecutar la optimización de hiperparámetros en paralelo con el entrenamiento del modelo.
- *Superación de óptimos locales*: Utilizar las capacidades de PSO para evitar mínimos locales y explorar eficientemente el espacio de búsqueda.

1.4. Conclusiones

La revisión llevada a cabo señala los desafíos y beneficios de la sintonización de hiperparámetros en CNN. Se identifican como principales desafíos la alta dimensionalidad y la no convexidad del espacio de búsqueda, mientras que los beneficios incluyen mejoras significativas en el rendimiento y la eficiencia computacional de los modelos sintonizados automáticamente.

- La búsqueda continua de nuevas alternativas y los avances en sintonización de hiperparámetros ha enriquecido la comprensión y ha permitido perfeccionar las estrategias aplicadas, consolidando a la propuesta de uso de algoritmos bioinspirados en la vanguardia de la investigación en optimización de modelos de aprendizaje profundo.
- La importancia de abordar la complejidad de la sintonización de hiperparámetros en CNN se destaca, subrayando la influencia crítica de hiperparámetros clave de arquitectura, de optimización y de regularización.
- El optimizador de enjambre de partículas (PSO) es uno de los más prometedores como sintonizador bioinspirado para la optimización de hiperparámetros en redes neuronales convolucionales (CNN).

Capítulo 2

Reconocimiento facial

Resumen

El reconocimiento facial ha experimentado un notable crecimiento en los últimos años, impulsado por avances significativos en algoritmos de aprendizaje automático y la disponibilidad de grandes conjuntos de datos etiquetados. Este documento revisa los desarrollos recientes en el reconocimiento facial, incluyendo técnicas tradicionales y modernas, bases de datos utilizadas, y arquitecturas de modelos destacables. Además, se discuten los desafíos actuales y se proponen direcciones futuras para la investigación.

2.1. Introducción

El reconocimiento facial ha experimentado un notable crecimiento en los últimos años, impulsado por avances significativos en algoritmos de aprendizaje automático y la disponibilidad de grandes conjuntos de datos etiquetados. Este problema se centra en la identificación o verificación de individuos basada en características faciales únicas extraídas de diferentes fuentes de datos, como vídeos, imágenes y datos infrarrojos.

El origen del reconocimiento facial se remonta a los años 60, donde pioneros como Woodrow Wilson Bledsoe sentaron las bases para investigaciones y desarrollos posteriores en este campo. Desde entonces, se ha convertido en un área de estudio interdisciplinaria que combina conocimientos de visión por computadora, aprendizaje automático, procesamiento de imágenes y datos biométricos. [2]

El reconocimiento facial desempeña un papel crucial en una variedad de aplicaciones, ya que su capacidad para identificar personas de manera rápida y precisa lo hace invaluable en entornos donde se requiere verificación de identidad. A continuación, se enumeran algunas de estas aplicaciones:

1. *Autenticación biométrica:* en aplicaciones de autenticación, como el desbloqueo de dispositivos móviles y la verificación de identidad en sistemas bancarios.
2. *Marketing y análisis de audiencia:* en la industria del marketing para analizar la demografía de audiencias y personalizar la publicidad.
3. *Seguridad y vigilancia:* en sistemas de seguridad para controlar el acceso a instalaciones y para la vigilancia en espacios públicos.
4. *Gestión de asistencia:* en entornos educativos y laborales para el registro de asistencia automático y la gestión de horarios.[2]

El reconocimiento facial enfrenta diversos desafíos debido a la variabilidad en las condiciones de captura de las imágenes, como cambios de pose, iluminación, expresión facial, edad y etnia. Estos factores dificultan la obtención de resultados precisos en aplicaciones del mundo real. Además, la presencia de accesorios faciales, como gafas, *piercings* y, desde la pandemia, el uso de tapabocas, plantea desafíos adicionales para los sistemas de reconocimiento facial tradicionales.

El propósito de este trabajo es revisar los avances recientes en el reconocimiento facial, identificar los desafíos actuales y proponer futuras líneas de investigación. Las preguntas de investigación abordadas incluyen: ¿Cuáles son las técnicas más efectivas para el reconocimiento facial en condiciones variadas? ¿Qué bases de datos están disponibles? ¿Qué arquitecturas de modelos proporcionan los mejores resultados?

2.2. Metodología

La metodología utilizada en este estudio se basa en una revisión de la literatura existente, así como el estudio de experimentos realizados para evaluar diferentes técnicas y modelos de reconocimiento facial. La metodología se puede dividir en las siguientes etapas: búsqueda de recursos, selección de recursos pertinentes, revisión de recursos seleccionados y síntesis de hallazgos.

Esta metodología permitió una evaluación de las técnicas y modelos de reconocimiento facial, proporcionando una base sólida para futuras investigaciones y desarrollos en este campo.

Búsqueda de artículos por palabra clave y filtrado

- Se realizó una búsqueda de artículos utilizando focos clave relevantes en bases de datos académicas como IEEE Xplore, Google Scholar y ScienceDirect.
- La ecuación de búsqueda fue $(face|imagen)recognition(Databases|(Deep\,Learning)|Keras)$
- La fecha límite fue 2015
- Los temas claves incluyeron 'bases de datos utilizadas en reconocimiento facial', 'arquitecturas utilizadas en reconocimiento facial' y 'teoría de reconocimiento facial general'.

El resultado fue un conjunto de artículos que después de su debida revisión de resúmenes, introducciones y conclusiones se seleccionaron seis artículos relacionados con la temática.

2.3. Resultados

2.3.1. Tecnologías

En los últimos años, se han logrado avances significativos en el campo del reconocimiento facial, impulsados por el uso de técnicas de aprendizaje profundo y los grandes conjuntos de datos disponibles.

Las técnicas tradicionales incluyen métodos basados en características y métodos basados en modelos. Los primeros se centran en la extracción de características locales, mientras que los segundos modelan la apariencia global de la cara. Estos enfoques tradicionales, aunque útiles en ciertos contextos, a menudo carecen de la robustez necesaria para enfrentar la variabilidad en condiciones del mundo real.[2]

En contraste, el aprendizaje profundo ha revolucionado el campo del reconocimiento facial. Las redes neuronales convolucionales (CNN) han demostrado ser especialmente efectivas para extraer características discriminatorias de las imágenes faciales, lo que permite una mayor precisión y robustez en el reconocimiento facial, incluso en condiciones desafiantes [5]. Las CNN aprenden jerarquías de características directamente de las imágenes de entrada, mejorando así el rendimiento en comparación con los métodos tradicionales.

Las arquitecturas de CNN, como VGG, ResNet y sus variantes, han sido ampliamente adoptadas en el reconocimiento facial. Estas redes utilizan capas convolucionales y de agrupamiento para capturar características faciales de manera eficiente y robusta.[5]

2.3.2. Bases de datos

La construcción de bases de datos diversificadas ha sido fundamental para el avance del reconocimiento facial. Estas bases de datos contienen imágenes faciales de alta calidad y con anotaciones precisas, lo que permite un entrenamiento efectivo de los modelos basados en aprendizaje profundo. Algunos avances notables incluyen:

1. *Construcción de bases de datos diversificadas.* Se ha avanzado en la construcción de bases de datos grandes y diversas que contienen imágenes faciales de alta calidad y con anotaciones precisas. Estas bases de datos son fundamentales para el entrenamiento efectivo de los modelos de reconocimiento facial basados en aprendizaje profundo, ya que proporcionan una amplia variedad de datos para aprender patrones discriminatorios.
2. *Selección de identidades representativas.* Actualmente, se puede acceder a información y datos de diversas celebridades y figuras públicas gracias al internet, lo que facilita la construcción de bases de datos. Garantizando la disponibilidad de imágenes en la web y evita problemas de privacidad asociados con datos de personas comunes [11].
3. *Filtrado y mejora de la calidad de los datos:* Se han propuesto técnicas para filtrar y mejorar la calidad de los datos en las bases de datos faciales. Como por ejemplo, la aplicación de un clasificador automático para eliminar imágenes erróneas y el filtrado manual con la ayuda de anotadores humanos [1]. Esto permite que las bases de datos contengan imágenes de alta calidad y sean adecuadas para el entrenamiento de modelos de reconocimiento facial.
4. *Fusión de bases de datos.* En algunos casos, se han fusionado múltiples bases de datos para aumentar la diversidad y representatividad de los datos. Esta técnica ayuda a abordar los problemas de sesgo y datos limitados en conjuntos de datos individuales, lo que resulta en modelos de reconocimiento facial más robustos y generalizables.

Entre las bases de datos que más se destacan podemos encontrar las presentes en la tabla 2.1

Cuadro 2.1: Resumen de conjuntos de datos de imágenes faciales

Nombre	**# Individuos**	**# Imágenes**	**Descripción**
AgeDB	570	16,516	Imágenes recolectadas manualmente con rango de edad de 1 a 101. Todas las imágenes están tomadas de un entorno no controlado con diferentes poses, iluminación y ruido.
Large Age-Gap (LAG)	1,010	3,828	Un conjunto de datos con imágenes de personas con una gran diferencia de edad.
CAF	4,668	313,986	Es un conjunto de datos libre de ruido que contiene imágenes recolectadas de Google.
CAFR	25k	1,446,500	Todas las imágenes están anotadas con identidad, género, edad y raza. El rango de edad es de 1 a 99 y se divide en 7 fases de edad.
IJB	11,755	500,000	*Illuminating Jet Blackbox* las imágenes contemplan variaciones de luz, poses y expresiones.
LFW	5,749	13,233	Imágenes con variación en pose, etnia, iluminación, edad, expresión, fondo, género, ropa, peinados, cámara, calidad, saturación de color y otros parámetros.
CPLFW	5,749	11,652	La diferencia de pose y el número de imágenes son más equilibrados que LFW.
Trillion-Pairs	5.7k	274k	Principalmente utilizado para pruebas. Se divide en dos partes: ELFW, DELFW. ELFW tiene imágenes de celebridades de la lista de LFW. DELFW distractores para ELFW.
IMDb-Face	59K	1.7M	Conjunto de datos limpio recolectado de capturas de pantalla y carteles de películas.
MS1M-DeepGlint	86,876	3,923,399	Rostro alineado a gran escala para entrenamiento. Incluye celebridades.
Asian-DeepGlint	93,979	2,830,146	Rostro alineado a gran escala para entrenamiento.
GANFaces-500k	10k	500k	Conjunto de datos sintético utilizado principalmente para entrenamiento.
GANFaces-5M	10k	5,000k	Conjunto de datos sintético utilizado principalmente para entrenamiento.
CelebFaces	5,436	87,628	Imágenes de celebridades recolectadas de la web utilizadas para entrenamiento.
CASIA-WebFace	10,575	494,414	Contiene imágenes de celebridades que nacieron entre 1940 y 2014 y se utiliza principalmente para entrenamiento.
VGG Face	2,622	2.6M	Un gran conjunto de datos de imágenes públicas disponibles para entrenamiento.
VGG Face2	9.131	3.31M	Un gran conjunto de datos para entrenamiento con grandes variaciones en pose, edad, iluminación, etnia y profesión.
MegaFace	690k	IM	Se utiliza como galería de imágenes.
MS-Celeb-IM	100k	10M	Imágenes de celebridades, principalmente para conjuntos de entrenamiento.
RMFRD	525	95k	Dos tipos de imágenes para la misma persona; con máscara y sin máscara.
SMFRD	10k	500k	Uso de un software para crear automáticamente caras con máscara en conjuntos de datos faciales populares.
KomNET	-	39.6k	Las imágenes se recolectan de 3 fuentes diferentes: cámara del teléfono, cámara digital y redes sociales sin considerar la iluminación, el bigote, la barba, el fondo, el corte de pelo, la expresión y las gafas cubiertas de la cabeza.

Cuadro 2.2: Bases de datos para reconocimiento facial

Entre estas bases de datos se destaca `VGGFace2`, una base de datos grande y diversa que contiene imágenes faciales de alta calidad de una amplia variedad de celebridades. Es muy utilizada para entrenar y evaluar modelos de reconocimiento facial en entornos del mundo real [4, 7, 1].

2.3.3. Técnicas y arquitecturas

Las siguientes son algunas de las técnicas que han tenido impacto en la evolución del reconocimiento facial.

1. *Aprendizaje profundo.* El aprendizaje profundo ha revolucionado el campo del reconocimiento facial al permitir que los modelos aprendan jerarquías de características directamente de los datos de entrada, mejorando así el rendimiento en comparación con los métodos tradicionales.
2. *Redes neuronales convolucionales.* Las CNN han demostrado ser especialmente efectivas para extraer características discriminatorias de las imágenes faciales, lo que permite una mayor precisión y robustez en el reconocimiento facial, incluso en condiciones desafiantes [5].
3. *Optimización de hiperparámetros.* Se ha abordado la optimización de hiperparámetros en el contexto del reconocimiento facial. El uso de algoritmos de enjambre para sintonizar los hiperparámetros de una CNN puede ayudar a mejorar la generalización y el rendimiento del modelo en diversas condiciones.
4. *Transferencia de aprendizaje.* Se ha hecho hincapié en el uso de técnicas de transferencia de aprendizaje para el reconocimiento facial. Esto implica aprovechar los modelos preentrenados en conjuntos de datos grandes, como `VGGFace2`, y ajustarlos para tareas específicas de reconocimiento facial [1].
5. *Fusión de modelos y técnicas de aprendizaje.* En algunos casos, se ha explorado la fusión de múltiples modelos CNN para mejorar el rendimiento del reconocimiento facial. También se han utilizado técnicas especiales métricas de aprendizaje para mejorar la comparación de características faciales y aumentar la precisión del reconocimiento [5, 2].

Las redes neuronales convolucionales han demostrado ser la arquitectura dominante en el reconocimiento facial debido a su capacidad para aprender automáticamente características discriminativas de las imágenes. Una CNN típica consiste en una pila de capas convolucionales seguidas de capas de agrupación y finalmente una capa completamente conectada para la clasificación. Se han aplicado diversas variantes de CNN que han llegado a ser muy exitosas y modelos personalizados para abordar los desafíos en reconocimiento, como la variabilidad en la pose, iluminación, expresión facial, entre otros [4, 2].

Algunos modelos que hasta el momento han llegado a obtener buenos resultados son presentados en la Tabla 2.3.

Red	**Bases de datos**	**Exactitud (%)**	**Arquitectura**
ResNet50	IJB	99.5	ResNet50 es una red convolucional profunda que utiliza bloques residuales con conexiones de salto para facilitar el entrenamiento de redes más profundas.
SqueezeNet-ResNet-50	IJB	99.6	SqueezeNet-ResNet-50 combina las arquitecturas de SqueezeNet y ResNet-50. SqueezeNet es conocida por su tamaño compacto, mientras que ResNet-50 es conocida por su profundidad y uso de bloques residuales.
ResNet-64	IJB-A	93.22	ResNet-64 es una variante de ResNet con 64 capas.
ResNet-27	LFW	99.48	ResNet-27 es una variante más pequeña de ResNet con 27 capas.
LightCNN-v29	LFW	98.98	LightCNN es una red convolucional diseñada para ser eficiente en términos de memoria y cálculo. La versión 29 tiene 29 capas.
Deep CNN	LFW	99.77	Deep CNN se refiere a una red convolucional profunda. El término "Deep"sugiere que la red tiene muchas capas, pero los detalles específicos pueden variar.
MTCNN	YTF	95	MTCNN (*Multi-task Cascaded Convolutional Networks*) es una red convolucional específica diseñada para la detección de rostros. Utiliza una cascada de redes convolucionales para detectar y alinear caras en imágenes.
ReST	LFW	93.4	ReST es un método de identificación de rostros que utiliza una red convolucional inspirada en un transformador espacial.
VGG16	300WLP	98	VGG16 es una red convolucional profunda que consta de 16 capas. Aunque es más profunda que una CNN vainilla típica, sigue el mismo patrón general de tener capas convolucionales seguidas de capas de pooling, y finalmente capas completamente conectadas para la clasificación.
NAN	YouTube Face	95.72	NAN parece ser un tipo específico de red convolucional, no se encontraron detalles específicos sobre su arquitectura.
DDRL	YTF	94.2	DDRL (*Deep Discriminative Representation Learning*) es un método de aprendizaje de representaciones que utiliza una red convolucional para codificar las entradas y un módulo de métrica de distancia para medir las similitudes.
PDA	`VGGFace2`-FP	95.32	PDA (*Pose-Discriminative Attention*) es un método que introduce múltiples ramas locales basadas en atención a diferentes escalas para enfatizar las características importantes para el reconocimiento facial.

Cuadro 2.3: Arquitecturas destacables [4]

Entre las redes se destaca VGG16 por su simplicidad y buenos resultados.

2.4. Conclusiones

El reconocimiento facial ha evolucionado significativamente gracias a los avances en algoritmos de aprendizaje automático y el acceso a grandes conjuntos de datos etiquetados. Las técnicas tradicionales, aunque útiles en ciertos contextos, han sido superadas por el aprendizaje profundo, especialmente a través de redes neuronales convolucionales (CNN), que han demostrado una mayor precisión y robustez en la extracción de características faciales.

Los desafíos actuales en el reconocimiento facial, como la variabilidad en las condiciones de captura de imágenes y la presencia de accesorios faciales, continúan siendo objeto de investigación y desarrollo. Se necesitan soluciones innovadoras para abordar estos desafíos y mejorar la precisión y eficiencia de los sistemas de reconocimiento facial en aplicaciones del mundo real.

Para el futuro, se vislumbran varias líneas de investigación prometedoras. Una de ellas es la exploración de técnicas de aprendizaje más avanzadas, como la atención y la generación de imágenes adversarias, que podrían mejorar aún más la capacidad de los modelos de reconocimiento facial para manejar condiciones desafiantes. Además, la optimización de hiperparámetros mediante algoritmos de enjambre y la exploración de nuevas arquitecturas de redes neuronales podrían conducir a mejoras significativas en el rendimiento del reconocimiento facial.

Para experimentar en el área, el par `VGGFace2` y VGG16 es adecuado. `VGGFace2` tiene un extenso conjunto de datos con una amplia variedad de identidades y condiciones de iluminación, pose y expresión facial. Además, `VGGFace2` está especialmente diseñado para tareas de reconocimiento facial, lo que lo hace ideal para evaluar el rendimiento de arquitecturas de redes neuronales convolucionales en este dominio. Por otro lado, la arquitectura VGG16 se destaca debido a su simplicidad y buen rendimiento en tareas de clasificación de imágenes, lo que la convierte en una opción sólida para aplicaciones de reconocimiento facial donde se requiere precisión y eficiencia computacional.

Parte II

Soluciones

Capítulo 3

Desarrollo del sintonizador de hiperparámetros PSOT

Resumen

En este artículo se presenta el desarrollo de un Sintonizador de Hiperparámetros utilizando el algoritmo de Optimización por Enjambre de Partículas (PSO) aplicado a redes neuronales convolucionales. Se exploraron diversas estrategias para mejorar la exploración del espacio de hiperparámetros, adaptando PSO específicamente para esta tarea. Se diseñó un *framework* modular que incluye un `PSOTuner` y diferentes `PSOHypermodels` extensibles para la búsqueda automática de buensa configuraciones en modelos de redes neuronales.

3.1. Introducción

La sintonización de hiperparámetros es un componente crítico en el desarrollo de modelos de aprendizaje automático, especialmente en el contexto de redes neuronales convolucionales (CNNs), donde los ajustes precisos pueden significar la diferencia entre un modelo mediocre y uno altamente efectivo. Este proceso implica la optimización de variables que no se aprenden directamente durante el entrenamiento del modelo, como tasas de aprendizaje, número de capas y filtros, y otras configuraciones de red.

En este trabajo, exploramos el uso del algoritmo de Optimización por Enjambre de Partículas (PSO) para la automatización de la sintonización de hiperparámetros en CNNs. PSO se destaca por su capacidad para explorar eficientemente espacios de búsqueda multidimensionales, adaptando dinámicamente la posición de partículas en un enjambre para buscar soluciones óptimas. A través de una revisión de literatura y el análisis de herramientas existentes como `Keras Tuner`, diseñamos un `PSOTuner` que integra PSO para mejorar la eficiencia y efectividad de la sintonización de hiperparámetros.

En la sección teórica, se analizan estudios previos sobre el PSO para la optimización de hiperparámetros, enfocándose en su utilidad específica en redes neuronales convolucionales. Se detalla el desarrollo del `PSOTuner`, explicando cómo se adaptó el algoritmo PSO para manejar configuraciones complejas y mejorar la exploración del modelo. El artículo también describe la implementación del `PSOTuner`, destacando la estructura modular de PSOHipermodels y el proceso experimental con *datasets* como MNIST y CIFAR10. En conjunto, el `PSOTuner` representa un avance significativo al automatizar la sintonización de hiperparámetros, optimizando tanto la precisión del modelo como el costo computacional asociado a la búsqueda manual de

configuraciones óptimas en problemas de aprendizaje automático con redes neuronales convolucionales.

3.2. Marco Teórico

En la etapa de revisión, llevamos a cabo la recopilación y el análisis de material teórico y práctico relacionado con la sintonización de hiperparámetros de redes neuronales utilizando diferentes métodos de búsqueda y herramientas específicas relacionadas. Uno de los enfoques que exploramos fue el algoritmo de Optimización por Enjambre de Partículas (PSO, por sus siglas en inglés), el cual se ha mostrado ser prometedor en la optimización automática de arquitecturas de redes neuronales convolucionales.

El artículo *"Particle Swarm Optimization for Automatically Evolving Convolutional Neural Networks for Image Classification"* [8] proporcionó un marco teórico sólido al explicar cómo adaptar el PSO para el contexto específico de la sintonización de hiperparámetros en redes neuronales convolucionales y destacar las ventajas y desventajas del PSO para la sintonización de hiperparámetros. En el trabajo se exploran diversas estrategias para representar los hiperparámetros como partículas en un enjambre, adaptando estas estrategias para mejorar la exploración del espacio de soluciones. Además, el artículo proporciona directrices sobre cómo estructurar la evaluación de la calidad de las soluciones encontradas. Entre las ventajas, se incluye la capacidad del PSO para explorar eficientemente el espacio de búsqueda multidimensional mediante la iteración y ajuste dinámico de las partículas; y entre las desventajas, incluyen su susceptibilidad a quedar atrapado en óptimos locales y su sensibilidad a la configuración de los parámetros del algoritmo.

Por otro lado, el artículo *"Hyper-Parameter Optimization: A Review of Algorithms and Applications"* [16] amplió nuestra comprensión sobre la importancia de los métodos de sintonización y optimización de hiperparámetros. El artículo subrayó la relevancia de estas técnicas en la reducción del costo computacional y temporal asociado con la búsqueda manual de configuraciones óptimas. Propuso soluciones que facilitan la selección de hiperparámetros para usuarios inexpertos, promoviendo la accesibilidad y la eficiencia en la aplicación práctica de algoritmos de aprendizaje automático.

Además, investigamos herramientas existentes como el sintonizador `Keras Tuner`, que nos proporcionó un punto de partida práctico para desarrollar nuestra solución. Este enfoque nos permitió establecer objetivos claros y medibles para la sintonización de hiperparámetros, basados en resultados previamente obtenidos en condiciones similares.

Estas fuentes fueron fundamentales para sentar las bases teóricas y prácticas de nuestra investigación, guiando el desarrollo de una solución efectiva y eficiente para la sintonización automatizada de hiperparámetros en redes neuronales convolucionales mediante PSO.

3.3. Solución

3.3.1. Diseño

Para el diseño nos inspiramos en el estándar de los sintonizadores de `Keras`, pero optamos por no integrarlo directamente debido a la necesidad de mantener el sisntionizador independiente.

El `PSOTuner` utiliza una representación de partícula donde cada partícula en el enjambre contiene una configuración única de hiperparámetros. Durante la optimización, el algoritmo PSO ajusta estas partículas basándose en la evaluación de la función objetivo, que en nuestro caso se define según el rendimiento del modelo bajo una métrica establecida con un conjunto de datos de validación. El proceso de optimización se centra en encontrar la combinación de hiperparámetros que maximice o minimice esta función objetivo, dependiendo del problema específico de optimización.

El diseño también incluye una clase de Hipermodelos (`PSOHyperModel`) que define la arquitectura base sobre la cual se aplicará la optimización. Este hipermodelo puede ser configurado con diferentes capas y configuraciones de red según las necesidades del usuario y las características del conjunto de datos.

Además, se desarrollaron clases de soporte como `ClassingDataLoader` para cargar diferentes conjuntos de datos, y `ClassingModelViewer` para visualizar los hipermodelos y los resultados obtenidos durante el proceso de sintonización.

Este diseño modular permite una fácil extensión y adaptación a diferentes problemas y tipos de modelos, asegurando que el proceso de sintonización de hiperparámetros con PSO sea robusto y efectivo en la búsqueda de configuraciones óptimas para redes neuronales.

El diseño del Sintonizador, junto con los Hipermodelos y algunos ejemplos posteriormente implementados, puede consultarse en la figura 3.1.

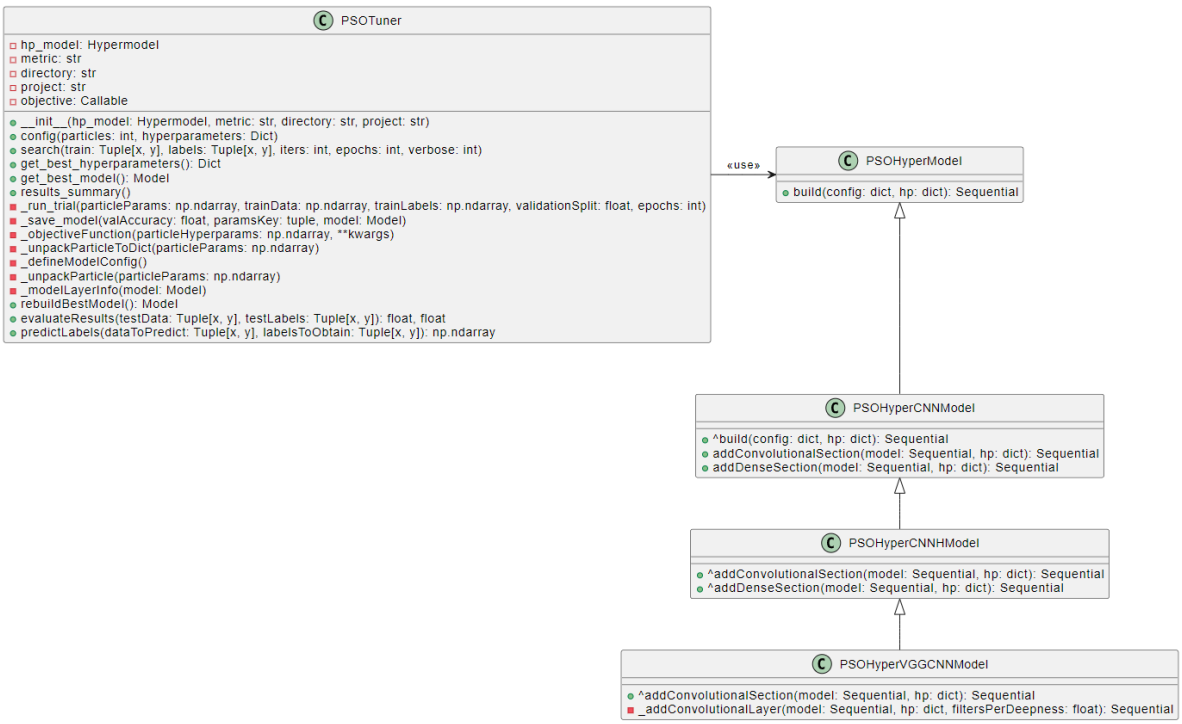


Figura 3.1: Sintonizador PSO Tuner (`PSOT`) y los diferentes hipermodelos.

`PSOTuner` Para el sintonizador `PSOTuner`, y tomando como referencia práctica `Keras Tuner`, se diseñó su estructura de manera consistente con esta herramienta conocida, compartiendo al-

gunos métodos como `search` (para buscar el modelo con mejores resultados), `run_trial` (para ejecutar los métodos de búsqueda como `Grid Search` o `Hyperband Optimization` en `Keras Tuner` y el algoritmo de PSO en nuestra solución), y `results_summary` para visualizar los datos resultantes de las ejecuciones. Sin embargo, nuestro `PSOTuner` requiere la implementación de métodos propios para optimizar las partículas, como `objectiveFunction`, encargada de determinar la calidad de las soluciones encontradas por el algoritmo, y `unpackParticleToDict` para ajustar los hiperparámetros al sintonizador como partículas en el enjambre.

Este componente se inicializa indicando el hipermodelo y la métrica que es la función objetivo que se desea optimizar sobre el conjunto de validación. Considerando el número de partículas y los hiperparámetros de PSO, el hipermodelo entregado será entrenado con un conjunto de datos provisto y sus resultados de validación se utilizarán como representante de su calidad como partícula y posición en el espacio. Las partículas interactúan en el enjambre comparando las funciones objetivo de validación alcanzadas por los hipermodelos construidos con sus configuraciones de hiperparámetros, y tras un número definido de iteraciones, se entregará al usuario el mejor resultado alcanzado junto con la configuración específica utilizada.

En relación con los coeficientes utilizados en PSO, estos parámetros controlan el comportamiento de las partículas durante la búsqueda de soluciones óptimas. Incluyen el coeficiente cognitivo que guía a cada partícula hacia su mejor posición histórica, el coeficiente social que dirige a las partículas hacia la mejor posición global encontrada por el enjambre y el coeficiente de inercia que modula la velocidad de las partículas para equilibrar exploración y explotación [8]. Estos coeficientes juegan un papel crucial en el rendimiento y la convergencia del algoritmo PSO, como se detalla en el Cuadro 3.1.

Coefficient	**Option**	**Example Value**
Cognitive Component	c1	0.5
Social Component	c2	0.3
Inertia Weight	w	0.9

Cuadro 3.1: Opciones para `PSOT`, siendo los distintos coeficientes de PSO

Es importante notar que, a mayor número de partículas o iteraciones, mayor es el número de modelos a entrenar. Por ello, se incluyó una memoria que, en caso de encontrar partículas previamente almacenadas, no entrenaría un nuevo modelo, sino que retornaría este mismo junto con los resultados obtenidos.

`PSOHypermodel` En este esquema, los modelos posibles se enmarcarán en una clase de hipermodelos que permite definir la configuración general y los hiperparámetros a ser sintonizados. Este diseño posibilita la extensión de nuevos hipermodelos, lo que abre la posibilidad de usar el sintonizador de forma personalizada y específica, o de manera general y predefinida. La sintonización dependerá del hipermodelo entregado al sintonizador.

Para cada uno de los hiperparámetros a sintonizar se debe definir los rangos determinando un límite inferior (indicando que no puede adoptar valores inferiores al indicado), un límite superior (indicando que no puede adoptar valores superiores al indicado) y la precisión (refiriéndose a con cuántos decimales debe ser tratado, diferenciando los valores enteros de los reales), como se muestra en la Cuadro 3.2.

Hyperparameter	Min Value	Max Value	Precision	Particle Example
convLayers	3	5	0	4
denseLayers	2	3	0	2
dropoutRate	0	1	2	0.53

Cuadro 3.2: Ejemplo de espacio de búsqueda de hiperparámetros para `PSOHyperModel`

Una vez definidos los hiperparámetros y sus posibles valores, estos son abstraídos a una representación como partícula dentro del enjambre [8].

Para la sintonización automática de hiperparámetros se definieron tres hipermodelos principales.

`PSOHyperCNNModel` El primer hipermodelo es una red neuronal convolucional básica, adecuada para aplicaciones generales, donde se sintonizan los siguientes hiperparámetros: el número de capas convolucionales, el número de capas densas, el número de filtros por capa, el tamaño del kernel, el tamaño del kernel del *pooling*, el número de neuronas por capa densa, la tasa de aprendizaje, el tamaño del batch y el factor de *dropout*.

Hyperparameter	Min Value	Max Value	Precision	Particle Example
convLayers	3	5	0	4
denseLayers	2	3	0	2
filters	16	64	0	32
kernelSize	3	5	0	3
*pooling*KernelSize	1	2	0	2
neurons	30	1024	0	512
learningRate	0	1	6	0.001211
batchSize	50	100	0	64
dropoutRate	0	1	2	0.53

Cuadro 3.3: Ejemplo de espacio de búsqueda de hiperparámetros para `PSOHyperCNNModel`

`PSOHyperCNNHModel` El segundo hipermodelo es una variante más compleja de red convolucional, diseñada para aplicaciones específicas que requieren configuraciones especiales para la extracción de características. Este hipermodelo utiliza heurísticos para definir los valores de los hiperparámetros a medida que se profundiza en la red: (i) el aumento progresivo del número de filtros en las capas convolucionales, para capturar características de distintas escalas; (ii) la adaptación del tamaño del kernel dependiendo del nivel de profundidad, (iii) el ajuste del tamaño del *pooling* dependiendo del nivel de profundidad; (iv) la reducción del número de neuronas por capa densa, para evitar el sobreajuste; (v) el aumento del factor de *dropout*.

Hyperparameter	Min Value	Max Value	Precision	Particle Example
convLayers	3	5	0	4
denseLayers	2	3	0	2
filters	16	64	0	32
filterGrowthFactor	0	1	4	0.5162
kernelSize	3	5	0	3
*pooling*KernelSize	1	2	0	2
deepnessIndicator	0	1	4	0.7148
neurons	30	1024	0	512
neuronDecreaseFactor	0	1	4	0.3657
learningRate	0	1	6	0.001211
batchSize	50	100	0	64
dropoutRate	0	1	2	0.53
dropoutIncreaseFactor	0	0	4	0.0178

Cuadro 3.4: Ejemplo de espacio de búsqueda de hiperparámetros para `PSOHyperCNNHModel`

`ClassingDataLoader` y `ClassingModelViewer` Después de definir el funcionamiento del `PSOT` y los requerimientos de los Hipermodelos, era necesario considerar cómo se entregarían los datos al sintonizador para entrenar y validar los hipermodelos, así como visualizar sus resultados y las configuraciones específicas resultantes.

Para ello, diseñamos dos clases adicionales. `ClassingDataLoader` se encargará de recibir los conjuntos de datos, prepararlos y distribuirlos en los conjuntos de entrenamiento y pruebas, mientras que `ClassingModelViewer` tomará los resultados del `PSOT`, construirá imágenes de los modelos resultantes y mostrará los resultados en forma de matriz de confusión. El diseño de estas clases puede consultarse en la Figura 3.2.

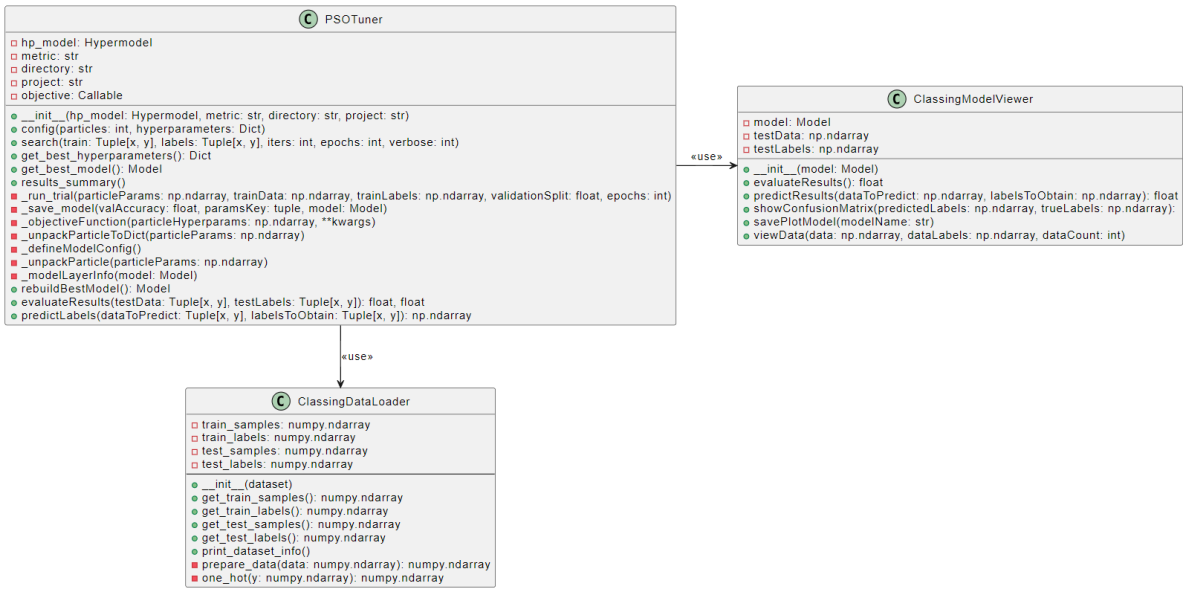


Figura 3.2: Sintonizador PSO Tuner (`PSOT`), el cargador de datos y el visualizador de arquitecturas y resultados.

3.3.2. Implementación

Finalmente, se tomó el diseño construido y, utilizando `Keras` para la construcción de hipermodelos y la gestión de las bases de datos, así como `PySwarms` para la implementación de `PSO`. Se implementó el *framework* de Sintonización de Hiperparámetros `PSOT` en Jupyter Notebook.

`PySwarms` en una biblioteca en `Python` especializada en la implementación de algoritmos de optimización por enjambre de partículas (PSO), que facilita la búsqueda y ajuste de hiperparámetros mediante la exploración inteligente del espacio de soluciones.

3.3.3. Experimentación

Para evaluar el funcionamiento de nuestro sintonizador `PSOT`, se realizaron una serie de experimentos utilizando hipermodelos distintos y dos *datasets* con diferentes niveles de complejidad. Además, se compararon los resultados obtenidos por `PSOT` con los obtenidos utilizando cuatro sintonizadores de `Keras`: `RandomSearch`, `GridSearch`, `Bayesian Optimization` e `Hyperband`, así como con un modelo sintonizado manualmente.

Para asegurar una comparación justa, se estableció el mismo número de épocas e intentos para todos los métodos e hipermodelos, así como los hiperparámetros a sintonizar definidos entre rangos, como se ve en el cuadro 3.5.

Modelos

- *Sintonización manual*: Un modelo sintonizado manualmente que sirve como referencia para comparar con métodos. El modelo consiste en dos capas convolucionales. La primera capa convolucional tiene 64 filtros de tamaño 3x3 y utiliza la función de activación ReLU. Toma imágenes de entrada con dimensiones especificadas por 'input_shape'. La segunda capa convolucional tiene 32 filtros de tamaño 3x3 y también utiliza ReLU como función de activación. Después de estas capas convolucionales, se aplica una capa de aplanado (flatten) para convertir los mapas de características en un vector unidimensional. Finalmente, hay una capa densa con 10 unidades y activación softmax, que produce las salidas de clasificación para 10 clases diferentes.
- *Sintonización con `Keras`*: Un modelo con rangos fijos establecidos con baja complejidad y poca profundidad.
- *Sintonización con `PSOT`*:
  - *Hipermodelo básico*: Una red neuronal convolucional simple con una sección convolucional, una sección densa y una salida establecida. Cada capa dentro de sus secciones tenía características idénticas, con rangos semejantes al modelo sintonizado manualmente.
  - *Hipermodelo con heurísticos*: Una arquitectura de red convolucional más compleja que implementa heurísticos definidos por expertos.

Conjuntos de datos Se utilizaron dos *datasets* con diferentes niveles de complejidad para evaluar los modelos:

- *MNIST*: Un dataset de baja complejidad que contiene 7000 imágenes de dígitos escritos a mano, en blanco y negro, con una resolución de 28x28 píxeles.

- *Distribución*:
  - *Entrenamiento*: 80 %
  - *Validación (del conjunto de entrenamiento)*: 10 %
  - *Pruebas*: 20 %

- *CIFAR10*: Un dataset de dificultad intermedia que contiene 6000 imágenes a color de 32x32 píxeles distribuidas en 10 clases diferentes.
  - *Distribución*:
    - *Entrenamiento*: 80 %
    - *Validación (del conjunto de entrenamiento)*: 10 %
    - *Pruebas*: 20 %

Métodos de Sintonización Los métodos de sintonización comparados fueron los siguientes:

- *Modelo Original*: Sintonizado a mano.
- *RandomSearch*: Utilizando el modelo similar al original
- *GridSearch*: Utilizando el modelo similar al original
- *Bayesian Optimization*: Utilizando el modelo similar al original
- *Hyperband*: Utilizando el modelo similar al original
- `PSOT` *Básico*: Utilizando el hipermodelo básico.
- `PSOT` *con Heurísticos*: Utilizando el hipermodelo con heurísticos.

Finalmente, los hiperparámetros que sintonizaría cada método y los rangos que se definirían para todos se presentan en el cuadro 3.5.

		Keras Tuners				`PSOT`		
	Manual Tuning	**Random Search**	**Hyperband**	**GridSearch**	**Bayesian**	**PSO**	**PSO**	
Hyperparameter	**Base Model**	**Base Model**	**Base Model**	**Base Model**	**Base Model**	**Basic HP**	**Heuristic HP**	**Ranges**
Convolutional Layers	X	X	X	X	X	X	X	$3 \rightarrow 5$
Dense Layers	X	X	X	X	X	X	X	$2 \rightarrow 3$
Filters		X	X	X	X	X	X	$16 \rightarrow 64$
Filter Growth Factor							X	$0 \rightarrow 1$
Kernel Size	X	X	X	X	X	X	X	$3 \rightarrow 5$
Deepness Indicator							X	$0 \rightarrow 1$
Strides								
Padding								
pooling						X	X	$1 \rightarrow 2$
Neurons		X	X	X	X	X	X	$30 \rightarrow 1024$
Neuron Decrease Factor							X	$0 \rightarrow 1$
Learning Rate		X	X	X	X	X	X	$1 \times 10^{-6} \rightarrow 1 \times 10^{-4}$
Dropout Rate						X	X	$0 \rightarrow 1$
Epochs								
Batch Size						X	X	$50 \rightarrow 100$
Activation								
Optimizer								
Kernel Initializer								
Metric	Accuracy	X	X	X	X	X	X	Accuracy

Cuadro 3.5: Comparación de hiperparámetros sintonizables por diferentes métodos

Con las condiciones establecidas, los rangos definidos, los *datasets* escogidos con dificultad incremental e hiperparámetros sintonizables, se procedió con la ejecución y recopilación de resultados.

Resultados Se midieron las siguientes métricas para cada combinación de método de sintonización e hipermodelo:

- *Train Loss*: Pérdida durante el entrenamiento.

- *Train Accuracy*: Exactitud durante el entrenamiento.
- *Val Accuracy*: Exactitud durante la validación.
- *Test Accuracy*: Exactitud durante las pruebas.

Los resultados obtenidos se resumen en el cuadro 3.6 para el dataset MNIST y en el cuadro 3.7 para CIFAR10.

Model	**Train Loss**	**Train Accuracy**	**Validation Accuracy**	**Test Accuracy**
Manual	0.0271	99.1999 %	98.68 %	98.6199 %
Random Search	0.0374	98.8866 %	98.68 %	98.3200 %
Hyperband	0.0414	98.8766 %	98.73 %	98.2500 %
GridSearch	0.0370	98.9533 %	94.81 %	98.1199 %
Bayesian	0.0191	99.4216 %	98.83 %	98.8099 %
`PSOT` Basic HP	0.0081	99.7518 %	99.3499 %	99.0899 %
`PSOT` Heuristic HP	0.0201	99.3962 %	99.4666 %	99.4599 %

Cuadro 3.6: Resultados para el dataset MNIST

Model	**Train Loss**	**Train Accuracy**	**Validation Accuracy**	**Test Accuracy**
Manual	1.0433	64.3760 %	60.54 %	59.6899 %
Random Search	1.0993	61.3259 %	58.85 %	57.4000 %
Hyperband	1.1420	59.5960 %	58.89 %	57.6499 %
GridSearch	1.0969	61.6819 %	57.95 %	56.4400 %
Bayesian	1.2269	57.3040 %	57.56 %	56.1800 %
`PSOT` Basic HP	0.0841	97.2555 %	70.6200 %	69.4400 %
`PSOT` Heuristic HP	0.4132	85.3177 %	73.4600 %	71.2999 %

Cuadro 3.7: Resultados para el dataset CIFAR10

Análisis de Resultados Los resultados muestran que nuestro `PSOT`, tanto con el hipermodelo básico como con heurísticos, obtiene un rendimiento competitivo en comparación con los métodos de sintonización de Keras, e incluso con la sintonización manual. Es notable que el `PSOT` con heurísticos consistentemente supera a otros métodos en términos de exactitud de validación y prueba.

En el dataset MNIST, el `PSOT` con heurísticos alcanza una exactitud de prueba del 99.46 %, superando al mejor método de Keras (Bayesian) que obtuvo un 98.81 %, representando una mejora de 0.65 puntos porcentuales. En el figura 3.3 se presenta la matriz de confusión.

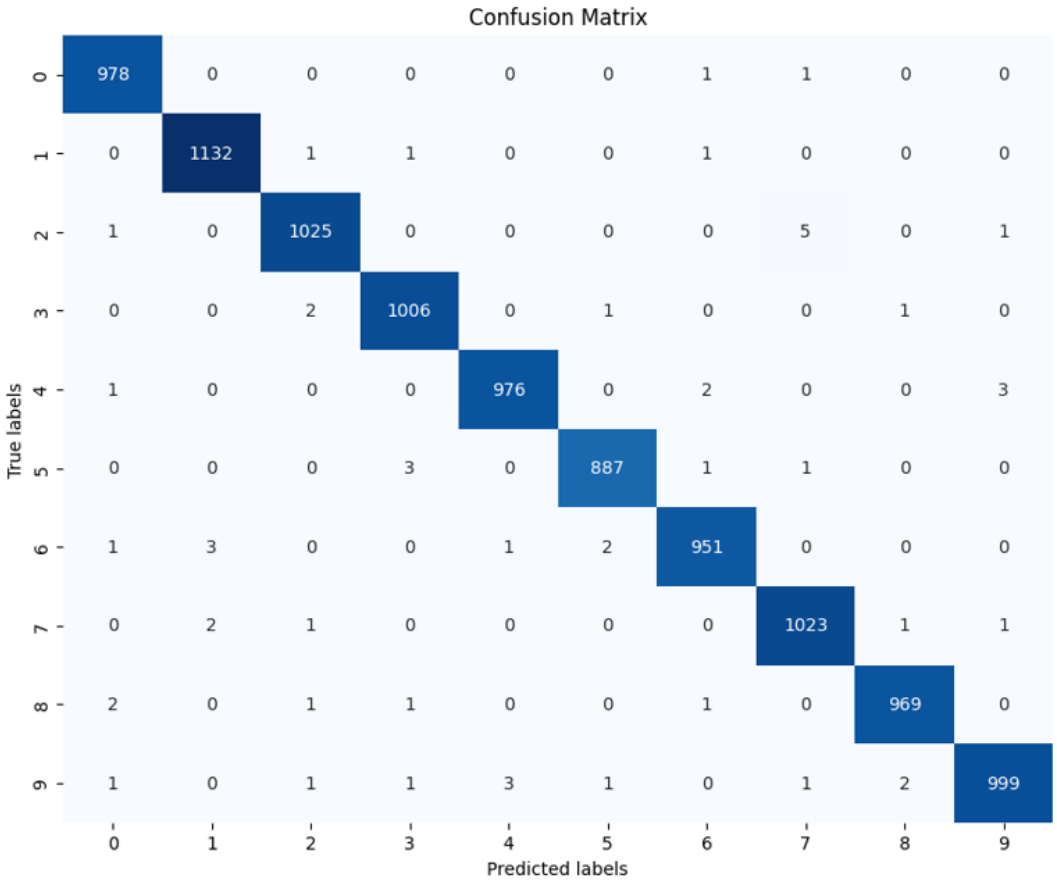


Figura 3.3: Matriz de confusión con datos de prueba para el hipermodelo con heurísticos entrenado con el dataset MNIST.

En el dataset más complejo CIFAR10, el `PSOT` con heurísticos logra una exactitud de prueba del 71.30 %, mientras que el mejor método de Keras (GridSearch) alcanza un 59.68 %, lo que representa una mejora significativa de 11.62 puntos porcentuales. Estos resultados demuestran la eficacia de nuestro enfoque en comparación con otros métodos de sintonización de hiperparámetros. En la figura 3.4 se presenta la matriz de confusión correspondiente.

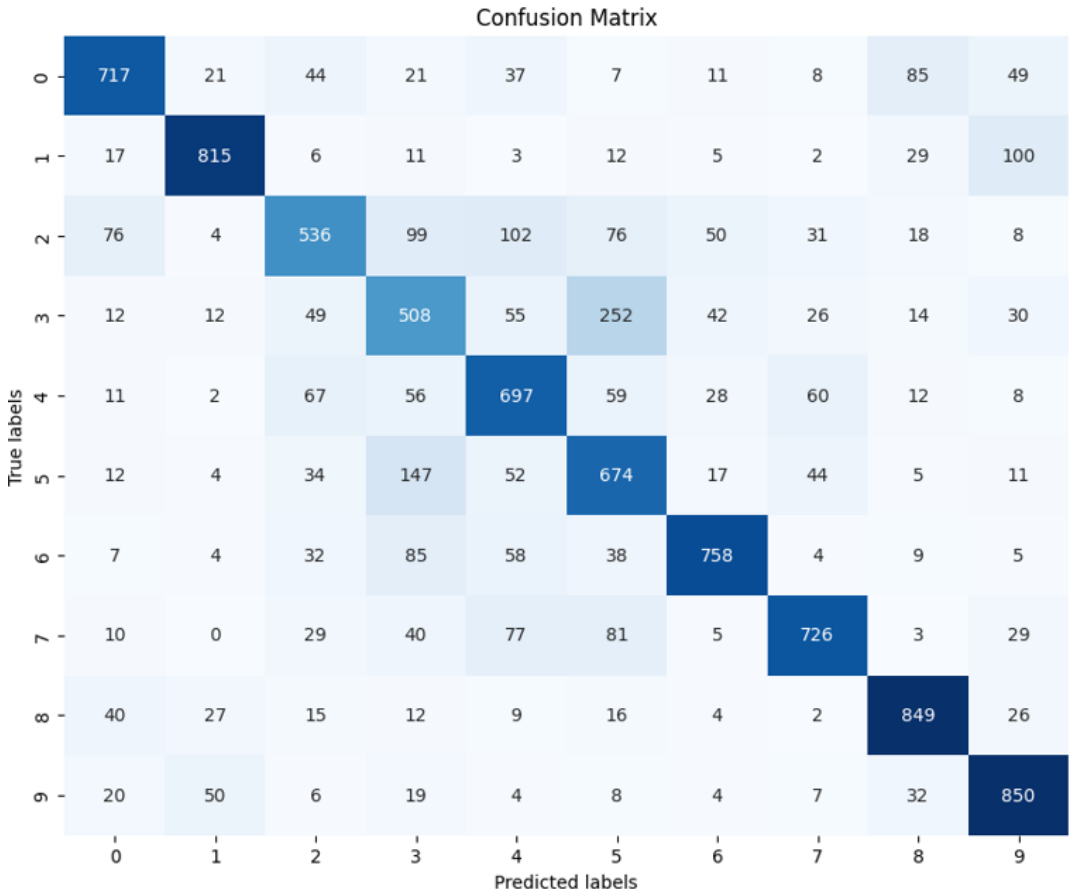


Figura 3.4: Matriz de confusión con datos de prueba para el hipermodelo con heurísticos entrenado con el dataset CIFAR10.

Se observa que los métodos basados en `PSO` pueden ofrecer una ventaja significativa en tareas de sintonización de hiperparámetros complejas, adaptándose mejor a la estructura del problema en comparación con métodos más tradicionales como `Random Search` o `GridSearch`.

3.4. Conclusiones y trabajo futuro

Conclusiones En este trabajo, se ha desarrollado un sintonizador de hiperparámetros utilizando el algoritmo PSO (Particle Swarm Optimization), haciendo uso de `Keras` para la construcción de hipermodelos y con `PySwarms` para la implementación del algoritmo de optimización. Se ha demostrado que el `PSOT` (`PSO Tuner`) es una herramienta eficaz y efectiva para la sintonización de hiperparámetros, logrando resultados competitivos y, en muchos casos, superiores a los obtenidos con los métodos de sintonización de `Keras`, como `Random Search`, `GridSearch`, `Bayesian Optimization` e `Hyperband`.

1. *Exactitud*: El `PSOT`, tanto con el hipermodelo básico como con el de heurísticos, ha mostrado un rendimiento sobresaliente. En el dataset MNIST, el `PSOT` con heurísticos alcanzó una exactitud de prueba del 99.46 %, mejorando en 0.65 puntos porcentuales

sobre el mejor método de Keras. En el dataset CIFAR10, la mejora fue aún más notable, con el `PSOT` alcanzando una exactitud de prueba del 71.30 %, superando al mejor método de Keras en 11.62 puntos porcentuales.

2. *Robustez*: La implementación de heurísticos en el hipermodelo de `PSOT` no solo mejora la exactitud, sino que también incorpora conocimiento que pudiera necesitar un aprendiz en la materia, permitiendo una mejor generalización en *datasets* de mayor complejidad, así como disponibilidad de este conocimiento para los usuarios.
3. *Flexibilidad*: El diseño del `PSOT` permite una fácil integración y personalización de diferentes arquitecturas de hipermodelos, lo que lo convierte en una herramienta versátil para una amplia gama de aplicaciones en aprendizaje automático.

Trabajo Futuro Como trabajo futuro, necesitamos experimentar con los coeficientes de PSO para analizar su impacto en el desempeño del sintonizador. Además, se podrían explorar diferentes hipermodelos más complejos y aplicar el `PSOT` a *datasets* con más categorías de clasificación, lo que permitiría evaluar su desempeño en contextos más diversos y desafiantes. Esta investigación adicional podría proporcionar una comprensión más profunda de las capacidades y limitaciones del `PSOT`, así como potenciales mejoras en su implementación.

Un desafío particular a abordar será aplicar el sintonizador a un problema de reconocimiento facial. Este reto implica enfrentarse a un dataset más complejo, como VGGFace2, y sintonizar un hipermodelo con arquitectura VGG. Además, se compararían los resultados obtenidos con la sintonización manual y con los métodos de Keras para evaluar la eficacia del sintonizador `PSOT` en un entorno más exigente. Este problema, con aplicaciones relevantes y múltiples desafíos, serviría como una prueba interesante para el sintonizador, permitiéndonos evaluar su robustez y capacidad de generalización en un escenario realista y práctico.

Capítulo 4

Desarrollo de una red neuronal convolucional para reconocimiento facial

Resumen

El reconocimiento facial ha avanzado considerablemente debido a los desarrollos en aprendizaje automático y la disponibilidad de grandes conjuntos de datos etiquetados. Las CNN han demostrado ser cruciales para mejorar la exactitud y robustez en la identificación facial, enfrentando desafíos como variaciones en iluminación, pose y accesorios faciales.

Este trabajo presenta el proceso de desarrollo de una red neuronal de reconocimiento facial adecuada para experimentar el proceso de sintonización de hiperparámetros tanto manual como automático. Para el desarrollo se seleccionó la arquitectura `VGG` y un subconjunto reducido de 11 individuos de la base de datos `VGG Face2` a la cual se adicionaron dos nuevos individuos.

Los resultados logrados fueron los siguientes: (i) 88.09 % sintonización manual; (ii) 76.70 % sintonización automática con `RandomSearch`; y (iii) 77.91 % sintonización automática con `PSO Tuner`.

4.1. Introducción

El reconocimiento facial ha experimentado un notable crecimiento en los últimos años, impulsado por avances significativos en algoritmos de aprendizaje automático y la disponibilidad de grandes conjuntos de datos etiquetados. Este problema se centra en la identificación o verificación de individuos basada en características faciales únicas extraídas de diferentes fuentes de datos, como vídeos, imágenes y datos infrarrojos.

El origen del reconocimiento facial se remonta a los años 60, donde pioneros como Woodrow Wilson Bledsoe sentaron las bases para investigaciones y desarrollos posteriores en este campo. Desde entonces, se ha convertido en un área de estudio interdisciplinaria que combina conocimientos de visión por computadora, aprendizaje automático, procesamiento de imágenes y datos biométricos. [2]

Para abordar estos desafíos, este trabajo se centra en revisar y analizar avances recientes en el campo del reconocimiento facial, con un enfoque particular en la implementación de la arquitectura `VGG16` y técnicas avanzadas como la transferencia de aprendizaje. La elección de

VGG16 se fundamenta en su eficiencia y adaptabilidad para este tipo de aplicaciones comparada con otras arquitecturas, mientras que la transferencia de aprendizaje permite mejorar significativamente la exactitud del modelo al utilizar datos previamente entrenados.

Este trabajo se centró en la implementación y entrenamiento del modelo VGG16, se exploraron diversas configuraciones para evaluar su desempeño en términos de exactitud y robustez en condiciones variables. Dando así pauta a la experimentación con variantes del modelo y la sintonización de hiperparámetros utilizando métodos manuales y encaminándonos a métodos automáticos, buscando optimizar el rendimiento del modelo.

4.2. Metodología

El proceso metodológico adoptado para este proyecto se estructuró en varias etapas clave que aseguraron una planificación y un desarrollo ordenado. En primer lugar, se procedió a la fase de investigación y recopilación de datos. Aquí, se llevó a cabo un análisis profundo de las tendencias actuales en relación con las soluciones presentes en el área de reconocimiento facial y las mejores prácticas. Este paso fue crucial para informar las decisiones estratégicas y garantizar que las soluciones propuestas estuvieran alineadas con los estándares y necesidades.

Con la información recopilada, se procedió a la etapa de diseño conceptual. Durante este período, se generaron múltiples ideas y soluciones potenciales que fueron evaluadas en función de su viabilidad técnica y capacidad para cumplir con los requisitos establecidos. Este enfoque iterativo permitió refinar y optimizar continuamente las propuestas antes de avanzar a la implementación.

Una vez finalizado el diseño conceptual, se pasó a la fase de desarrollo de los modelos. Aquí, se implementaron las soluciones seleccionadas, y nuevas soluciones surgieron en el proceso a razón de problemáticas presentes.

4.3. Marco teórico

En el ámbito del reconocimiento facial, los avances tecnológicos han sido impulsados por el desarrollo de algoritmos de aprendizaje automático y la disponibilidad de grandes conjuntos de datos etiquetados. Este progreso ha permitido a los investigadores explorar técnicas tanto tradicionales como modernas para mejorar la exactitud y robustez de los sistemas de reconocimiento facial.

Inicialmente, las técnicas tradicionales se centraron en métodos basados en características locales y globales para extraer información discriminativa de las imágenes faciales. Sin embargo, estos enfoques mostraron limitaciones significativas en la adaptabilidad a condiciones variables del entorno real [2].

El advenimiento del aprendizaje profundo marcó un hito en el campo, especialmente con la introducción de las redes neuronales convolucionales (CNN). Las CNN han demostrado ser altamente efectivas para la extracción automática de características faciales relevantes a partir de imágenes, mejorando así la exactitud y la capacidad de generalización del reconocimiento facial bajo diversas condiciones adversas [5].

Entre las arquitecturas de CNN más destacadas, se encuentran VGG, ResNet y sus variantes, las cuales han sido ampliamente adoptadas debido a su capacidad para capturar características faciales de manera eficiente y robusta [5].

VGG16 VGG16 es una arquitectura CNN reconocida que ha mostrado buen rendimiento en tareas de clasificación de imágenes, incluyendo reconocimiento facial [4]. Esta arquitectura se caracteriza por su simplicidad y eficiencia, lo cual la hace adecuada para aplicaciones donde se requiere exactitud y capacidad de manejar grandes volúmenes de datos, sin dejar atrás que entre todas las arquitecturas es una de las más planas, esto quiere decir que se caracteriza por no utilizar métodos de aprendizaje tan complejos lo cual es muy útil para el trabajo de sintonización que tenemos presente.

VGGFace2 La base de datos `VGG Face2` se destaca por su extensivo conjunto de datos que incluye una amplia diversidad de identidades y condiciones de iluminación, pose y expresión facial [4]. Esto la convierte en una opción ideal para evaluar y entrenar modelos de reconocimiento facial en entornos del mundo real, proporcionando datos ricos y variados que permiten mejorar la generalización y robustez de los modelos entrenados.

4.4. Solución

4.4.1. Desarrollo de redes

Selección de arquitectura y base de datos En la etapa inicial, se llevó a cabo un análisis detallado de las diferentes arquitecturas predominantes en el ámbito del reconocimiento facial. La meta radica en seleccionar una base de datos y una arquitectura apropiada a sintonizar, considerando simultáneamente la eficiencia computacional y la capacidad de adaptación a nuestros requisitos específicos. Llegando a la conclusión de que la red neuronal convolucional `VGG`, entrenada con la base de datos `VGG Face2`, presentaba las características más prometedoras para nuestro proyecto.

Preparación de la base de datos La base de datos `VGG Face2`, disponible en Kaggle [12], es una fuente rica en datos, que incluye información facial de más de 8000 individuos. Cada individuo está representado por una serie de fotografías que capturan sus rasgos faciales en diversas condiciones, incluyendo diferentes poses, variaciones de iluminación y perspectivas.

Sin embargo, la magnitud de esta base de datos, que actualmente pesa 40 GB, plantea desafíos computacionales significativos y supera los límites de capacidad de nuestro entorno de desarrollo. Por lo tanto, en primera instancia se hizo una reducción de este dataset. Esta versión más pequeña comprende datos de más de 500 personas, con la cantidad de fotografías por individuo variando entre 100 y 700 imágenes, versión que más adelante sería modificada con el objetivo de mejorar los resultados.

Dado que el enfoque de nuestro proyecto se centra en un ámbito académico, se decidió ampliar la base de datos con la colaboración los autores. Cada uno de nosotros contribuyó con 250 fotografías adicionales, enriqueciendo así el conjunto de datos con imágenes de personas pertenecientes a la comunidad académica. Esta ampliación de la base de datos no solo aumenta la diversidad de muestras, sino que también fortalece la relevancia de los resultados obtenidos en el contexto académico.

Entrenamiento. Primera estrategia Con la base de datos dada, se procedió a realizar en entrenamiento utilizando un modelo construido con base en la arquitectura `VGG` proporcionada en una de las referencias [14].

Un modelo `VGG` consta de dos componentes principales: la parte convolucional y las capas totalmente conectadas. Las capas totalmente conectadas están compuestas por tres capas, cada una con 4096 neuronas, seguidas de una función softmax para la clasificación. Por otro lado, la parte convolucional está estructurada en cinco bloques, donde el número de capas convolucionales puede variar en cada bloque. Cada bloque finaliza con una capa de Max *pooling*.

La característica distintiva de la arquitectura `VGG` radica en el número de capas convolucionales por bloque, el cual varía según la versión del modelo. Por ejemplo, para la `VGG11`, la configuración de capas convolucionales por bloque es 11223, refiriéndose a que en los primeros dos bloques convolucionales se tiene una capa convolucional, seguida por otros dos bloques con dos capas y finalizando con un bloque de tres capas, mientras que para la `VGG`13 es 22222, para la `VGG`16 es 2233 y para la `VGG`19 es 22444. Cada capa convolucional utiliza 64 filtros, un tamaño de kernel de 3x3 y un max *pooling* con un tamaño de 2x2.

Para esta etapa temprana se hizo una división del dataset en dos conjuntos para entrenamiento un 60 % y para pruebas 40 %. Los modelos resultantes se entrenaron durante 22 épocas, utilizando un tamaño de lote de 128 para cada iteración. Durante el proceso de entrenamiento, se monitorearon y registraron los resultados obtenidos, tanto en términos de exactitud como de pérdida. Los resultados de entrenamiento de los modelos implementados por nosotros varían en términos de exactitud entre el 1.47 % y el 1.49 %, mientras que la pérdida se mantiene en alrededor del 4.88. Estas cifras proporcionan una evaluación inicial del rendimiento de los modelos y establecen una base para las siguientes iteraciones y mejoras. Para este punto el conjunto de validación no se contempló, más adelante se integraría y tendría la relevancia respectiva.

Entrenamiento. Segunda estrategia Durante la búsqueda por obtener mejores resultados, nos encontramos con otro modelo de referencia [9], el cual utiliza la versión `VGG`16 implementada y optimizada por `Keras`. Aquí, se plantean dos enfoques: utilizar transferencia de aprendizaje o no.

Probamos inicialmente sin transferencia de aprendizaje. A pesar de registrar una ligera mejora en comparación con nuestros modelos previos, con una exactitud de entrenamiento del 1.59 % y una pérdida del 4.80. Este hallazgo sugiere que el problema no residía en la arquitectura del modelo que implementamos previamente en sí misma, sino que podría estar relacionado con otros factores, como la varianza de datos por individuo, ya que algunos pueden tener muy pocos datos relacionados.

Ante este panorama, decidimos explorar la opción de transferencia de aprendizaje de `Keras`. Esta técnica utiliza los pesos obtenidos del entrenamiento original del modelo `VGG` en el conjunto de datos ImageNet, conocido por su rendimiento sobresaliente en la competencia ILSVRC. Los resultados utilizando transferencia de aprendizaje fueron notablemente superiores a nuestras implementaciones anteriores. La exactitud en entrenamiento se elevó significativamente al 10.62 %, con una pérdida del 3.90.

Entrenamiento. Tercera estrategia Extrajimos una base de datos con 10 clases, para hacer el entrenamiento, ya que en la referencia base se usa de esta manera, como esta muestra buenos resultados queremos ceñirnos lo más posible a su propuesta, sin dejar a un lado que también nos facilitaría el análisis de los datos en un conjunto mucho más reducido que el inicial,

Modelo	Pérdida en Entrenamiento	Exactitud en Entrenamiento
VGG11	4.8230	0.0147
VGG13	4.8230	0.0145
VGG16	4.8231	0.0149
VGG19	4.8232	0.0145
Keras - `VGG16` + LT	3.9025	0.1062
Keras - `VGG16`	4.8051	0.0159

Cuadro 4.1: Resultados de pérdida y exactitud en entrenamiento para diferentes modelos.

redujimos el tamaño de lote a 64, ampliamos a 50 épocas de entrenamiento y ajustamos la tasa de aprendizaje a 0.001. Los resultados que utilizaron la transferencia de aprendizaje fueron mucho más alentadores que los obtenidos hasta el momento, con una exactitud en entrenamiento del 68.78 %, una pérdida del 0.94.

No obstante, para validar nuestros hallazgos y explorar la otra posibilidad, también realizamos pruebas sin utilizar la transferencia de aprendizaje con el modelo implementado sin `Keras`. Los resultados no fueron tan prometedores como los obtenidos con la transferencia de aprendizaje. Estos hallazgos nos llevan a especular que los desafíos encontrados podrían estar asociados, en gran medida, con el subconjunto de la base de datos utilizada, ya que este fue escogido aleatoriamente y algunos de los individuos presentes siguen teniendo mucha variabilidad en el número de datos inherentes a ellos, y probablemente la sintonización manual no sea la más adecuada, sugiriendo así la necesidad de una mayor exploración y mejora en este aspecto específico del proyecto como lo refleja el Cuadro 4.2.

Modelo	Pérdida en Entrenamiento	Exactitud en Entrenamiento
Keras - `VGG16` + LT	0.9425	0.6878
Keras - `VGG16`	2.2392	0.1855

Cuadro 4.2: Resultados de pérdida y exactitud para los modelos `Keras - VGG16` + LT y `Keras - VGG16`.

Entrenamiento. Cuarta estrategia Después de analizar los resultados anteriores, tomamos la decisión de filtrar la base de datos actual para incluir solo a los individuos que tuvieran más de 500 fotos disponibles. Se realizó la misma prueba con los dos mejores modelos hasta el momento: el modelo `VGG16` implementado por nosotros en base la referencia inicial [14], cuyos resultados son muy similares a la implementación del modelo de `Keras` sin transferencia de aprendizaje, y el modelo de `Keras` con transferencia de aprendizaje. Después del filtrado, se obtuvo un número de 49 individuos. Con el modelo implementado por nosotros sin `Keras`, logramos una exactitud del 2.82 % en entrenamiento, una pérdida del 0.0388 y una exactitud en pruebas del 2.82 %. Este resultado representa el mejor desempeño que hemos logrado con nuestra implementación basada en la referencia 1. Por otro lado, con la versión implementada por `Keras`, añadiendo transferencia de aprendizaje, conseguimos una exactitud del 35.67 % durante el entrenamiento, una pérdida del 2.30. Al observar los gráficos de convergencia, identificamos que el modelo tiende a alcanzar un punto de estabilidad en términos de pérdida antes de llegar al máximo de épocas. Por lo tanto, aumentar el número de épocas no parece ser una solución efectiva para mejorar el desempeño del modelo, como se muestra en el Cuadro 4.3.

Modelo	Pérdida en Entrenamiento	Exactitud en Entrenamiento
VGG16	3.8884	0.0282
Keras - `VGG16` + LT	2.3021	0.3561

Cuadro 4.3: Resultados de pérdida y exactitud en entrenamiento para los modelos `VGG16` y `Keras - VGG16` + LT.

Entrenamiento. Quinta estrategia Con el objetivo de mejorar nuestros resultados, decidimos seleccionar el modelo con transferencia de aprendizaje y reducir el número de clases a 10 individuos más los datos de los integrantes. Estos 10 individuos poseen más de 500 fotos cada uno, lo que nos permite contar con un conjunto de datos más representativo que el inicial atacando el problema de la variabilidad de datos por individuos. Los resultados obtenidos al aumentar el número de fotos por cada individuo fueron significativamente mejores que la versión anterior. Logramos una exactitud en entrenamiento del 69.40 %, una pérdida de 0.88. Al analizar la matriz de confusión dada con base en este entrenamiento, observamos que la mayoría de los individuos son clasificados correctamente, e incluso nosotros, los integrantes del proyecto, somos reconocidos por la red. Esto indica que la cantidad de datos por individuo son aspectos fundamentales para mejorar el rendimiento del modelo de reconocimiento facial. Los resultados con mejor desempeño hasta el momento se atribuyen a la versión de arquitectura que usa transferencia de aprendizaje, pero de igual manera la versión sin utilizar `Keras` da resultados rescatables que serán importantes en el desarrollo de este proyecto.

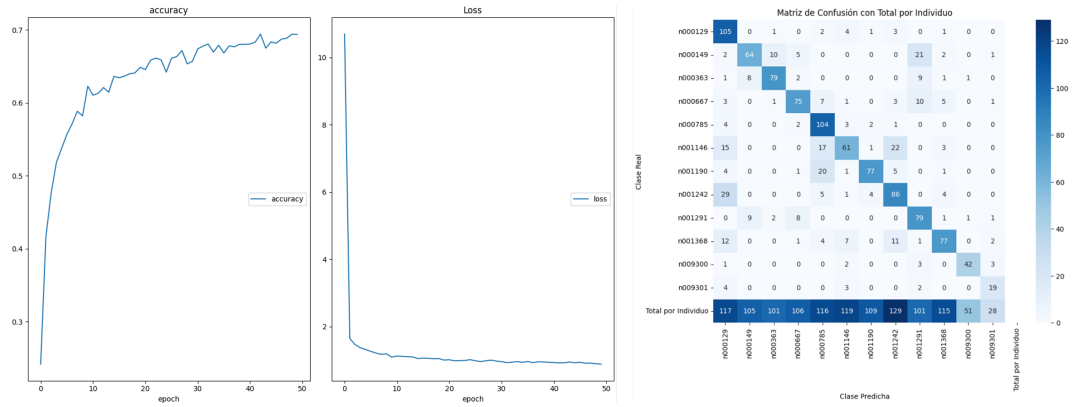


Figura 4.1: Mejores resultados en términos de exactitud. Versión con Transferencia de Aprendizaje

Desarrollo de librería Para facilitar la experimentación, se desarrolló una *librería* que implementa el modelo con mejor rendimiento hasta el momento, completamente funcional, en el constructor se logró hacer la abstracción de la mayoría de hiperparámetros posibles, teniendo en cuenta que cuando estamos usando la versión con transferencia de aprendizaje el número de estos hiperparámetros inherentes a una arquitectura convolucional se redujo severamente, aun así tenemos los suficientes para controlar la construcción de la capa densa del modelo, los respectivos a la configuración de entradas y los que monitorean el proceso de entrenamiento y evaluación.

Gracias a esta *librería* se facilita el entendimiento y se minimiza el código necesario para el desarrollo de redes. Estas ventajas se comprobaron al momento de generar diferentes casos

con el objetivo de realizar una sintonización supervisada, desde un punto de vista "experto" y así ilustrar la complejidad de dicho proceso.

Entorno de desarrollo El entorno de desarrollo actual se caracteriza por recursos limitados pero significativos. Actualmente, contamos con 100 unidades de procesamiento gracias a la suscripción de Colab Pro, lo que nos proporciona una memoria RAM de 12.7 GB, una GPU de 15 GB y un espacio de almacenamiento en disco de 201.2 GB. Estos recursos, aunque valiosos, imponen ciertas restricciones que deben ser consideradas en cada etapa del desarrollo.

Lecciones aprendidas El proceso de desarrollo de la solución para el proyecto de reconocimiento facial ha sido un viaje complejo y riguroso que ha implicado múltiples etapas de investigación, experimentación y optimización. Desde el análisis de diversas arquitecturas hasta la exploración de técnicas avanzadas como la transferencia de aprendizaje, cada paso ha sido guiado por el objetivo de mejorar la exactitud y la eficiencia del modelo. A lo largo de este proceso, hemos identificado y abordado desafíos como la limitación de recursos computacionales, la variabilidad de la base de datos y la necesidad de ajustar parámetros clave.

A medida que avanzábamos, hemos observado mejoras significativas en el rendimiento del modelo, especialmente al reducir el número de clases y aumentar la cantidad de datos disponibles. En última instancia, estos hallazgos subrayan la importancia de una metodología rigurosa y adaptable en el desarrollo de soluciones de reconocimiento facial, así como la necesidad de una comprensión profunda de los factores e hiperparámetros que influyen en su desempeño.

En el proceso de sintonización de hiperparámetros para el modelo de reconocimiento facial, se aprendió que cada hiperparámetro tiene un impacto significativo en el rendimiento del modelo: (i) el tamaño del lote fue ajustado de 128 a 64, lo que mejoró la eficiencia computacional y permitió un entrenamiento más estable; (ii) la tasa de aprendizaje, fijada en 0.001, fue esencial para equilibrar la velocidad de convergencia y la estabilidad del entrenamiento, evitando saltos bruscos y permitiendo un ajuste fino de los pesos, y (iii) el número de épocas, aumentado a 50, proporcionó al modelo suficiente tiempo para aprender sin sobreajustarse a los datos.

Por otro lado, la elección de la arquitectura `VGG16` y el uso de transferencia de aprendizaje fueron decisivos, demostrando que la reutilización de pesos preentrenados en ImageNet podía mejorar drásticamente la exactitud del modelo, alcanzando hasta un 69.40 % de exactitud en entrenamiento y una pérdida de 0.88. Estos ajustes reflejan un aprendizaje profundo de la relación entre los hiperparámetros y el rendimiento del modelo, subrayando la importancia de un enfoque sistemático y adaptativo en la sintonización para lograr mejoras significativas en la tarea de reconocimiento facial.

4.4.2. Sintonización manual

Selección de redes candidatas Para iniciar nuestro proceso de experimentación, decidimos seleccionar dos modelos que previamente habíamos planificado para nuestro estudio. El primero de ellos consiste en una librería diseñada desde cero, con el objetivo de emular la arquitectura `VGG` (Visual Geometry Group). Esta elección nos permite explorar la construcción de un modelo desde sus cimientos, comprendiendo en detalle cada capa y operación. La segunda opción que consideramos es la versión que utiliza transferencia de aprendizaje, utilizando la implementación de `VGG` mediante la biblioteca `Keras`. Optamos por esta alternativa

para contrastar el rendimiento y eficacia de nuestro modelo hecho desde cero con uno que se beneficia de conocimientos previos adquiridos en una tarea similar.

Preparación de la base de datos Para ambos modelos, utilizamos la base de datos que ya habíamos empleado en fases anteriores de nuestro estudio. Esta base de datos consta de imágenes de 12 individuos, cada uno con más de 500 fotografías. Esta elección nos permite mantener la consistencia en nuestros datos y facilita la comparación entre los modelos.

Dividimos nuestro conjunto de datos en tres partes: un conjunto de entrenamiento que representa el 70 % de los datos disponibles, un conjunto de validación que abarca el 15 %, y finalmente, un conjunto de prueba también del 15 %. La decisión de realizar la división del conjunto de datos de esta manera se basa en la necesidad de entrenar nuestros modelos con suficientes datos para aprender patrones generales, mientras que la validación y prueba nos permiten evaluar su capacidad para generalizar a datos no vistos y detectar posibles problemas de sobreajuste o subajuste. Esta división nos permite seguir un marco metodológico estándar en la evaluación de modelos de aprendizaje automático, garantizando una adecuada validación y evaluación del rendimiento de nuestros modelos.

Sintonización red sin transferencia de aprendizaje La sintonización de la red manualmente nos permite comprender en profundidad el comportamiento de nuestros modelos y afinar los hiperparámetros de manera personalizada para nuestro conjunto de datos específico. Esto nos brinda la oportunidad de ajustar la arquitectura de la red y sus parámetros de manera más precisa, maximizando su desempeño en la tarea que estamos abordando.

Para evaluar la eficacia de diferentes arquitecturas de modelos para reconocimiento facial, llevamos a cabo un proceso de experimentación. Inicialmente, implementamos cuatro variantes de la arquitectura `VGG16`, cada una con ajustes específicos en sus hiperparámetros.

El modelo 0 (64 filtros, tasa de aprendizaje de 0.0001, 4096 neuronas densas, 64 de batch size) mostró un rendimiento prometedor, con una tendencia ascendente tanto en la exactitud del entrenamiento como en la validación. A pesar de ello, observamos un ligero sobreajuste hacia el final del entrenamiento.

El modelo 1 (128 filtros, tasa de aprendizaje de 0.0005, 4096 neuronas densas, 80 de batch size) experimentó problemas de estabilidad durante el entrenamiento, con fluctuaciones erráticas en las métricas de exactitud y pérdida.

El modelo 2 (32 filtros, tasa de aprendizaje de 0.0001, 2048 neuronas densas, 64 de batch size) mostró una tendencia similar al modelo 0, pero con una brecha más notable entre la exactitud del entrenamiento y la validación, sugiriendo un mayor grado de sobreajuste.

Finalmente, el modelo 3 (96 filtros, tasa de aprendizaje de 0.0001, 3072 neuronas densas, 64 de batch size) exhibió un rendimiento sólido durante el entrenamiento, con una convergencia estable de la pérdida en ambos conjuntos de datos. Sin embargo, aún se detectaron señales de sobreajuste y fluctuaciones en la curva de pérdida.

Después de una evaluación minuciosa, los modelos 0 y 3 se destacaron como los más prometedores, a pesar de sus respectivos desafíos de sobreajuste. Estos modelos demostraron una capacidad de generalización relativamente sólida durante la etapa de pruebas, lo que los convierte en candidatos viables para su implementación en el reconocimiento facial.

A continuación, se presenta una tabla resumen con las métricas de rendimiento de cada modelo:

Experimento Pruebas	VGG_conf	Num_Filtros	Kernel_Size	Learning_Rate	Num_Neuronas	*pooling*_Type	*pooling*_Size	Num_Classes	Batch_Size	Épocas	Entrenamiento	Validación
0 89.41 %	[2, 2, 3, 3, 3]	64	(3, 3)	0.0001	4096	max	(2, 2)	12	64	30	90.24 %	89.65 %
1 -	[2, 2, 3, 3, 3]	128	(3, 3)	0.0005	4096	max	(2, 2)	12	80	30	10.79 %	10.80 %
2 -	[2, 2, 3, 3, 3]	32	(3, 3)	0.0001	2048	max	(2, 2)	12	64	30	87.51 %	88.19 %
3 86.55 %	[2, 2, 3, 3, 3]	96	(3, 3)	0.0001	3072	max	(2, 2)	12	64	30	87.51 %	89.99 %

Cuadro 4.4: Resumen de resultados de experimentación.

Este resumen proporciona una visión general de las métricas de rendimiento clave de cada modelo, incluyendo la exactitud en el entrenamiento, la validación y la prueba, así como los hiperparámetros utilizados en cada caso.

Sintonización red con transferencia de aprendizaje Luego el proceso de experimentación para la versión que implementa transferencia de aprendizaje con `Keras` se llevó a cabo con el objetivo de evaluar y comparar diferentes modelos de reconocimiento facial utilizando transferencia de aprendizaje. Para ello, se realizaron pruebas con cinco variantes del modelo, cada una con diferentes configuraciones de hiperparámetros.

En primer lugar, se definieron los conjuntos de hiperparámetros para cada modelo, incluyendo tasas de aprendizaje, tamaños de lote, épocas de entrenamiento, tasas de abandono (dropout), regularizaciones L1 y L2, y momentos del optimizador. Estos hiperparámetros fueron seleccionados con el objetivo de explorar una variedad de configuraciones y encontrar la combinación óptima que maximice el rendimiento del modelo.

El modelo 0 se configuró con una tasa de aprendizaje de 0.001, un tamaño de lote de 32, 50 épocas de entrenamiento, una tasa de abandono de 0.2, y regularizaciones L1 y L2 de 0.001, y un momentum de 0.9. La lógica detrás de esta selección fue utilizar valores moderados en la tasa de aprendizaje y el tamaño del lote, mientras se aplicaba una regularización para prevenir el sobreajuste.

El modelo 1, por otro lado, utilizó una tasa de aprendizaje de 0.0001, un tamaño de lote de 64, 40 épocas de entrenamiento, una tasa de abandono de 0.3, y regularizaciones L1 y L2 de 0.01, sin momentum. Aquí, la lógica fue reducir la tasa de aprendizaje y aumentar la regularización y la tasa de abandono para ver si esto mejoraba la estabilidad y evitaba el sobreajuste.

El modelo 2 fue configurado con una tasa de aprendizaje de 0.0005, un tamaño de lote de 32, 50 épocas de entrenamiento, una tasa de abandono de 0.2, y regularizaciones L1 y L2 de 0.0001, con un momentum de 0.5. La intención fue equilibrar el tamaño del lote con una tasa de aprendizaje más baja y aplicar una regularización mínima.

El modelo 3 utilizó una tasa de aprendizaje de 0.0005, un tamaño de lote de 64, 50 épocas de entrenamiento, una tasa de abandono de 0.2, y sin regularización L1 y L2, con un momentum de 0.95. La lógica fue aumentar el tamaño del lote y el momentum para analizar su impacto en la estabilidad y la convergencia.

Finalmente, el modelo 4 se configuró con una tasa de aprendizaje de 0.001, un tamaño de lote de 128, 70 épocas de entrenamiento, una tasa de abandono de 0.2, sin regularización L1 y L2, y sin momentum. Aquí, se probó un tamaño de lote grande con una tasa de aprendizaje más alta para evaluar su efecto en la velocidad de entrenamiento y la precisión.

Cada modelo fue entrenado y evaluado utilizando un conjunto de datos de entrenamiento y validación. Durante el entrenamiento, se monitorearon métricas clave como la exactitud y la

pérdida en ambos conjuntos de datos para analizar el comportamiento del modelo a lo largo de las épocas.

El modelo 0 mostró un comportamiento prometedor durante el entrenamiento, con una tendencia ascendente tanto en la exactitud del entrenamiento como en la de validación. Sin embargo, se observó una brecha entre estas métricas, indicando un posible sobreajuste del modelo. Por otro lado, el modelo 1 mostró un rendimiento deficiente, con una baja exactitud y estancamiento tanto en el entrenamiento como en la validación.

En contraste, los modelos 2, 3 y 4 exhibieron una tendencia ascendente tanto en la exactitud del entrenamiento como en la de validación a lo largo de las épocas. Aunque se observaron algunas fluctuaciones, estos modelos demostraron un aprendizaje efectivo y una capacidad de generalización adecuada a nuevos datos.

Después de analizar el comportamiento de los cinco modelos, se seleccionaron los modelos 3 y 4 para la etapa de evaluación final debido a su rendimiento consistente y prometedor. Ambos modelos mostraron una tendencia ascendente en la exactitud y una convergencia estable de la pérdida, lo que sugiere un aprendizaje efectivo y una capacidad de generalización adecuada.

Finalmente, se realizaron evaluaciones adicionales, incluyendo la construcción de matrices de confusión para analizar el rendimiento de clasificación de cada modelo en detalle. Estas evaluaciones proporcionaron información adicional sobre el rendimiento de los modelos en diferentes clases, lo que ayudó a tomar decisiones informadas sobre la selección del modelo final para la tarea de reconocimiento facial.

La siguiente tabla resume los hiperparámetros y métricas de rendimiento de cada modelo:

Modelo	Tasa de aprendizaje	Tamaño Batch	Epocas	Factor Dropout	L1	L2	Momentum	Entrenamiento pérdida	Entrenamiento exactitud	Validación pérdida	Validación exactitud	Pruebas exactitud
0	0.001	32	50	0.2	0.001	0.001	0.9	14.1751	50.60 %	13.9013	68.05 %	-
1	0.0001	64	40	0.3	0.01	0.01	0.0	14.3366	10.76 %	14.3401	10.80 %	-
2	0.0005	32	50	0.2	0.0001	0.0001	0.5	1.9148	78.45 %	1.6757	87.18 %	-
3	0.0005	64	50	0.2	0.0	0.0	0.95	0.2534	91.25 %	0.2792	92.13 %	88.09 %
4	0.001	128	70	0.2	0.0	0.0	0.0	0.2444	91.59 %	0.3189	90.55 %	87.21 %

Cuadro 4.5: Resumen de resultados de experimentación.

La configuración más prometedora hasta el momento fue la 3, basándose en los resultados de los modelos evaluados anteriormente. Esta fue la red eleccionada. Esta estrategia garantizó una base sólida para avanzar en el proceso de experimentación.

Aumento automático de datos El uso de generadores en el proceso de entrenamiento y evaluación de los modelos fue una decisión clave durante el desarrollo, hasta el momento siempre estuvo inmersa sigilosamente. Los generadores permiten aplicar aumentos de datos en tiempo real, como rotaciones, desplazamientos, y *zooms* aleatorios, lo que enriquece el conjunto de datos de entrenamiento y ayuda a mejorar la capacidad de generalización del modelo. No utilizar generadores podría resultar en un modelo menos robusto y con una mayor tendencia al sobreajuste, ya que estaría entrenado en un conjunto de datos menos variado.

Aumento manual de datos Finalmente, se decidió enriquecer la base de datos de entrenamiento mediante la incorporación de nuevos datos. Esta acción implicaba la inclusión de información adicional en el proceso de aprendizaje del modelo, con la expectativa de observar mejoras en su capacidad de generalización a nuevos ejemplos. Se partió de la premisa de que un modelo entrenado con una base de datos más grande tendría una mejor capacidad para

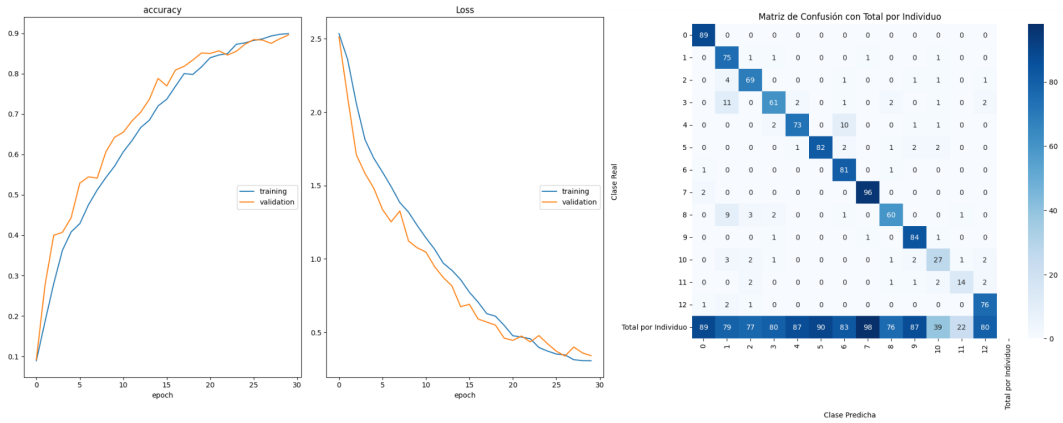


Figura 4.2: Rendimiento del modelo enriquecido

adaptarse y generalizar, lo que potencialmente conduciría a un rendimiento mejorado en la tarea de clasificación.

La evaluación del modelo después de este paso reveló un progreso constante tanto en la exactitud como en la pérdida durante el entrenamiento. Esto sugiere un aprendizaje efectivo y una capacidad de generalización a datos no vistos. Sin embargo, se observó una discrepancia entre la exactitud del entrenamiento y la de validación, indicando cierto sobreajuste del modelo a los datos de entrenamiento. Aunque se alcanzaron altos valores de exactitud, cercanos al 90 %, tanto en el entrenamiento como en la validación, las fluctuaciones en la exactitud de la validación y el ligero aumento en la pérdida hacia el final del proceso sugirieron que el modelo podría estar teniendo dificultades para generalizar completamente a nuevos datos.

Además, el análisis de la matriz de confusión proporcionó información detallada sobre el rendimiento del modelo en la clasificación de las diferentes clases. Se identificaron puntos fuertes, como una alta exactitud en la mayoría de las clases y una distribución equilibrada de los totales por individuo. Sin embargo, también se destacaron áreas de mejora, como las confusiones notables entre ciertas clases y la necesidad de aumentar la cantidad de datos para clases con pocos ejemplos.

4.4.3. Sintonización automática

La sintonización automática se realizó usando dos tecnologías `Keras Tuner RandomSearch` y el *framework* sintonizador de hiperparámetros que se estaba desarrollando paralelamente: `PSO Tuner`. La idea es validar el sintonizador en desarrollo con una tecnología ya probada.

El rango de búsqueda de hiperparámetros se seleccionó cuidadosamente en función de la experiencia previa y la literatura especializada. Se realizaron múltiples intentos para cada experimento, garantizando condiciones equivalentes en ambos casos.

Sintonización `Keras Tuner` La sintonización automática de línea base se realizada con `Keras Tuner RandomSearch` dio resultados aceptables. Sin embargo, la optimización manual demostró una superioridad evidente en términos de exactitud y eficiencia.

Métrica	Valor
Mejor exactitud de Validación	0.7789
Tiempo Total Transcurrido	00h 50m 14s
exactitud de Prueba del Mejor Modelo	76.70 %
Entorno de Ejecución	T4 GPU
Hiperparámetros del Mejor Modelo	
Capas Convolucionales Internas	1
Capas Convolucionales Externas	1
Número de Filtros	16
Tamaño del Kernel	2
Tamaño del Kernel de Capa *pooling*	3
Neuronas en Capa Densa	1024
Tasa de Aprendizaje	0.0003
Tipo de *pooling*	Max
Tamaño de Strides	2

Cuadro 4.6: Resumen de los mejores resultados e hiperparámetros del modelo sintonizado con `Keras Tuner`.

Sintonización con `PSO Tuner` Antes de usar `PSO Tuner`, nos encontramos con la necesidad de adaptar el modelo base seleccionado. Con el propósito de hacer que el modelo fuera más adaptable al entorno del sintonizador, una de las decisiones clave tomadas en este punto fue la eliminación de los generadores utilizados para la creación de datos de aumento. Sabíamos que prescindir de generadores limita la capacidad del modelo para aprender patrones más complejos y variados, lo que podría resultar en un rendimiento inferior en la generalización y en la capacidad de manejar datos de prueba que difieran significativamente del conjunto de entrenamiento.

A pesar de ello, los resultados obtenidos no fueron tan desalentadores, ya que aún mantenían un nivel aceptable de desempeño, como se refleja en las siguientes métricas:

Métrica	Valor
Entrenamiento pérdida	0.0259
Entrenamiento exactitud	99.12 %
Validación pérdida	1.3316
Validación exactitud	76.65 %

Cuadro 4.7: Resumen de los resultados obtenidos sin los generadores

A pesar de este contratiempo, logramos mejoras significativas en la exactitud del modelo con el `PSO Tuner`, haciendo pruebas limitadas por cinco épocas, cinco partículas y cinco iteraciones, como se muestra a continuación:

Hiperparámetros	Límite Inferior	Límite Superior	`VGG`
Capas Convolucionales	1	2	1
Internas	1	4	3
Filtros	32	64	46
Tamaño del kernel	2	4	3
Tamaño del kernel de capa *pooling*	2	3	3
Tamaño de strides de capa *pooling*	2	4	2
Neuronas por Capa Densa	1024	4096	2052
Ritmo de Aprendizaje	0.0001	0.00001	0.000024
Tamaño de los Lotes	0	16	4
	Exactitud en validación	80.7851 %	
	Exactitud en pruebas	77.9128 %	
	Tiempo de ejecución	22 Minutos	
	Entorno de ejecución	A100 GPU	

Cuadro 4.8: Resumen de los resultados e hiperparámetros resultantes con el PSOT.

Estos resultados evidencian que el `PSO Tuner` es una herramienta competente para la sintonización de hiperparámetros, en comparación con la sintonización en `Keras`. No obstante, aún no alcanza los niveles de exactitud y eficiencia logrados mediante la sintonización manual realizada por un experto en hiperparámetros.

Lecciones aprendidas Esta comparación resalta la importancia del conocimiento experto en la optimización de modelos de aprendizaje automático, incluso en el contexto de herramientas avanzadas de sintonización automática como el `PSO Tuner`. El conocimiento experto aporta un entendimiento profundo de cómo ajustar los hiperparámetros de manera específica para cada problema y conjunto de datos, maximizando así el desempeño del modelo. Por otro lado, herramientas como el `PSO Tuner` ofrecen la capacidad de explorar de manera más amplia y sistemática el espacio de hiperparámetros, buscando configuraciones óptimas que pueden no ser evidentes para un experto de manera manual.

Sin embargo, el éxito de la sintonización automática depende en gran medida de cómo se configure el espacio de búsqueda de los hiperparámetros del sintonizador, así como de la complejidad y la naturaleza del problema en cuestión. En casos donde se requieren exactitudes elevadas y un ajuste fino, la combinación del conocimiento experto y el uso de herramientas automatizadas como el `PSO Tuner` puede ofrecer resultados sobresalientes.

4.5. Conclusiones y trabajo futuro

Conclusiones El desarrollo de esta solución para el proyecto de reconocimiento facial ha sido un proceso meticuloso y multidimensional que ha abarcado múltiples fases de investigación, experimentación y optimización.

Desde el principio, nos enfrentamos al desafío crucial de seleccionar e implementar la arquitectura adecuada para nuestro modelo. Después de evaluar varias opciones, optamos por la red neuronal convolucional `VGG16`, potenciada con transferencia de aprendizaje desde la base de datos ImageNet.

Durante la fase inicial, exploramos diversas configuraciones y ajustes de hiperparámetros para afinar el rendimiento del modelo. Experimentamos con diferentes tamaños de lote, tasas de aprendizaje y números de épocas, observando cómo estas variables influían en la convergencia del modelo y su capacidad de generalización. Los resultados iniciales con nuestro modelo

desarrollado desde cero mostraron mejoras modestas, lo que nos llevó a investigar soluciones adicionales para abordar la variabilidad de los datos y la selección de arquitecturas.

Un hito significativo fue la expansión de nuestra base de datos mediante la inclusión de fotografías adicionales de individuos académicos, enriqueciendo así la diversidad de muestras y fortaleciendo la relevancia de nuestros resultados dentro del contexto educativo. Esta expansión no solo mejoró la variabilidad de datos, sino que también permitió entrenamientos más robustos y precisos.

Durante el proceso de experimentación, enfrentamos desafíos como el sobreajuste y la variabilidad en los datos por individuo, lo que nos llevó a ajustes continuos en la arquitectura del modelo y en los hiperparámetros seleccionados. Esto fue crucial para maximizar tanto la exactitud del entrenamiento como la capacidad de generalización del modelo a nuevos datos, aumentando nuestra comprensión sobre la sintonización manual de hiperparámetros.

La evaluación final demostró que el modelo `VGG16` con transferencia de aprendizaje fue altamente efectivo para nuestra tarea de reconocimiento facial inicial, alcanzando una exactitud del 69.40 % en entrenamiento y una pérdida mínima de 0.88. Este rendimiento validó nuestra elección de arquitectura y metodología, destacando la utilidad de utilizar recursos preentrenados para mejorar el desempeño en tareas complejas como el reconocimiento facial. No obstante, en el proceso de sintonización se logró superar dicha marca y se demostró que una sintonización adecuada puede mejorar potencialmente los resultados.

Además, implementamos una librería funcional para la configuración y entrenamiento del modelo, simplificando el proceso de experimentación y sintonización de hiperparámetros. Esta abstracción facilitó una mayor eficiencia en el desarrollo y una comprensión más profunda de los factores que afectan el rendimiento del modelo.

En el proceso de integración con el sintonizador automático, enfrentamos desafíos adicionales relacionados con la adaptación de nuestro modelo al *framework* del sintonizador para una integración fluida y eficiente. Decidimos explorar la eliminación de generadores utilizados para la creación de datos de aumento, buscando hacer el modelo más adaptable al entorno del sintonizador.

Este proyecto avanzó significativamente en términos de exactitud y eficiencia computacional, mientras contribuyó a nuestro entendimiento de la importancia de una metodología sistemática y adaptable en el desarrollo de soluciones de reconocimiento facial. Los resultados subrayan la necesidad de considerar cuidadosamente cada etapa del proceso, desde la selección de arquitectura hasta la optimización de hiperparámetros, para lograr mejoras significativas en la aplicación práctica de tecnologías de visión por computadora.

Trabajo futuro El trabajo futuro se enfocará en varias áreas clave para continuar mejorando nuestro sistema de reconocimiento facial. Uno de los principales enfoques será la exploración y adopción de técnicas avanzadas de aumento de datos y regularización para abordar el desafío del sobreajuste, especialmente en situaciones donde la variabilidad de los datos por individuo es alta. Integrar generadores y estrategias de aumento de datos específicas para mejorar la robustez y la generalización del modelo será crucial.

Además, un campo abierto a explorar es la indagación de arquitecturas más avanzadas y complejas de redes neuronales convolucionales, más allá de `VGG16`, para evaluar si podemos mejorar aún más el rendimiento del sistema. Esto incluirá investigar modelos como *ResNet*, *EfficientNet*, o arquitecturas personalizadas diseñadas específicamente para la tarea de reconocimiento facial u otros enfoques.

Otro aspecto crucial será la implementación de técnicas de sintonización automática de hiperparámetros más avanzadas y adaptativas para encontrar configuraciones óptimas de hiperparámetros de manera más eficiente y efectiva.

Finalmente, para validar la utilidad y efectividad de nuestro sistema en escenarios del mundo real, planeamos realizar pruebas en entornos diversos y con conjuntos de datos más grandes y variados. Esto nos permitirá evaluar la escalabilidad, la robustez y la fiabilidad del sistema en condiciones más desafiantes y diversas.

Conclusiones y trabajo futuro

Conclusiones

Optimizador PSO PSOT (*Particle Swarm Optimizacion Tuner*), el *framework* desarrollado basado en PSO, evidencia un rendimiento sólido en la sintonización de una amplia gama de hiperparámetros para redes neuronales. Este enfoque innovador abstrae los hiperparámetros como partículas dentro de un enjambre, permitiendo una exploración eficiente del espacio de búsqueda y guiando la selección hacia los mejores resultados.

Diseño reutilizable y extensible La diversidad de hipermodelos desarrollados permite al sintonizador adaptarse a diferentes escenarios y optimizar los hiperparámetros proporcionados por cada red. Asimismo, al independizar el sintonizador y de los hipermodelos, brindamos la oportunidad a cualquier usuario de crear un hipermodelo personalizado y sintonizarlo eficientemente.

Desarrollo evolutivo La selección de conjuntos de datos con una dificultad incremental, comenzando por MNIST, conocido por su relativa simplicidad, y avanzando a CIFAR10, que presenta desafíos más complejos, fue una estrategia que nos permitió observar cómo el rendimiento del sintonizador se adapta a problemas cada vez más difíciles, destacando su capacidad para explorar el espacio de hiperparámetros y optimizar configuraciones para los hipermodelos de redes neuronales en una variedad de contextos de aprendizaje automático. Comparativamente, PSOT logró exactitudes en validación del 99.4 % y del 73.4 % para MNIST y CIFAR10, respectivamente, superando a los resultados obtenidos para los mismos conjuntos de datos haciendo uso de los métodos de Keras Tuner, representando mejoras del 0.3 % y del 13 % aproximadamente.

Recursos computacionales Sin embargo, es importante tener en cuenta que cada método de optimización de hiperparámetros tiene sus propias fortalezas y limitaciones. Aunque PSO demostró un rendimiento prometedor en este estudio, reconocemos que su eficacia puede depender de varios factores como la configuración de los hipermodelos y sus hiperparámetros, o la complejidad del espacio de búsqueda, que generan un costo computacional elevado y un alto consumo de memoria. El gran consumo de GPU y de memoria RAM fue un factor determinante en los resultados obtenidos, por lo que, consideramos importante resaltar que para manejar mayores espacios de búsqueda, incrementar la complejidad de las soluciones y obtener mejores resultados, es indispensable poseer gran cantidad de recursos computacionales.

Limitaciones Teniendo esto en cuenta, es crucial reconocer las limitaciones de nuestro proyecto, como la limitación en la experimentación y la restricción a un número limitado de iteraciones y partículas en el proceso de PSO debido al alto consumo de recursos. Esta limitación podría afectar la generalización de nuestros resultados a otros conjuntos de datos y configuraciones experimentales dependiendo de su complejidad, por lo que es necesario ampliar este trabajo explorando la optimización del *framework*, refactorizando y evitando reprocesos.

Reconocimiento facial como reto El proceso de desarrollo de la solución de reconocimiento facial, orientado hacia la sintonización como objetivo principal y el reconocimiento facial como reto específico, ha sido una experiencia compleja y multifacética que involucró diversas fases de investigación, experimentación y optimización. La investigación inicial permitió identificar la red neuronal convolucional VGG, entrenada con la base de datos VGGFace2, como la arquitectura más adecuada para nuestras necesidades, gracias a su balance entre precisión y eficiencia computacional. La disponibilidad de recursos computacionales, aunque limitada, fue un factor crítico que influyó en cada etapa del desarrollo. La selección de una versión reducida de la base de datos VGGFace2, que aún conservaba la diversidad necesaria para un entrenamiento efectivo, fue una decisión estratégica para trabajar dentro de las restricciones impuestas por nuestro entorno de desarrollo.

Sintonización manual Durante la fase de experimentación, se implementaron y evaluaron diversas variantes del modelo VGG, incluyendo versiones con y sin transferencia de aprendizaje. Los resultados iniciales, aunque desalentadores, impulsaron la adopción de estrategias de regularización y ajustes en los hiperparámetros. Sin embargo, estas medidas no siempre produjeron las mejoras esperadas, subrayando la importancia de la calidad y representatividad de los datos de entrenamiento. El uso de la transferencia de aprendizaje demostró ser significativamente beneficioso. La implementación de VGG16 con pesos preentrenados en el conjunto de datos ImageNet proporcionó un notable incremento en la precisión, tanto en entrenamiento como en pruebas. Este hallazgo resaltó la relevancia de aprovechar conocimientos previos en la construcción de modelos de reconocimiento facial, aun así, este estudio no pasa a la etapa de sintonización debido a los pocos hiperparámetros inherentes a redes convolucionales, objetivo crucial para el sintonizador.

La posterior reducción y filtrado de la base de datos, enfocándose en individuos con una cantidad significativa de fotos, permitió obtener resultados aún más precisos. Este enfoque evidenció que la calidad y la cantidad de datos son fundamentales para el rendimiento del modelo. La incorporación de datos adicionales provenientes de los propios autores enriqueció la base de datos y fortaleció la relevancia de los resultados en un contexto académico.

Adecuando el sintonizador y el modelo para el reto La experimentación supervisada, guiada por un enfoque experto, ilustró la complejidad inherente al proceso de sintonización de modelos de reconocimiento facial. La optimización manual de la red y el ajuste de hiperparámetros específicos han permitido maximizar el rendimiento en un modelo sobre conjunto de datos particular, lo que subraya la importancia de un enfoque meticuloso y adaptable. Los modelos seleccionados para la experimentación final han demostrado un desempeño prometedor, destacándose por su precisión y estabilidad.

Por otro lado, el desarrollo de un *framework* que facilita la sintonización y optimización del

modelo representó un avance crucial para el modelo. Este *framework* permitió la abstracción de hiperparámetros clave, simplificando la experimentación y minimizando el código necesario.

Sintonización automática La implementación de hipermodelos en PSOT, emulando las arquitecturas de los modelos seleccionados, ha demostrado ser una estrategia altamente efectiva. Esta estrategia permite explorar y ajustar una amplia gama de configuraciones y tipos de datos, optimizando así el rendimiento del modelo de manera exhaustiva. En este sentido, se evidencian las ventajas de implementar hipermodelos para la sintonización.

PSOT demostró su capacidad para sintonizar hiperparámetros específicos de un hipermodelo con arquitectura VGG, lo que permitió que sus características se ajustaran correctamente al problema en cuestión. Esta capacidad de ajuste fino demuestran la flexibilidad del *framework* implementado y su correcto funcionamiento ante problemas de mayor complejidad. Los resultados obtenidos al abordar el reto con PSOT fueron superiores a los obtenidos mediante la sintonización con los métodos de Keras Tuner. Estos resultados representan un hito significativo en la capacidad del sintonizador para generar buenas configuraciones de hiperparámetros de manera eficiente y efectiva.

Sin embargo, es importante mencionar el alto costo computacional asociado con el manejo de un conjunto de datos tan grande y un hipermodelo como el utilizado para enfrentar este reto. El procesamiento y análisis de un gran volumen de imágenes requiere recursos significativos en términos de potencia de cálculo y almacenamiento, lo que puede representar un desafío para la implementación y ejecución eficiente de búsquedas en los espacios de soluciones como lo hace PSOT.

Centrado en humano Consecuentemente, sostenemos la importancia de la experiencia y el juicio humano en la obtención de buenos resultados. Si bien el sintonizador ofrece una alternativa valiosa y automatizada para la optimización de hiperparámetros, la intervención de un experto en la sintonización manual aún logra mejores resultados cercanos a exactitudes del 89 % frente al 81 % obtenido por PSOT. La intervención del experto es crucial para superar limitaciones y obtener resultados aún más precisos y adaptables a casos específicos.

Finalmente, la experiencia adquirida en la resolución de este desafío de reconocimiento facial ha resaltado la importancia de la sintonización informada y automatizada en el diseño y optimización de modelos de aprendizaje automático. Este enfoque, centrado en nuestro sintonizador, no solo ha demostrado su eficacia al mejorar el rendimiento y resultados de los modelos de reconocimiento facial, sino que también ha proporcionado una sólida base para abordar problemas similares en el futuro. Estos conocimientos y habilidades son esenciales para seguir avanzando en la investigación y desarrollo de soluciones innovadoras en el campo de la inteligencia artificial y la computación natural.

Trabajo futuro

Algunas ideas de trabajo futuro para nuestro proyecto son:

- Debemos realizar investigaciones adicionales para explorar cómo diferentes configuraciones de PSO pueden afectar su rendimiento y su aplicabilidad en una variedad de escenarios más complejos, así como optimizar el funcionamiento del *framework* para mejorar rendimientos y disminuir su consumo de recursos.

- Investigar enfoques para mejorar la eficiencia computacional del proceso de sintonización, incluyendo la optimización de algoritmos y la exploración de técnicas de paralelización y distribución de carga de trabajo en entornos de computación distribuida.
- Explorar la posibilidad de integrar funciones multiobjetivo para el sintonizador, permitiendo no solo tener en cuenta las métricas como exactitud, sino también del uso de recursos así como los tiempos de ejecución, balanceando no solo buenos resultados sino también eficacia y eficiencia en la búsqueda de soluciones.
- Continuar la exploración de la sintonización automática e informada de hiperparámetros para el reconocimiento facial, incluyendo la experimentación con otros hipermodelos y mejores arquitecturas para el proceso de construcción y evaluación.
- Investigar la adaptabilidad y la escalabilidad del *framework* PSOT para problemas de reconocimiento facial en contextos más amplios y variados, incluyendo la evaluación de su desempeño con el conjunto completo de VGGFAce2 o en diferentes conjuntos de datos, así como la extensión de su funcionalidad para abordar problemas más complejos y variados.

Agradecimientos

Queremos expresar nuestro más sincero agradecimiento a la profesora María Irma Díaz Rozo, cuya guía y apoyo invaluables fueron fundamentales para el desarrollo de este proyecto. Su experiencia y sus sugerencias nos orientaron en cada etapa del proceso, permitiéndonos superar los desafíos y alcanzar nuestros objetivos con éxito.

Agradecemos profundamente al jurado de investigación, quienes proporcionaron valiosos comentarios y críticas constructivas que enriquecieron significativamente nuestro trabajo. Sus observaciones y recomendaciones fueron esenciales para mejorar la calidad de nuestra investigación.

Finalmente, extendemos nuestro agradecimiento a todos aquellos que, directa o indirectamente, contribuyeron al éxito de esta investigación, desarrollo e implementación. Sus aportes, aunque no siempre visibles, fueron esenciales para llevar a cabo este trabajo.

Apéndices

Apéndice A

Presentación

EVALUACIÓN DE TRABAJO DIRIGIDO 2024-1

Sintonizador de hiperparámetros de redes neuronales usando computación natural (PSO: Optimización de enjambre de partículas

Santiago Andrés Rocha Cristancho - Sebastián Rojas Bueno

	Criterio	Insuficiente	Suficiente	Sobresaliente
1	Originalidad y alcance del problema		X	XX
2	Rigor en el desarrollo del trabajo			XXX
3	Pertinencia de los resultados		XX	X
4	Calidad de las respuestas a las inquietudes planteadas			XXX
5	Claridad en la presentación y sustentación de las ideas			XXX

Comentarios

- Clara la sustentación de cada una de las etapas. Como único punto a consideración, sustentar y realizar análisis más profundos sobre los resultados y mejoras.
- Excelente presentación, a pesar que el tema fue denso. Se ve un gran trabajo, faltaría la aplicabilidad, pero los resultados son evidencias de un trabajo excelente.
- Colocar los objetivos específicos en la presentación.

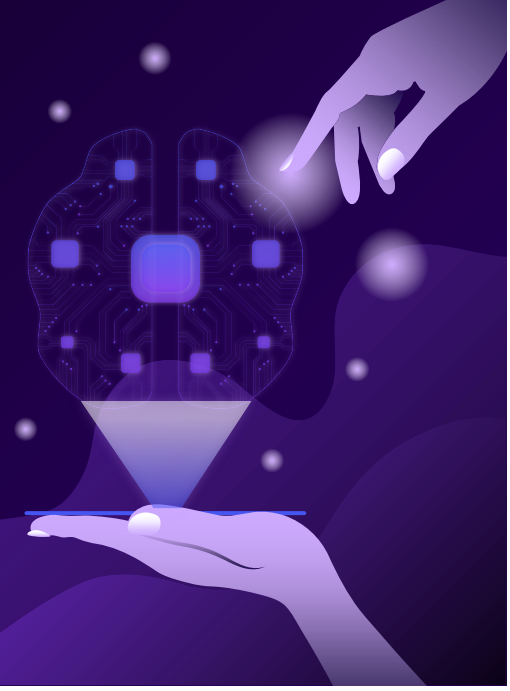
Evaluadores:

- Jossie Esteban Murcia. Experto en Investigación. Fundación Universitaria Sanitas.
- Luis Eduardo Rodriguez Cheu. Director de Investigación e Innovación. Universidad Escuela Colombiana de Ingeniería Julio Garavito
- Gerardo Ospina Hernández. Profesor. Universidad Escuela Colombiana de Ingeniería Julio Garavito

PSO Tuner

Challenge: Facial Recognition

Santiago Andrés Rocha
Sebastián Rojas Bueno



¿Cuál es el Problema?_

El problema central radica en la **sintonización de los hiperparámetros** de una Convolutional Neural Network (CNN).

Encontrar la combinación óptima de estos hiperparámetros de manera manual es una tarea compleja y consume mucho tiempo.

Reto:
CNN es comúnmente utilizada para Procesamiento de imágenes. Proponemos sintonizar CNNs orientadas al Reconocimiento Facial.

¿Por qué es importante resolverlo?_

Eficiencia

Reducir el tiempo de entrenamiento y mejorar la eficiencia computacional

Precisión, confiabilidad

Obtener los resultados deseados, garantizar que el sistema pueda identificar y evitar errores.

Reto

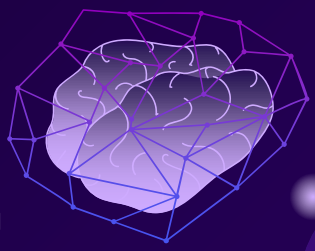
Reconocimiento facial de estudiantes Universitarios

Metodología

- Investigación: Marco Teórico → Documentación: Documentos de Revisión FR – PSOT / Artículos / Cronograma
- Investigación: Bases de Datos FR - PSOT → Desarrollo y Experimentación: Primer Sintonizado – Modelo Existente – Keras Tuner
- Desarrollo y Experimentación: Uso y expansión de BD FR → Desarrollo: PSOT
- Experimentación: Entrenamiento de CNN FR Sintonizado de CNN con PSOT → Documentación: Constante y en paralelo

Objetivos específicos

- **Desarrollar un framework de sintonización de hiperparámetros**
- **Crear un modelo de referencia para comparación**
- **Demostrar la eficacia del framework**
- **Aplicar el framework a problemas complejos de reconocimiento facial**
- **Contribuir al campo de la sintonización de hiperparámetros**



Marco Teorico - Sintonizadores_

Qué son los Sintonizadores?

- "Herramienta que automatiza la búsqueda del conjunto óptimo de hiperparámetros para un modelo de aprendizaje automático."
- Exploración del espacio de búsqueda con estrategias como:
  - Clásicas -> Ej:Grid Search – Random Search
  - Bioinspiradas -> Ej: Particle Swarm Optimisation (PSO)
- Herramienta conocida:
  - Keras Tuner

Orígenes

- Búsqueda de mejoras en modelos implementados:
  - Reducción de tiempos de ejecución
  - Reducción en costo computacional
  - Mejora en la generalización de los modelos
- Contribuciones Significativas en el desarrollo y la popularización de técnicas de optimización de hiperparámetros:
  - James Bergstra
    - HyperOpt: Librería de Opt. basada en Bayesiana
  - Yoshua Bengio
    - Investigaciones en técnicas y aplicaciones en Deep Learning
  - Jasper Snoek
    - Hyperas: Librería que usa HyperOpt en Keras

¿Qué se ha hecho hasta el momento?_ Sintonizadores_

Optimizadores de Hiperparámetros

Bayesian Opt
Grid Search Opt
Random Search Opt
Hyperband Opt

Keras Tuner

Hyperas

PSO

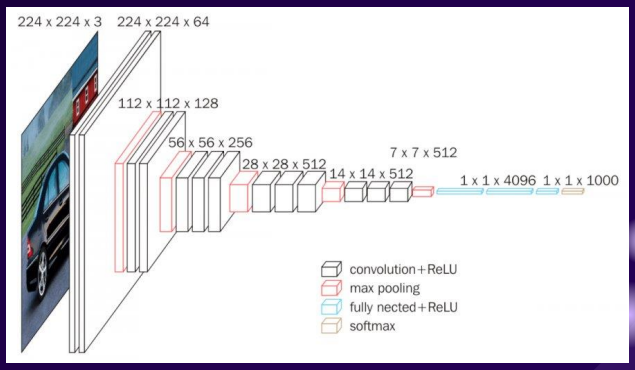
"Evolución de CNNs"

Características	Presente
Partícula = Arquitectura Específica	Si
Codificación basada en grupos	No
Parámetros: Coeficiente Cognitivo Coeficiente Social Inercia	Si

Marco Teórico_

Convolutional Neural Networs

- **Arquitectura**:
  - No. de Capas Convolucionales
  - No. de Capas Densas
  - No. De neuronas por Capa Densa
  - No. De Filtros
  - Tamaño del Kernel de Convolución
  - Tamaño de Pooling
  - Tipo de Pooling
  - Strides
  - Padding
- **Optimización**
  - Tasa de Aprendizaje
  - Épocas
  - Tamaño de los lotes
  - Funciones de Activación
  - Optimizador
- **Regularización**
  - Tasa de Dropout



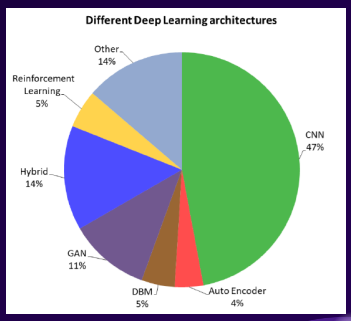
Marco Teorico Reconocimiento facial_

Reconocimiento Facial

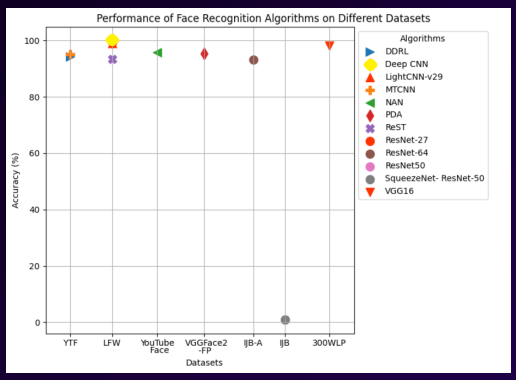
Tecnología que identifica o verifica personas usando sus rostros. Se basa en algoritmos que analizan características faciales únicas. Aunque sus orígenes se remontan a los años 60, recientes desarrollos en aprendizaje profundo han revolucionado el campo.

Aplicaciones

Crucial en sistemas de seguridad, autenticación biométrica, marketing, entre otras.



¿Qué se ha hecho hasta el momento?_ Reconocimiento facial_



Arquitectura

VGG16 es una de las arquitecturas de CNN más populares y ampliamente utilizadas en reconocimiento facial y otros campos de visión por computadora.

Bases de datos

VGGFace2 es una base de datos grande y diversa que contiene imágenes faciales de alta calidad de una amplia variedad de celebridades.

¿Cuál es la Idea de Solución?_

Sintonizador de Hiperparámetros

Existentes | Construido

Keras Tuner | Basado en PSO

BD

Bases de datos preliminares, de experimentación.

Test – 15%
Validación – 15%
Entrenamiento – 70%

Reconocimiento Facial

Extracción de características.

CNNs

CNNs para clasificación de imágenes.
Alta complejidad, alto número de Hiperparámetros, sintonizado específico.

¿Cuál es la Idea de Solución? Sintonizadores_

Bases de Datos - PSOT

Experimental

MNIST:
- Contenido: Dígitos
- Resolución: 28x28
- No. de Datos: 70000

CIFAR10
- Contenido: Animales + Vehículos
- Resolución: 36x36
- No. de Datos: 60000

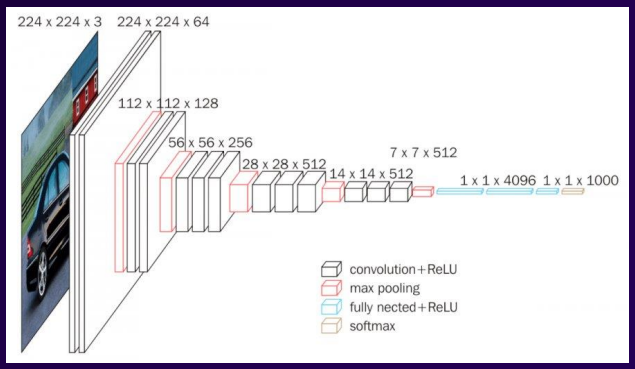
Implementar Heurísticos

- **Filters x Increasing Depth:**
  - Mayor profundidad = Mayor número de filtros
- **Pooling x Deepnes:**
  - Avg Pooling en capas intermedias
  - Max Pooling en capas externas
- **Decreasing Density:**
  - Mayor profundidad = Menor número de neuronas
- **Constant Regularization:**
  - Dropout para reducir sobreajuste y mejorar generalización

¿Cuál es la Idea de Solución? Sintonizadores_

Hiperparámetros Sintonizables

- **Arquitectura**:
  - No. de Capas Convolucionales
  - No. De Filtros
  - Tamaño del Kernel de Convolución
  - Tamaño de Pooling
  - No. de Capas Densas
  - No. De neuronas por Capa Densa
- **Optimización**
  - Tasa de Aprendizaje
  - Tamaño de los lotes
- **Regularización**
  - Factor de Crecimiento de Filtros
  - Factor de Decrecimiento de Neuronas
  - Tasa de Dropout
  - Factor de Crecimiento de Dropout



¿Cuál fue la Solución?

PSOT

Framework de Sintonización de Hiperparámetros usando Particle Swarm Optimization.

Hiperparámetros como Partículas

Abstracción de hiperparámetros sintonizables como partículas en el enjambre

Hipermodelos configurables

Modelos con arquitectura variante dependiendo de los hiperparámetros.
Desde lo más básico, hasta arquitecturas específicas.

Heurísticos de CNN

Hipermodelo que implementa heurísticos que ayudan a un mejor funcionamiento / resultados.
Ej: Filtros, Neuronas, Dropout

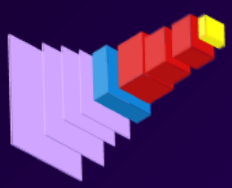
Visualizador de Soluciones

Modelo construido, Matriz de confusión, ejemplos de datos y resultados

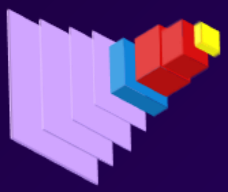
Resultados_ Hiperparámetros

Hiperparámetro	Límite Inferior	Límite superior	MNIST	CIFAR10
Capas Convolucionales	2	5	4	4
Capas Densas	2	3	3	2
Filtros	16	64	31	53
Factor de Crecimiento de Filtros	0	1	0.98	0.10
Tamaño del Kernel	3	5	5	4
Tamaño del Kernel de Capa Pooling	1	2	2	1
Indicador de Profundidad	0	1	0.38	0.20
Neuronas por Capa Densa	30	1024	590	410
Factor de Decrecimiento de Neuronas	0	1	0.99	0.90
Ritmo de Aprendizaje	0	0.001	0.00098	0.00051
Tamaño de los Lotes	50	100	83	70
Ritmo de Dropout	0	1	0.79	0.44
Factor de Dropout	0	0	0	0
Exactitud de Validación			**99.4666%**	**73.4600%**
Exactitud de Pruebas			**99.4599%**	**71.2999%**
Tiempo de Ejecución			19 Minutos	21 Minutos

MNIST



CIFAR10



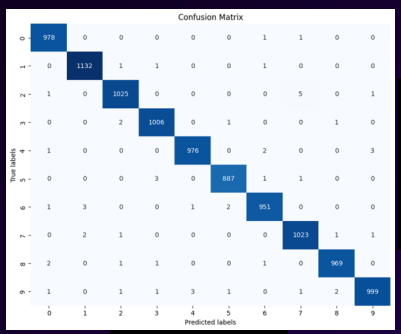
Épocas: 5
Partículas: 5
Iteraciones: 5
Coeficiente:

- Cognitivo: 0.5
- Social: 0.3
- Inercia: 0.9

Capa Convolucional
Capa de Aplanamiento
Capa Densa
Capa de Salida

Resultados Obtenidos_

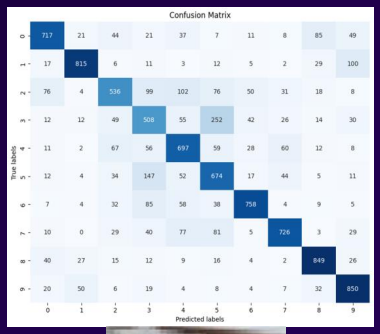
PSOT - MNIST

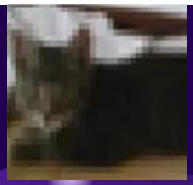


¿  ?

99.4666%	**Exactitud de Validación**	**73.4600%**
99.4599%	**Exactitud de Pruebas**	**71.2999%**
+0.5% / +0.9%	**Mejoras frente a Keras Tuner**	**+10% / +15%**

PSOT - CIFAR10

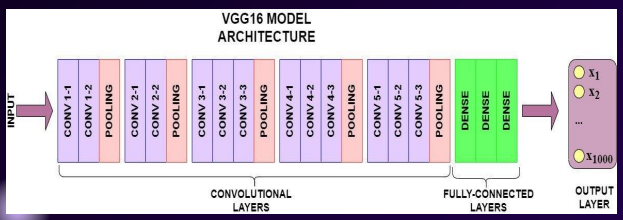


¿  ?

¿Cuál es la Idea de Solución?
Reconocimiento Facial_

Arquitectura de CNN

VGG16



Base de Datos - FR

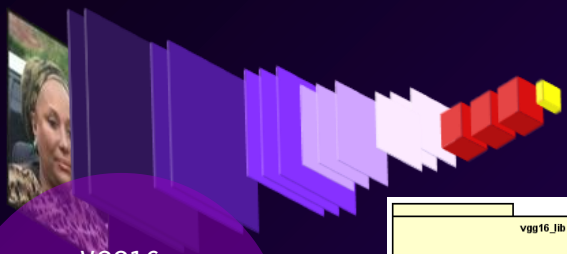
VGGFace 2 MAX

10 + 2 + 1 Personas

+ 500 fotos c/u

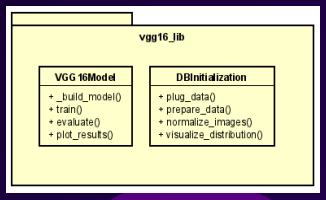
Una vez seleccionado el modelo se planea integrar con el sintonizador

¿Cuál fue la solución?_ FR



VGG16

- Mayor libertad de parámetros a sintonizar
- Menor Rendimiento



KVGG16 + LT

- Menor libertad de parámetros a sintonizar
- Mayor Rendimiento
- Transferencia de aprendizaje

Hyper VGG16

- Tiene mayor libertad de asignación de HP
- Se adapta a cualquier tipo de DB

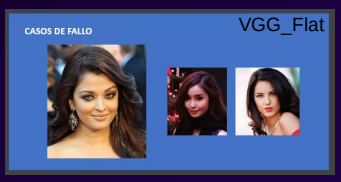
¿Qué resultados han obtenido?

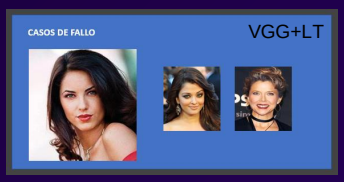
Resultados con VGG_Flat

Experimento	vgg_conf	Num_Filtros	Kernel_Size	Learning_Rate	Num_Neuronas	Pooling_Type	Pooling_Size	Num_Classes	Batch_Size	Épocas	Precisión Entrenamiento	Precisión Validación	Precisión Test
0	[2, 2, 3, 3, 3]	64	(3, 3)	0.0001	4096	max	(2, 2)	12	64	30	90.24%	89.65%	89.41%
1	[2, 2, 3, 3, 3]	128	(3, 3)	0.0005	4096	max	(2, 2)	12	80	30	10.79%	10.80%	-
2	[2, 2, 3, 3, 3]	32	(3, 3)	0.0001	2048	max	(2, 2)	12	64	30	87.51%	88.19%	-
3	[2, 2, 3, 3, 3]	96	(3, 3)	0.0001	3072	max	(2, 2)	12	64	30	87.51%	89.99%	86.55%

Resultados con VGG+TL

Modelo	Learning Rate	Batch Size	Epochs	Dropout Rate	L1 Regularization	L2 Regularization	Momentum	Train Loss	Train Accuracy	Validation Loss	Validation Accuracy	Accuracy
0	0.001	32	50	0.2	0.001	0.001	0.9	14.1751	50.60%	13.9013	68.05%	-
1	0.0001	64	40	0.3	0.01	0.01	0.0	14.3366	10.76%	14.3401	10.80%	-
2	0.0005	32	50	0.2	0.0001	0.0001	0.5	1.9148	78.45%	1.6757	87.18%	-
3	0.0005	64	50	0.2	0.0	0.0	0.95	0.2534	91.25%	0.2792	92.13%	88.09%
4	0.001	128	70	0.2	0.0	0.0	0.0	0.2444	91.59%	0.3189	90.55%	87.21%





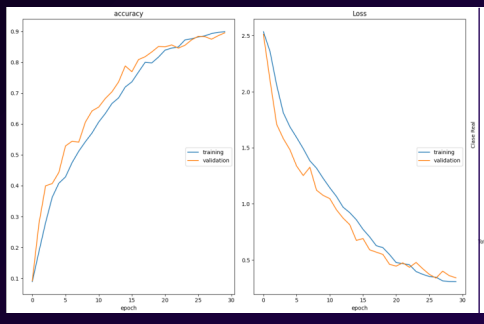
Resultados de Sintonización con Keras

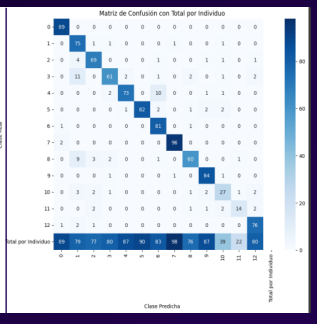
Random Search

```
Mejores hiperparámetros:
- Internal Conv Layers: 1
- External Conv Layers: 1
- Num Filters: 16
- Kernel Size: 2
- Pooling Kernel Size: 3
- Dense Neurons: 1024
- Learning Rate: 0.0003000000000000003
- Pooling Type: max
- Stride Size: 2
```

```
accuracy: 0.6819
val_accuracy: 0.7789
Test accuracy: 76.70%
```

¿Qué resultados han obtenido?

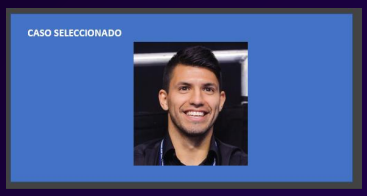


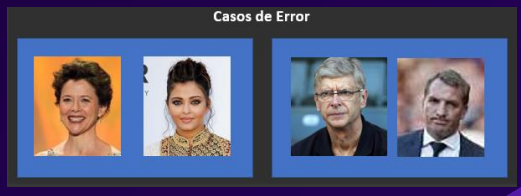


loss: 0.3064
accuracy: 0.8992

val_loss: 0.3408
val_accuracy: 0.8957

VGG16 Model
Accuracy: 89.87%

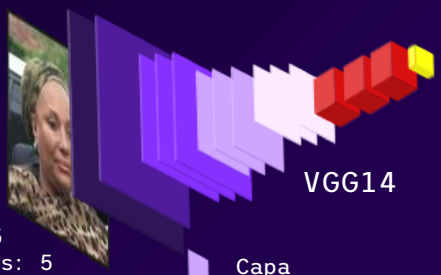




Resultados_ Hiperparámetros

Hiperparámetros	Límite Inferior	Límite superior	VGG
Capas Convolucionales Internas	1	2	1
Capas Convolucionales Externas	1	4	3
Filtros	32	64	46
Tamaño del Kernel	2	4	3
Tamaño del Kernel de Capa Pooling	2	3	3
Tamaño de Strides	2	4	2
Neuronas por Capa Densa	1024	4096	2052
Ritmo de Aprendizaje	0.0001	0.00001	0.000024
Tamaño de los Lotes	0	16	4
Validation Accuracy			**80.7851%**
Test Accuracy			**77.9128%**
Tiempo de Ejecución			**22 Minutos**
Entorno de Ejecución			**A100 GPU**

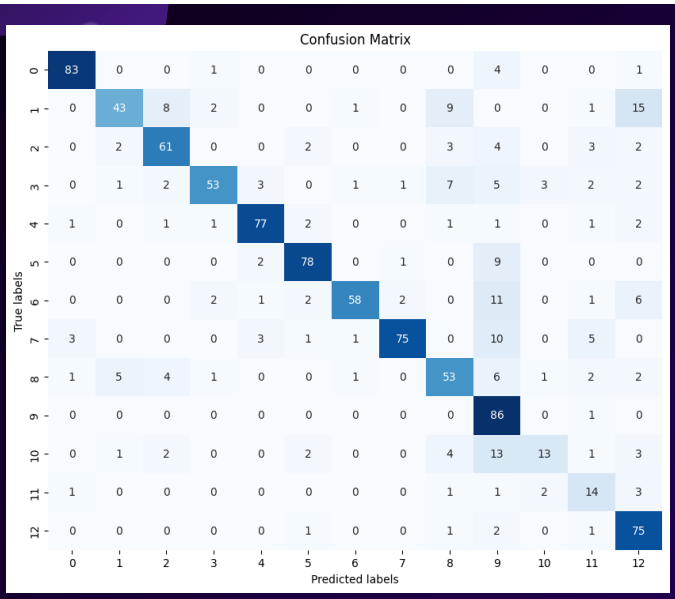
VGGFace2MaxMini+1



Épocas: 5
Partículas: 5
Iteraciones: 5
Coeficiente:

- Cognitivo: 0.5
- Social: 0.3
- Inercia: 0.9

Capa Convolucional
Capa de Aplanamiento
Capa Densa
Capa de Salida



Resultados Obtenidos_

VGGFace2Max Mini+1

¿  ?

	Train Loss	Train Accuracy	Validation Accuracy	Test Accuracy
PSOT	**0.1919**	**93.7045%**	**80.7851%**	**77.9128%**

¿Qué podemos aprender de los resultados?_

Mejores resultados que los sintonizadores de Keras

No mejores que la sintonización manual

Generación de hipermodelos eficaces, pero gran costo computacional

Contribuciones Principales_

PSOT
Framework de sintonización de hiperparámetros para Hipermodelos de CNN

Hipermodelo de CNN
Heurísticos e hiperparámetros específicos

Un Framework con múltiples aplicaciones en campos como:

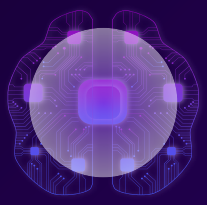
- Automatización de la Sintonización de Hiperparámetros
- Optimización de Modelos en Diversos Dominios (Visión por Computadora o Reconocimiento Facial)
- Comparación y Evaluación de Sintonizadores
- Aplicaciones Industriales y Comerciales

Hipermodelo VGGXX
Experimentación con arquitecturas VGGXX, sintonización manual y obtención de resultados

VGGFace2MaxMin+1
Conjunto de imágenes modificado y ampliado para su uso y experimentación

Trabajo Futuro_

Visualización de resultados, implementación de interfaz que los enseñe de mejor forma



Optimización del framework, uso de recursos, consumo de memoria

Experimentacion con hiperparámetros de PSO: Coeficientes

Despliegue

Gracias!_

Preguntas?
Dudas?
Sugerencias?

Referencias_

Tuners

- Franklin Johnson, Alvaro Valderrama, Carlos Valle, Broderick Crawford, Ricardo Soto, and Ricardo Ñanculef. Automating configuration of convolutional neural network hyperparameters using genetic algorithm, 2020.
- Tom Lawrence, Li Zhang, Chee Peng Lim, and Emma-Jane Phillips. Particle swarm optimization for automatically evolving convolutional neural networks for image classification, 2021.
- Vladyslav Yaloveha, Andrii Podorozhniak, and Heorhii Kuchuk. Convolutional neural network hyperparameter optimization applied to land cover classification, 2022.

Facial Recognition

- Jason Brownlee. How to perform face recognition with vggface2in
- Andre Budiman, Fabian, Ricky Aryatama Yaputera, Said Achmad, and Aditya Kurniawan. Student attendance with face recognition (lbph or cnn): Systematic literature review. Procedia Computer Science, 216:31–38, 2023. 7th International Conference on Computer Science and Computational Intelligence 2022.
- Md. Tahmid Hasan Fuad, Awal Ahmed Fime, Delowar Sikder, Md. Akil Raihan Iftee, Jakaria Rabbi, Mabrook S. Al-Rakhami, Abdu Gumaei, Ovishake Sen, Mohtasim Fuad, and Md. Nazrul Islam. Recent advances in deep learning techniques for face recognition. IEEE Access, 6:32, 07 2021.
- Guosheng Hu, Yongxin Yang, Dong Yi, Josef Kittler, William Christmas, Stan Z. Li, and Ti- mothy Hospedales. When face recognition meets with deep learning: an evaluation of convolu- tional neural networks for face recognition, 2015.
- Shaoqing Ren Jian Sun Kaiming He, Xiangyu Zhang. Deep residual learning for image recogni- tion. arXiv, 12 2015.
- Omkar M. Parkhi, Andrea Vedaldi, and Andrew Zisserman. Deep face recognition. In British Machine Vision Conference, 2015.

Apéndice B

Artículos

B.1. Sintonización de Hiperparámetros

B.1.1. Automating Configuration of Convolutional Neural Network Hyperparameters Using Genetic Algorithm [6]

Propósito El problema abordado en el artículo es la optimización de arquitecturas de Redes Neuronales Convolucionales (CNN) para tareas de clasificación de imágenes mediante el uso de algoritmos genéticos. La optimización de arquitecturas de CNN es crucial para mejorar el rendimiento en tareas de clasificación de imágenes. La automatización de este proceso puede dar como resultado modelos más eficientes y exactos.

Resultados Se propone un algoritmo genético para optimizar automáticamente la estructura de las CNN. La solución incluye un enfoque "novedoso"para la representación de cromosomas, operadores genéticos adaptativos y estrategias de selección que permiten explorar la diversidad en las primeras generaciones y concentrarse en soluciones exitosas para el final.

- Representación del cromosoma: se utiliza una representación especial de cromosomas para codificar la profundidad de la red, el número de filtros en cada capa convolucional y el tamaño del kernel de cada capa. Esto permite una representación compacta y completa de la arquitectura de la CNN.
- Operadores genéticos:
  - Mutación: se implementa una mutación que puede cambiar aleatoriamente los valores de los genes (número de filtros, tamaño del kernel) o incluso ajustar la profundidad de la red.
  - Cruce: se introduce un operador de cruce secuencial que combina secuencias exitosas de arquitecturas. También se emplea un cruce binario para diversificar en las primeras generaciones.
- Estrategias de selección: la estrategia de selección se adapta a lo largo del tiempo. En las primeras generaciones, se impulsa la diversidad aceptando individuos mutados y aleatorios. Hacia las últimas generaciones, se privilegia la selección basada en el rendimiento para intensificar la búsqueda.
- Entrenamiento y evaluación: utilizan el conjunto de datos de entrenamiento y validación para evaluar el rendimiento de cada arquitectura. Se emplea el optimizador Adam para el entrenamiento y parada temprana (deteniendo después de 3 iteraciones consecutivas sin mejora en la precisión de validación).
- Experimentos y evaluación: la solución se evalúa en conjuntos de datos populares como MNIST, CIFAR10 y CALTECH256. Se realizan múltiples ejecuciones del algoritmo con diferentes semillas para obtener estadísticas significativas.

Los resultados de experimentos en conjuntos de datos como MNIST, CIFAR10 y CALTECH256 muestran que el algoritmo propuesto supera a otros en términos de precisión de la red final. Con tiempos de cómputo comparables, la propuesta destaca por su eficacia en la optimización de las arquitecturas.

Aportes Entre los aportes notables se incluye un operador de cruce secuencial que preserva partes exitosas de arquitecturas, así como un diseño que diferencia entre las primeras y últimas generaciones para fomentar la exploración y la explotación, respectivamente.

B.1.2. Convolutional Neural Network Hyperparameter Optimization Applied to Land Cover Classification [15]

Propósito El artículo aborda el problema de la optimización de hiperparámetros en redes neuronales convolucionales (CNN) aplicadas a la clasificación de cobertura terrestre, utilizando imágenes de teledetección. El reto consiste en encontrar la combinación óptima de arquitectura de red y parámetros que maximice las métricas de rendimiento, lo que tiene aplicaciones en campos como la teledetección y la monitorización ambiental.

Resultados Se propone utilizar el método de optimización de hiperparámetros BOHB (Bayesian Optimization and Hyperband) para encontrar la combinación óptima de arquitectura de CNN y parámetros. Este enfoque combina técnicas bayesianas con métodos basados en bandas y ha demostrado ser eficaz en la optimización de algoritmos de aprendizaje profundo. El proceso de desarrollo de la solución se dividió en varias etapas clave, centradas en la implementación de la optimización de hiperparámetros utilizando el método BOHB y la mejora del rendimiento de la CNN aplicada.

1. Preprocesamiento de datos: el preprocesamiento de datos incluyó la normalización de las imágenes de teledetección, la segmentación del terreno y la generación de etiquetas para el conjunto de entrenamiento y prueba.
2. Diseño de la arquitectura de la CNN: se definió una arquitectura base para la CNN, que sirvió como punto de partida. Esta arquitectura incluyó capas convolucionales, capas de *pooling* y capas totalmente conectadas.
3. Implementación de BOHB: se integró una biblioteca BOHB para la optimización de hiperparámetros. BOHB explora el espacio de búsqueda de manera eficiente, realizando evaluaciones paralelas de configuraciones.
4. Optimización Iterativa: se llevaron a cabo iteraciones de optimización, donde BOHB ajustó continuamente los hiperparámetros de la CNN. Durante este proceso, se evaluaron diferentes combinaciones de parámetros y arquitecturas, buscando maximizar las métricas de rendimiento.
5. Evaluación del modelo optimizado: después de varias rondas de optimización, se evaluó el modelo resultante en el conjunto de prueba. Se registraron y compararon métricas clave como precisión, recall, F1-score y matriz de confusión.
6. Ajustes finales y análisis: se realizaron ajustes finales en la arquitectura y los parámetros seleccionados por BOHB. Se llevó a cabo un análisis detallado de los resultados, identificando patrones en las predicciones y posibles áreas de mejora.
7. Validación comparativa: el rendimiento del modelo optimizado se comparó con un modelo de referencia previamente establecido. La validación comparativa ayudó a destacar las mejoras logradas y la eficacia del enfoque de optimización de hiperparámetros.

Después de realizar iteraciones de optimización, se logró una mejora significativa en las métricas de clasificación. La exactitud final de la red optimizada en el conjunto de prueba aumentó del 84.19 % al 95.31 %, y la puntuación F1 mejoró de 84.05 % a 95.33 %. Los resultados superaron al modelo de referencia previamente seleccionado.

Aportes

- La implementación de BOHB demostró ser efectiva en la optimización de hiperparámetros para CNN en la clasificación de cobertura terrestre.
- Se lograron mejoras sustanciales en las métricas de rendimiento, lo que destaca la importancia de la optimización de hiperparámetros en tareas de aprendizaje profundo aplicadas a la teledetección.
- El modelo optimizado mostró resultados competitivos con modelos más complejos, destacando la eficiencia en términos de tiempo de entrenamiento y número de parámetros.

B.1.3. Hyper-Parameter Optimization: A Review of Algorithms and Applications [16]

Propósito El artículo presenta un resumen algunos tópicos de optimización de hiperparámetros: los hiperparámetros clave, los algoritmos de optimización, los recursos disponibles y los problemas actuales.

Resultados

1. Hiperparámetros relevantes: se identifican y explican los hiperparámetros clave en el entrenamiento de modelos, por ejemplo:
    - Learning Rate (Tasa de Aprendizaje): Controla la magnitud de los ajustes de los pesos durante el entrenamiento.
    - Batch Size (Tamaño del Lote): Número de ejemplos de entrenamiento considerados en una iteración.
    - Number of Layers (Número de Capas): Determina la profundidad de la red neuronal.
    - Activation Function (Función de Activación): Define la salida de cada neurona.
    - Regularization (Regularización): Controla el sobre ajuste mediante penalización de parámetros.
2. Enfoques tradicionales de optimización: se presentan métodos de optimización
    - *Grid Search* (Búsqueda en cuadrícula): - Método sistemático que evalúa todas las combinaciones posibles de hiperparámetros dentro de un rango predefinido. - Limitación: Su aplicación se vuelve inviable en espacios de búsqueda amplios debido a la maldición de la dimensionalidad.
    - *Random Search* (Búsqueda Aleatoria): - Estrategia más flexible que selecciona combinaciones de hiperparámetros de manera aleatoria. - Eficiente en las primeras etapas de la optimización, especialmente cuando se combina con estrategias de parada temprana.
    - *Multi-armed Bandit*: - Métodos basados en el problema del bandido de un solo brazo, incluyendo SHA, HB, BOHB, y ASHA. - Introducen una estrategia de parada temprana para equilibrar rendimiento y tiempo de entrenamiento, aumentando la eficiencia computacional.
    - *Population-Based Training* (PBT): - Enfoque único que combina la optimización de hiperparámetros y el entrenamiento del modelo en paralelo. - Eficiente al ajustar los hiperparámetros durante el entrenamiento, adaptándose a cambios no estacionarios.

3. Herramientas y servicios de optimización de hiperparámetros:
    - *Google Vizier*: - Servicio en la nube que ofrece optimización de hiperparámetros con algoritmos como Tree-structured Parzen Estimator (TPE). - Integración sencilla con la infraestructura de *Google Cloud*, proporcionando una interfaz intuitiva.
    - *Amazon SageMaker*: - Ambiente de aprendizaje automático en la nube que incluye módulos para optimización automática de modelos. - Destaca la integración completa de todas las etapas de entrenamiento de modelos, lo que evita la necesidad de cambiar entre herramientas.
    - *Ray.Tune* : - *toolkit* de optimización de código abierto - Escalable y compatible con algoritmos de "vanguardia Proporciona libertad de personalización - Implementa una amplia variedad de algoritmos de búsqueda, brindando flexibilidad a los usuarios.
    - NNI (*Neural Network Intelligence*): - Herramienta de código abierto desarrollada por Microsoft - Se despliega en diversas configuraciones, como máquinas locales, servidores remotos y docker. - Alta extensibilidad, permitiendo a los usuarios implementar y probar nuevos algoritmos de búsqueda.
4. Comparación entre algoritmos: se realiza un análisis comparativo de algoritmos clave, incluyendo *Grid Search*, *Random Search*, *Bayesian Optimization* (BO), métodos de *bandit* y *Population-Based Training* (PBT), evaluando ventajas y desventajas en términos de eficiencia, capacidad de paralelización y aplicabilidad para redes neuronales profundas. Destaca que PBT es especialmente eficiente para modelos a gran escala y revela las limitaciones de otros algoritmos en términos de complejidad conceptual y paralelización.

Aportes Se resume y evalúan enfoques y herramientas para la optimización de hiperparámetros en redes neuronales profundas, destacando la importancia de la eficiencia computacional y la elección adecuada de algoritmos en la optimización de modelos.

B.1.4. Particle Swarm Optimization for Automatically Evolving Convolutional Neural Networks for Image Classification [8]

Propósito Afrontar la evolución automática de arquitecturas de redes neuronales convolucionales (CNN) mediante optimización por enjambre de partículas (PSO). Se plantea la necesidad de superar limitaciones existentes en métodos actuales.

Resultados El enfoque propuesto se basa en dos ideas:

- Estrategia de codificación basada en grupos: la estrategia de codificación se basa en grupos de parámetros que definen la arquitectura de la CNN. Estos grupos representan componentes clave, como capas convolucionales, capas de agrupación, y capas completamente conectadas. La elección de esta estrategia se justifica en su capacidad para equilibrar la diversidad de las soluciones generadas y la eficiencia computacional.
- Esquema de cálculo de distancia de partículas: el esquema específico de cálculo de distancia de partículas es diseñado especialmente para evaluar la similitud entre dos arquitecturas de CNN. Este componente resulta ser fundamental para el proceso de búsqueda y optimización del PSO, ya que guía la exploración del espacio de soluciones hacia regiones prometedoras.

Los rangos seleccionados para los parámetros específicos del algoritmo PSO utilizados en la investigación (la velocidad y posición iniciales de las partículas; las ponderaciones de inercia, cognitivas y sociales; y los límites de velocidad)se basa en experimentos preliminares y consideraciones teóricas. La configuración experimental incluye el número de partículas en el enjambre, el número máximo de iteraciones, y los criterios de convergencia.

Aportes

- Logros
  - Mejora en el rendimiento de CNN automáticamente evolucionadas: la capacidad del PSO para explorar de manera eficiente el espacio de búsqueda llevó a la identificación de arquitecturas más efectivas en comparación con enfoques tradicionales.
  - Eficiencia en el uso de recursos computacionales: la estrategia de codificación y la implementación específica del cálculo de distancia contribuyeron a una utilización más eficiente de los recursos computacionales, permitiendo la evolución de arquitecturas de CNN con un menor costo computacional, e indirectamente, menor costo ambiental.
- Limitaciones
  - El algoritmo PSO puede ser sensible a la configuración específica de sus parámetros. Este aspecto destaca la importancia de realizar ajustes cuidadosos y experimentos sistemáticos para obtener los mejores resultados.
  - La capacidad de generalización de las arquitecturas evolucionadas a diferentes conjuntos de datos y dominios aún requiere una exploración más detallada, pues la investigación estuvo enfocada en tareas específicas de clasificación de imágenes.
- Trabajo futuro
  - Explorar enfoques para la optimización de hiperparámetros automática en conjunto con la evolución de arquitecturas, pues podría mejorar aún más el rendimiento y la generalización de las CNN resultantes.
  - Extensión a otros dominios de aplicación como el procesamiento de lenguaje natural o reconocimiento de voz, funcionaria para evaluar la versatilidad y aplicabilidad del desarrollo en diferentes contextos.
  - Inclusión de restricciones adicionales durante la evolución, como limitaciones en el uso de recursos específicos o la optimización de múltiples objetivos, permite acercarse de manera más efectiva problemas del mundo real.
  - Estudio detallado de robustez y la capacidad de transferencia de las arquitecturas evolucionadas en entornos con ruido o cambios en los datos con la finalidad de fortalecer la aplicabilidad práctica del método propuesto.

B.2. Reconocimiento Facial

B.2.1. Advances in Deep Learning-Based Face Recognition Systems [4]

Propósito El artículo aborda diversos desafíos en sistemas de reconocimiento facial basados en aprendizaje profundo, incluyendo variaciones de pose, iluminación, expresión facial, edad

y etnia. Estos desafíos dificultan la obtención de resultados precisos en situaciones del mundo real. El artículo es relevante para el trabajo de Convolutional Neural Networks (CNN) para Face Recognition (RF) reforzada con un algoritmo de colmena, ya que proporciona una comprensión detallada de los problemas y desafíos actuales en este campo. Esto puede ayudar a informar el diseño y la implementación de algoritmos más eficaces y robustos.

Resultados Presentan varios enfoques que los investigadores podrían considerar para reconocimiento facial.

1. Redes Neuronales Convolucionales (CNN): las redes neuronales convolucionales (CNN) son el algoritmo de aprendizaje profundo más popular para el reconocimiento de imágenes, clasificación de imágenes, reconocimiento de patrones y extracción de características de imágenes. Se describen dos tipos básicos de algoritmos CNN: el extractor de características y el clasificador. La arquitectura básica de una CNN incluye cuatro capas: convolucional, de agrupación, no lineal y completamente conectada. Varias arquitecturas de CNN, como VGGNet, GoogLeNet y ResNet, se han aplicado exitosamente en sistemas de reconocimiento facial. Se menciona el modelo ResNet50, entrenado con el conjunto de datos VGGFace2, que logra una precisión del 99.5 %. Otros modelos, como SqueezeNet-ResNet-50 y ResNet-64, introducen mejoras para abordar desafíos como la variación de edad y la iluminación. Además, se mencionan estudios que combinan CNN con técnicas de fusión tensorial para mejorar la precisión del reconocimiento facial en diversos escenarios.
2. *Autoencoder*: es una red neuronal profunda de aprendizaje no supervisado para la codificación y decodificación eficientes de datos. Se destaca su capacidad para aprender características robustas automáticamente a partir de grandes conjuntos de datos no etiquetados. La arquitectura consta de una o más capas ocultas con capas de entrada y salida. El proceso de codificación reduce las dimensiones de entrada a una representación significativa, seguido de la decodificación que reconstruye las características esenciales. Estas redes se han usado para resolver el problema de la invariancia a la edad en sistemas de reconocimiento facial. En este caso se usa la técnica de reconocimiento facial basada en *autoencoder* acoplado (CAN), que utiliza dos *autoencoders* para abordar procesos no lineales complejos relacionados con el envejecimiento y el rejuvenecimiento facial.
3. *Generative Adversarial Networks*: estas redes descubren y aprenden patrones o regularidades a partir de datos de entrada. La estructura básica de un GAN tiene dos submodelos: un generador para crear nuevas características y un discriminador para clasificar si las características generadas son reales o falsas. GANs se basan en un esquema teórico del juego, donde el generador compite contra un adversario. Las GANs se utilizan en diversas aplicaciones de reconocimiento facial, abordando problemas como reconocimiento facial a través de edades, síntesis facial, reconocimiento facial invariante a la pose, reconocimiento facial basado en video, y más.
4. *Deep belief network*(DBN): se presentan como una solución a los problemas comunes de las redes neuronales profundas tradicionales, como atascarse en óptimos locales, aprendizaje lento y la necesidad de conjuntos de datos de entrenamiento extensos. Una DBN es una composición de múltiples unidades ocultas que están conectadas internamente entre capas, pero no entre unidades de la misma capa. Se puede entender como una secuencia de máquinas de Boltzmann restringidas (RBM) o autoencoders, donde cada

subcapa oculta funciona como una capa visible para la siguiente subcapa oculta.

5. Modelos híbridos: los modelos híbridos son una combinación de dos o más modelos de aprendizaje automático genéricos para mejorar el rendimiento general del modelo. Esta técnica combina algoritmos para abordar problemas de manera más precisa que utilizando un solo algoritmo. Al combinar algoritmos como CNN, GAN, AE, RL, entre otros, se pueden obtener beneficios complementarios y mejorar el rendimiento del sistema. Existen varios modelos híbridos utilizados en el reconocimiento facial, como la combinación de ConvNet y RBM para verificación facial propuesta por Sun et al., DBMs y AEs para problemas de conocimiento facial invariante propuestos por Sing et al., y MLDFace, un modelo híbrido que combina DBM y Autoencoders para reconocimiento facial basado en video.
6. Aprendizaje Profundo por Refuerzo (RL): es una técnica que permite a un agente aprender a tomar decisiones mediante la interacción con un entorno. Este enfoque se inspira en el proceso de toma de decisiones humano y permite que el agente aprenda de sus experiencias a través de ensayo y error.

Adicionalmente, presenta el despeño de las diferentes funciones de activación, de costo, y de arquitecturas bajo diferentes datasets, comparándolas todas mediante a una métrica global, la exactitud.

Aportes Entre los aportes destacables del artículo se incluye la revisión exhaustiva de desafíos en el reconocimiento facial basado en aprendizaje profundo, así como la identificación de posibles áreas de investigación futura, como el reconocimiento facial en situaciones de uso de mascarillas y la exploración de imágenes infrarrojas para mejorar la detección facial.

B.2.2. Deep Face Recognition [11]

Propósito El artículo aborda el desafío del reconocimiento facial en condiciones variables, como cambios de pose, iluminación, expresión facial, edad y etnia. Estas variaciones dificultan la obtención de resultados precisos en aplicaciones del mundo real. Este artículo es relevante para el avance de los sistemas de reconocimiento facial basados en aprendizaje profundo, ya que proporciona una comprensión detallada de los problemas actuales y desafíos en este campo, tales como la construcción de una BD óptima sobre la cual trabajar para el entrenamiento de una CNN enfocada al FR. Esto puede ayudar a informar el diseño y la implementación de algoritmos más efectivos y robustos.

Resultados El articulo propone la construcción de su propia BD, dando los pasos a seguir:

- Etapa 1. *Bootstrapping* y filtrado de una lista de nombres de identidades candidatas: el primer paso es obtener una lista de nombres de identidades candidatas para obtener rostros. Se centran en celebridades y figuras públicas, como actores o políticos, para asegurar la disponibilidad de imágenes en la web y evitar problemas de privacidad. La lista inicial se obtiene extrayendo nombres de celebridades, principalmente actores, de la lista de celebridades de la *Internet Movie Data Base (IMDB)*. Luego, se hace la intersección de esta lista con todas las personas en el grafo de conocimiento *Freebase*, resultando en una lista de 5,000 nombres populares con información adicional sobre

etnia, edad, parentesco, etc. Después de un proceso de filtrado basado en la pureza de las imágenes, se obtiene una lista final de 2,622 nombres de celebridades.
- Etapa 2. Recopilación de más imágenes para cada identidad: se realizan búsquedas en *Google* y *Bing Image Search* para cada uno de los 2,622 nombres de celebridades, obteniendo un total de 2,000 imágenes por identidad.
- Etapa 3. Mejora de la pureza con un filtro automático: se utiliza un clasificador automático para eliminar caras erróneas en cada conjunto de imágenes. Se entrena un SVM (*Support Vector Machine*) lineal utilizando las mejores 50 imágenes para cada identidad como muestras positivas y las mejores 50 imágenes de todas las demás identidades como muestras negativas. El SVM se utiliza para clasificar las 2,000 imágenes descargadas, conservando las mejores 1,000.
- Etapa 4. Eliminación de duplicados cercanos: se eliminan imágenes duplicadas exactas y cercanas. Para esto, se utilizan descriptores VLAD (*Vector of Locally Aggregated Descriptors*) y se retiene un solo elemento por grupo de imágenes similares.
- Etapa 5. Filtrado manual final: se realiza un filtrado manual final con la ayuda de anotadores humanos. Se utiliza un clasificador CNN para ordenar las imágenes por la probabilidad de pertenecer a una identidad específica. Los anotadores validan bloques de 200 imágenes, declarando un bloque como bueno si la pureza es superior al 95 %. Se obtiene un total de 982,803 imágenes finales, aproximadamente el 95 % de las cuales son frontales y el 5 % perfiles.

Además, destaca la importancia de considerar nuevas formas de detección facial en situaciones como el uso de mascarillas, proponiendo posibles enfoques basados en cámaras infrarrojas.

Para el desarrollo de la red neuronal consideran los siguientes aspectos.

- Clasificador de rostros: inicialmente, las arquitecturas profundas se inician considerando el problema de reconocer N = 2,622 individuos únicos, configurado como un problema de clasificación de N clases. La CNN asocia a cada imagen de entrenamiento t, t = 1, ..., T, un vector de puntajes mediante una capa completamente conectada final que contiene N predictores, uno por identidad. Estos puntajes se comparan con la identidad de clase verdadera calculando la pérdida *softmax*. Después del aprendizaje, la capa de clasificación se elimina para obtener los vectores de puntajes que son mejorados usando la pérdida de tripletas.
- Incrustación de rostros usando una pérdida de tripletes: el entrenamiento con pérdida de tripletes tiene como objetivo aprender vectores de puntajes que funcionen bien en la aplicación final, es decir, la verificación de identidad comparando descriptores de rostros en el espacio euclidiano. Este entrenamiento es similar en espíritu al .ªprendizaje métricoz se utiliza para aprender una proyección que sea distintiva y compacta al mismo tiempo, logrando una reducción de dimensionalidad.
- Arquitectura: se consideran tres arquitecturas basadas en las arquitecturas A, B y D de [19]. La arquitectura A consta de 11 bloques, cada uno con un operador lineal seguido de una o más no linealidades como ReLU y max pooling. Las primeras ocho bloques son convolucionales, mientras que los últimos tres son totalmente conectados. Todas las capas de convolución están seguidas por una capa de rectificación (ReLU), pero no incluyen la normalización local de respuesta. Las dos primeras capas totalmente conectadas tienen

una salida de 4,096 dimensiones, mientras que la última capa tiene N = 2,622 o L = 1,024 dimensiones, dependiendo de las funciones de pérdida utilizadas.
- Entrenamiento: el entrenamiento de la CNN A sigue los pasos de [10] con las modificaciones sugeridas. Se utiliza descenso de gradiente estocástico con lotes miniatura de 64 muestras y un coeficiente de momento de 0,9. El modelo se regulariza utilizando dropout y decaimiento de peso. Las imágenes de entrenamiento se redimensionan y se alimentan a la red con parches aleatorios de 224x224 píxeles. La CNN A se entrena desde cero, mientras que las configuraciones B y D se entrenan comenzando desde A y agregando capas completamente conectadas adicionales.

Aportes

- Conclusiones
    - Curación de datos: mejora de rendimiento antes de la curación de datos, indicando la importancia de tener más datos, incluso con ruido en las etiquetas.
    - Alineación: la alineación 2D en imágenes de prueba mejora el rendimiento, pero no proporciona una mejora adicional en los datos de entrenamiento.
    - Arquitectura: cambios en la arquitectura muestran una ligera mejora en el rendimiento en ciertas configuraciones.
    - Pérdida de Tripletes: el aprendizaje de una métrica discriminativa mejora significativamente el rendimiento.
- Contribuciones
    - Desarrollo de un procedimiento para ensamblar un conjunto de datos a gran escala con anotación mínima.
    - Demostración de que una CNN profunda, sin adornos pero con entrenamiento adecuado, puede lograr resultados comparables al estado del arte.

B.2.3. Deep Residual Learning for Image Recognition [7]

Propósito El problema abordado en este artículo es la dificultad de optimizar redes neuronales convolucionales (CNN) extremadamente profundas debido a problemas como la degradación del rendimiento a medida que se incrementa la profundidad de la red. Al comprender y resolver la dificultad de entrenar redes profundas, se pueden mejorar significativamente los modelos de CNN utilizados en una variedad de aplicaciones, incluida la clasificación de imágenes e introduciendo modelos que en el momento son los más óptimos para este enfoque como lo son las redes residuales (ResNets).

Resultados La propuesta de solución en este artículo es el uso de redes residuales (*ResNets*) para abordar la dificultad de entrenar redes neuronales convolucionales extremadamente profundas. Las ResNets introducen conexiones de atajo que permiten que las capas aprendan las diferencias residuales en lugar de intentar aprender directamente la representación deseada. Esto ayuda a aliviar el problema de degradación del rendimiento que se encuentra al entrenar redes muy profundas.

Los tres conceptos básicos de estas redes son

- Red Residual (*Residual Network o ResNet*): es un tipo de arquitectura de red neuronal profunda que se centra en abordar el problema de degradación que surge al entrenar redes más profundas.
- Aprendizaje Residual: en lugar de esperar que las capas apiladas se ajusten directamente a un mapeo deseado $H(x)$, las *ResNet* proponen que estas capas aprendan un mapeo residual $H(x) - x$[1].
- Conexiones de atajo (*Shortcut Connections*): para facilitar el aprendizaje residual, se introducen conexiones de atajo que realizan una conexión directa desde la entrada hasta la salida de ciertas capas. Estas conexiones de atajo pueden ser de identidad (si las dimensiones coinciden) o pueden incluir proyecciones lineales para ajustar las dimensiones si es necesario.

El desarrollo de la solución implica la construcción y entrenamiento de *ResNets* con diferentes profundidades (18, 34, 50, 101, 152 capas) en conjuntos de datos como ImageNet y CIFAR-10. Se utilizan conexiones de atajo para facilitar el entrenamiento de redes profundas y se exploran diferentes estrategias para abordar el problema de degradación del rendimiento.

Los resultados experimentales muestran que las *ResNets* logran superar la dificultad de optimización y mejorar la precisión a medida que se aumenta la profundidad de la red. Se observa una reducción significativa en el error de entrenamiento y una mejora en la generalización en comparación con las redes neuronales convolucionales planas. Además, las *ResNets* logran resultados competitivos en comparación con otros métodos en conjuntos de datos como ImageNet y CIFAR-10.

Aportes Entre los aportes destacables se incluye la introducción de las conexiones de atajo en redes neuronales convolucionales, que han demostrado ser efectivas para entrenar redes extremadamente profundas. Además, el estudio detallado del problema de degradación del rendimiento y las estrategias propuestas para abordarlo proporcionan una comprensión más profunda de la optimización de redes neuronales convolucionales profundas.

B.2.4. How to Perform Face Recognition With VGGFace2in Keras [1]

Propósito El problema abordado en este artículo es como realizar el reconocimiento facial utilizando el modelo VGGFace2 en Keras. Se trata de identificar personas en imágenes utilizando modelos de redes neuronales convolucionales (CNN) pre-entrenados con grandes conjuntos de datos de celebridades. La importancia de este artículo radica en la aplicación práctica del reconocimiento facial utilizando modelos de vanguardia como VGGFace2, librerias conocidas como Keras y arquitecturas famosas como la ResNet50. El reconocimiento facial tiene numerosas aplicaciones en seguridad, autenticación, análisis de imágenes, entre otros campos.

Resultados La propuesta de solución consiste en utilizar el modelo pre-entrenado de VGGFace2 en Keras para realizar el reconocimiento facial. Se emplea una función `decode predictions`, para mapear las probabilidades de clase a nombres de celebridades y se utiliza otra, `preprocess input` para escalar adecuadamente las imágenes de entrada. La arquitectura de red VGGFace2, es una red neuronal convolucional (CNN) preentrenada. Específicamente, se utiliza la arquitectura *ResNet50*, que es una versión de la red ResNet que consta de 50 capas. Esta

[1]Supone dimensiones iguales

arquitectura ha demostrado ser efectiva en tareas de reconocimiento facial y clasificación de imágenes.

El desarrollo de la solución implica cargar el modelo VGGFace2 preentrenado, preparar las imágenes de entrada mediante el preprocesamiento adecuado, y luego realizar la predicción utilizando el modelo.

Los resultados presentados en el artículo muestran cómo el modelo VGGFace2 puede identificar correctamente a las personas en las imágenes con alta precisión. Se presentan ejemplos de reconocimiento exitoso de celebridades como Sharon Stone y Channing Tatum.

Aportes Entre los aportes destacables se incluye la aplicación práctica del modelo VGGFace2 en Keras para el reconocimiento facial de celebridades. Además, se proporciona una guía detallada sobre cómo utilizar el modelo y cómo interpretar los resultados.

B.2.5. Student attendance with face recognition (LBPH or CNN): Systematic literature review [2]

Propósito El artículo aborda el problema del registro de asistencia estudiantil en las universidades, destacando las limitaciones de los sistemas actuales basados en tarjetas RFID (*Radio Frequency Identification*) y los problemas asociados con el fraude y los errores humanos en los métodos manuales. El artículo presenta dos aproximaciones en la gestión de la asistencia estudiantil: Convolutional Neural Networks (CNN) para reconocimiento facial y el algoritmo de Local Binary Pattern Histogram (LBPH) . Proporciona una revisión sistemática de la literatura para comparar la eficacia de ambos algoritmos en entornos universitarios, lo que puede orientar la implementación de sistemas de asistencia más precisos, además de proporcionar una definición acertada del FR.

Resultados El reconocimiento facial es una tecnología que utiliza algoritmos para identificar o verificar a una persona a partir de una imagen digital o un video. Esta tecnología, que ha evolucionado significativamente desde los primeros trabajos de Woodrow Wilson Bledsoe en la década de 1960, implica la detección de rostros, la creación de un modelo matemático de los rostros y la comparación de estos modelos con una base de datos. Los algoritmos de reconocimiento facial pueden variar en su enfoque, con algunos centrados en características faciales específicas y otros que tratan la cara como una unidad completa. Muchos algoritmos modernos utilizan técnicas de aprendizaje profundo y redes neuronales artificiales para mejorar la precisión y eficiencia del reconocimiento facial.

La propuesta de solución implica la implementación de sistemas de reconocimiento facial utilizando algoritmos de CNN y LBPH para gestionar la asistencia estudiantil de manera más eficiente y precisa. Se plantea la selección del algoritmo más adecuado para cada caso, considerando factores como la precisión y la estabilidad frente a factores externos.

La solución se desarrolla considerando los siguientes pasos:

1. Recopilación de datos faciales de los estudiantes y su almacenamiento en una base de datos.

    - Grabación de videos de estudiantes: se realizan grabaciones de video de los estudiantes en diferentes situaciones y condiciones, como diferentes ángulos, iluminación y expresiones faciales.

- Conversión de videos en fotogramas: los videos grabados se convierten en una serie de fotogramas individuales para su análisis y procesamiento posterior.
- Conexión a una base de datos: los fotogramas de los videos se conectan a una base de datos facial que contiene imágenes de referencia de los estudiantes.
- Determinación de la presencia o ausencia: se compara cada fotograma de los videos con las imágenes de referencia en la base de datos para determinar si el estudiante está presente o ausente en cada momento.
- Registro de la asistencia: se marca la presencia o ausencia de cada estudiante en función de los resultados de la comparación con las imágenes de referencia, lo que permite mantener registros precisos de la asistencia.

2. Se lleva a cabo el entrenamiento de los algoritmos de reconocimiento facial utilizando los datos recopilados.
3. Se implementa un sistema de reconocimiento facial en las aulas utilizando cámaras para registrar y comparar los rostros de los estudiantes con los datos almacenados en la base de datos.
4. Los datos de asistencia se registran automáticamente en el sistema de asistencia de la universidad.

Los resultados obtenidos de la revisión de la literatura indican que el algoritmo de CNN supera en precisión al algoritmo LBPH, proporcionando una mayor estabilidad frente a factores externos que podrían afectar la precisión del reconocimiento facial.

Aportes El artículo destaca la superioridad en precisión y estabilidad del algoritmo de CNN sobre LBPH para el reconocimiento facial en entornos universitarios. Proporciona una guía útil para la implementación de sistemas de asistencia estudiantil basados en reconocimiento facial, lo que puede mejorar la eficiencia y la precisión en la gestión de la asistencia en las universidades.

B.2.6. When Face Recognition Meets with Deep Learning: an Evaluation of Convolutional Neural Networks for Face Recognition [5]

Propósito El artículo aborda el problema del reconocimiento facial utilizando redes neuronales convolucionales (CNN). Se centra en la optimización de la arquitectura de las CNN para mejorar el rendimiento en tareas de reconocimiento facial, específicamente en el contexto de la base de datos LFW (). Se analiza el uso de técnicas de fusión de redes neuronales puede ayudar a mejorar la precisión del reconocimiento facial en diferentes entornos y condiciones de iluminación.

Resultados Se propone fusionar múltiples redes neuronales convolucionales entrenadas con imágenes de diferentes regiones faciales y escalas para mejorar el rendimiento del reconocimiento facial. Se exploran dos estrategias de fusión: concatenación de características (C-Fusion) y promedio de valores de características (A-Fusion). La solución también incluye el uso de un algoritmo genético para optimizar automáticamente la estructura de las CNN y el uso de aprendizaje métrico (JB) para mejorar aún más el rendimiento del reconocimiento facial.

El trabajo utiliza la base de datos "Labeled Faces in the Wild"(LFW) para evaluar el rendimiento del sistema de reconocimiento facial basado en redes neuronales convolucionales

(CNN). Esta base de datos contiene 5,749 sujetos y 13,233 imágenes, y se utiliza para definir los conjuntos de entrenamiento y prueba, así como los protocolos estándar para evaluar el rendimiento del reconocimiento facial.

La solución propuesta se desarrolla en varias etapas:

1. Configuración de las CNN: se diseñan y entrenan tres arquitecturas de CNN (CNN-S, CNN-M y CNN-L) para el reconocimiento facial en la base de datos LFW. Se evalúa el rendimiento de cada arquitectura y se selecciona CNN-M como la más adecuada para el trabajo.
2. Fusión de Redes Neuronales: se entrenan 30 redes neuronales separadas utilizando imágenes de diferentes regiones faciales y escalas. Se exploran dos estrategias de fusión: concatenación de características (C-Fusion) y promedio de valores de características (A-Fusion).
3. Optimización automática: se utiliza un algoritmo genético para optimizar automáticamente la estructura de las CNN. Se emplea una representación especial de cromosomas para codificar la profundidad de la red, el número de filtros en cada capa convolucional y el tamaño del kernel de cada capa.
4. Aprendizaje métrico: se aplica el aprendizaje métrico (*Joint Bayesian*, JB) a las características extraídas de las CNN para mejorar la precisión del reconocimiento facial. Se utilizan características de la capa totalmente conectada (FC) y de la capa softmax para evaluar la efectividad del aprendizaje métrico. El aprendizaje métrico que tiene como objetivo encontrar una nueva métrica para hacer dos clases más separables.

Los resultados experimentales muestran que la fusión de redes neuronales mejora significativamente el rendimiento del reconocimiento facial en la base de datos LFW. Se observa que la combinación de características de diferentes capas de CNN y el aprendizaje métrico conducen a mejoras adicionales en la precisión del reconocimiento facial.

Aportes El artículo destaca por su enfoque novedoso de fusionar características de múltiples redes neuronales convolucionales para mejorar el reconocimiento facial. Además, la aplicación de un algoritmo genético para optimizar automáticamente la estructura de las CNN y el uso de aprendizaje métrico para mejorar la precisión son contribuciones significativas al campo del reconocimiento facial.

Apéndice C

Informes de avance

Actividades programadas

Actividades programadas:

1. Elaboración del Marco Teórico y revisión del Estado del Arte.
2. Recopilación de datos necesarios para el entrenamiento y funcionamiento de la Red Neuronal Convolucional y el optimizador PSO.
3. Consolidación de Datasets para uso en Sintonizador
4. Desarrollo del Sintonizador de Hiperparámetros basado en Enjambres (PSO).
5. Implementación de funcionalidades adicionales / secundarias.
6. Consolidación de Datasets para Reconocimiento Facial
7. Desarrollo de una Red Neuronal Convolucional para el reconocimiento facial.
8. Sintonización de la Red Neuronal construida para consolidación de la librería.
9. Perfeccionamiento del proyecto.
10. Estimación de esfuerzo y asignación de tareas.
11. Configuración de entornos de desarrollo.
12. Implementación de pruebas unitarias y de integración.
13. Pruebas de funcionamiento, verificación de pruebas implementadas e implementación de pruebas adicionales, corrección de errores.
14. Evaluación del proyecto.
15. Elaboración de la documentación requerida y necesaria.
16. Preparación de avances, entrega y presentación.

Marco Teórico e Investigación
Desarrollo y preparación de productos
Marco Teórico e Investigación
Aseguramiento de Calidad de productos

C.1. Informe de avance semana 5

Actividad	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	Esperado	Logrado
1	x	x	x	x														100 %	100 %
2			x	x														80 %	50 %
3				x	x							x						100 %	100 %
4					x	x	x	x	x	x								10 %	10 %
5													x	x	x			0 %	0 %
6				x	x	x					x							60 %	60 %
7				x	x	x	x											70 %	60 %
8											x	x	x					0 %	0 %
9														x	x			0 %	0 %
10	x																	100 %	100 %
11		x	x															100 %	100 %
12					x	x	x	x	x	x	x							20 %	0 %
13											x	x						0 %	0 %
14															x	x		0 %	0 %
15		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		40 %	35 %
16						x					x					x	x	0 %	0 %
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	Esperado	Logrado

Cuadro C.1: Cronograma

Logros y evidencias

1. Investigación y documentación general sobre reconocimiento facial
   Anexo B.Artículos
   Capítulo 2.Reconocimiento facial
2. Investigación y documentación general sobre sintonizadores de hiperparámetros
   Anexo B.Artículos
   Capítulo 1.Sintonización de hiperparámetros
3. Síntesis y resumen de investigaciones realizadas
   Anexo B.Artículos
4. Investigación y apropiación de bases de datos para reconocimiento facial
   Capítulo 2.Reconocimiento facial
5. Investigación y apropiación de proyecto base de CNN
   Capítulo 1.Sintonización de hiperparámetros
6. Desarrollo base de proyecto de sintonizador para CNN con Keras Tuner
   Capítulo 1.Sintonización de hiperparámetros

Problemas y acciones

1. *Problema actividad 7:* Carencia de recursos computacionales para lograr la compilación de modelos
   Acción: Compra de créditos en Colab
2. *Problema actividad 7:* Bajo rendimiento de los modelos
   Acción: Investigación de implementaciones alternativas

C.2. Informe de avance semana 10

Actividad	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	Esperado	Logrado
1	x	x	x	x														100 %	100 %
2			x	x														80 %	80 %
3				x	x							x						100 %	100 %
4					x	x	x	x	x	x								100 %	100 %
5													x	x	x			0 %	0 %
6				x	x	x					x							60 %	60 %
7				x	x	x	x											70 %	60 %
8											x	x	x					0 %	0 %
9														x	x			0 %	0 %
10	x																	100 %	100 %
11		x	x															100 %	100 %
12					x	x	x	x	x	x	x							95 %	95 %
13											x	x						0 %	0 %
14															x	x		0 %	0 %
15		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		55 %	55 %
16						x					x					x	x	25 %	25 %
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	Esperado	Logrado

Cuadro C.2: Cronograma

Logros y evidencias

1. Investigación y documentación general sobre reconocimiento facial

Anexo B.Artículos
Capítulo 2.Reconocimiento facial

2. Investigación y documentación general sobre sintonizadores de hiperparámetros
   Anexo B.Artículos
   Capítulo 1.Sintonización de hiperparámetros
3. Investigación y apropiación de bases de datos para Reconocimiento facial
   Capítulo 2.Reconocimiento facial
4. Investigación y apropiación de proyecto base de CNN
   Capítulo 3.Desarrollo del sintonizador de hiperparámetros PSOT
5. Desarrollo base de proyecto de sintonizador para CNN con Keras Tuner
   Capítulo 3.Desarrollo del sintonizador de hiperparámetros PSOT

Problemas y acciones

1. *Problema actividad 7:* Bajo desempeño de los modelos después de la regularización
   Acción : Experimentación con diferentes versiones de bases de datos
2. *Problema actividad 7:* Poca información respecto a resultados
   Acción : Implementación de gráficos y matrices de confusión

C.3. Informe de avance semana 17

Actividad	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	Esperado	Logrado
1	x	x	x	x														100 %	100 %
2			x	x														100 %	100 %
3				x	x							x						100 %	100 %
4					x	x	x	x	x	x								100 %	100 %
5													x	x	x			100 %	100 %
6				x	x	x					x							100 %	100 %
7				x	x	x	x											100 %	100 %
8											x	x	x					100 %	100 %
9														x	x			100 %	100 %
10	x																	100 %	100 %
11		x	x															100 %	100 %
12					x	x	x	x	x	x	x							100 %	100 %
13											x	x						100 %	100 %
14															x	x		100 %	100 %
15		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x		100 %	100 %
16						x					x					x	x	100 %	100 %
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	Esperado	Logrado

Cuadro C.3: Cronograma

Logros y evidencias

1. Documentación general
   Anexo B.Artículos
   Capítulo 1.Sintonización de hiperparámetros
   Capítulo 2.Reconocimiento facial
2. Experimentación independiente y de focos combinados
   Capítulo 3.Desarrollo del sintonizador de hiperparámetros PSOT

Capítulo 4.Desarrollo de una red neuronal convolucional para reconocimiento facial

3. Obtención de resultados
   Capítulo 3.Desarrollo del sintonizador de hiperparámetros PSOT
   Capítulo 4.Desarrollo de una red neuronal convolucional para reconocimiento facial
   Capítulo Conclusiones y trabajo futuro

Problemas y acciones

1. *Problema actividad 4:* Carencia de recursos computacionales para lograr la compilación de modelos
   Acción: Compra de créditos en Colab
2. *Problema actividad 4:* Estandarización a herramientas conocidas, como Keras Tuner
   Acción: Refactorización general del proyecto
3. *Problema actividad 8:* Obtención de resultados muy lento y demorado
   Acción: Compra de créditos en Colab
   Acción: Trabajo de horas adicionales,
4. *Problema actividad 10:* Problemas al sintonizar la VGG
   Acción: Creación de hipermodelos
   Acción: Refactorización en la forma de obtención de datos

Apéndice D

Actas de reuniones

D.1. Semana 1

1. S01: Enriquecer estado de arte. Previsto: S02. Real: S02
2. S01: Ideas para perfeccionar formulación de proyecto. Previsto: S02. Real: S02
3. S01: Iniciar protocolos de investigación. Previsto: S02. Real: S02
4. S01: Preparar formato control de tiempos. Previsto: S02. Real: S03

D.2. Semana 2

1. S02: Resúmenes de Artículos. Previsto: S03. Real: S03
2. S02: Documento de Revisión Reconocimiento Facial, por Sebastián Rojas. Previsto: S03. Real: S03
3. S02: Documento de Revisión Sintonizadores, por Santiago Rocha. Previsto: S03. Real: S03
4. S02: Corrección al documento de inicio. Previsto: S03. Real: S03
5. S02: Corregir Bibliografía. Previsto: S03. Real: S03

D.3. Semana 3

1. S03: Corrección de Documento de Revisión Reconocimiento Facial, por Sebastián Rojas. Previsto: S04. Real: S04
2. S03: Corrección de Documento de Revisión Sintonizadores, por Santiago Rocha. Previsto: S04. Real: S04
3. S03: Corrección de Cronograma y Actividades. Previsto: S04. Real: S04
4. S03: Corrección del documento de inicio. Previsto: S04. Real: S04
5. S03: Inclusión del documento de inicio en Overleaf. Previsto: S04. Real: S04
6. S03: Preparación de Base de Datos CNN. Previsto: S04. Real: S04
7. S03: Preparación de Base de Datos FR. Previsto: S04. Real: S04
8. S03: Construcción de Software: Sintonización de CNN con Keras Tuner. Previsto: S04. Real: S04
9. S03: Construcción de Software: CNN de Reconocimiento Facial en Keras. Previsto: S04. Real: S05

D.4. Semana 4

1. S04: Sintonizador: Software: Documentar, Sintonizar y Comparar. Previsto: S05. Real: S05
2. S04: Sintonizador: Investigación: Hiperparámetros y Capas. Previsto: S05. Real: S05
3. S04: Sintonizador: DB: Buscar DB con Color / con Ruido. Previsto: S05. Real: S05
4. S04: Sintonizador: Artículo: Corrección de Revisión. Previsto: S05. Real: S07
5. S04: Reconocimiento F: BD: Recolección de Imágenes Personales. Previsto: S05. Real: S05
6. S04: Reconocimiento F: Software: Avances en Arquitectura. Previsto: S05. Real: S05
7. S04: Presentación: Consolidación de la presentación. Previsto: S05. Real: S05
8. S04: Preparación de BD. Previsto: S05. Real: S05

D.5. Semana 5

1. S05: Sintonizador: Software: Implementar más sintonizadores. Real: S06
2. S05: Sintonizador: Investigación: Investigación sobre hiperparámetros. Real: S06
3. S05: Reconocimiento F: Software: Implementación de los modelos VGGs. Real: S06
4. S05: Reconocimiento F: Investigación: Investigación sobre modelos. Real: S06
5. S05: Reconocimiento F: Bases de datos: Adaptación de base de datos según restricciones. Real: S06
6. S05: General: General: Aclaración y clasificación de hiperparámetros: S06

D.6. Semana 6

1. S06: Reconocimiento F: Software: Adaptación de recursos. Real: S07
2. S06: Reconocimiento F: Software: Implementación de los modelos VGGs. Real: S07
3. S06: Sintonizador: Software: Trabajo en *framework*. Previsto: S10. Real: S10
4. S06: Sintonizador: BD: Revisión de Bases de Datos. Previsto: S07. Real: S07

D.7. Semana 7

1. S07: Reconocimiento F: Software: Experimentación con versiones de bases de datos. Real: S08
2. S07: Reconocimiento F: Software: Comparación entre modelos: S08
3. S07: Sintonizador: Software: Trabajo en *framework*. Previsto: S10. Real: S10
4. S07: Sintonizador: Investigación: Definición de Hiperparámetros. Previsto: S08. Real: S08

D.8. Semana 8

1. S08: Reconocimiento F: Software: Experimentación con versiones de bases de datos. Real: S09
2. S08: Reconocimiento F: Software: Comparación entre modelos: S08
3. S08: Reconocimiento F: Software: Regularización de modelos: S08
4. S08: Sintonizador: Software: Refactor y replanteamiento del *framework*. Previsto: S09. Real: S10
5. S08: Sintonizador: Investigación: Pyswarms, uso, hiperparámetros. Previsto: S09. Real: S09
6. S08: General: Documentación: Corrección y mejora de documentación. Previsto: S09. Real: S09

D.9. Semana 9

1. S09: Reconocimiento F: Software: Implementación de los modelos VGGs, experimentación. Real: S10
2. S09: Reconocimiento F: Documento de avance: Elaboración del documento: S10
3. S09: Reconocimiento F: Software: Elaboración de la matriz de confusión: S10
4. S09: Sintonizador: Software: Trabajo en *framework*. Previsto: S10. Real: S10

5. S09: Sintonizador: Software: Nueva refactorización y diferente uso de Pyswarms. Previsto: S10. Real: S10
6. S09: Sintonizador: Investigación: GlobalBestPSO, uso, personalización. Previsto: S10. Real: S10
7. S09: General: General: Aclaración y clasificación de hiperparámetros: S10
8. S09: General: Documentación: Corrección y mejora de documentación. Previsto: S10. Real: S10

D.10. Semana 10

1. S10: Reconocimiento F: Software: Creación del *framework*. Real: S11
2. S10: Reconocimiento F: Software: Experimentación variando hiperpárametros. Real: S11
3. S10: Reconocimiento F: Software: Renovación de licencia. Real: S11
4. S10: Reconocimiento F: Software: Comparación entre modelos: S11
5. S10: Reconocimiento F: Documento de avance: Asignación de referencias e integración al documento: S11
6. S10: Sintonizador: Software: Resultados del *framework*. Previsto: S11. Real: S11
7. S10: Sintonizador: Software: Ajustes del *framework*, reconstrucción del modelo. Previsto: S11. Real: S11
8. S10: General: Documentación: Corrección y mejora de documentación. Previsto: S11. Real: S11

D.11. Semana 11

1. S11: Sintonizador: Software: Refactorización: Paralelización. Previsto: S12. Real: S12
2. S11: Sintonizador: Software: Ajustes: Resultados de validación. Previsto: S12. Real: S12
3. S11: Reconocimiento F: Software: Implementación alternativa al *framework*. Previsto: S12. Real: S12
4. S11: Reconocimiento F: Software: Implementación de métodos de liberación de GPU. Previsto: S12. Real: S12
5. S11: Reconocimiento F: Diagramación: Elaboración de Diagrama de Conceptos. Previsto: S12. Real: S12
6. S11: General: Presentación: Elaboración y preparación de presentación. Previsto: S11. Real: S11

D.12. Semana 12

1. S12: Sintonizador: Software: Refactorización: Mejora en partículas y Mejora de modelos generados. Previsto: S13. Real: S13
2. S12: Sintonizador: Software: Refactorización: Implementación de Memoria. Previsto: S14. Real: S15
3. S12: Sintonizador: Software: Refactorización: Minimización de Reprocesos. Previsto: S13. Real: S14
4. S12: Sintonizador: Software: Experimentación: Verificación y experimentación con CIFAR10. Previsto: S13. Real: S13

5. S12: Reconocimiento F: Software: Preparación para el conjunto de validación: S13. Real: S13
6. S12: Reconocimiento F: Software: Experimentación: Variación de parámetros según el marco metodológico. Previsto: S13. Real: S13
7. S12: S11: Reconocimiento F: Software: Ajuste de la matriz de confusión. Previsto: S13. Real: S13

D.13. Semana 13

1. S13: Sintonizador: Software: Refactorización: Mejora en partículas y mejora de modelos generados. Previsto: S14. Real: S14
2. S13: Sintonizador: Teoría: Investigación: Investigación sobre heurísticos en CNNs. Previsto: S14. Real: S14
3. S13: Sintonizador: Software: Refactorización: Implementación de heurísticos. Previsto: S14. Real: S14
4. S13: Sintonizador: Software: Experimentación: Verificación y experimentación, obtención de resultados de validación. Previsto: S14. Real: S14
5. S13: Sintonizador: Teoría: Evidencia: Tabla comparativa de hiperparámetros sintonizados. Previsto: S14. Real: S14
6. S13: Reconocimiento F: Documento: Correcciones y redacción de progresos. Previsto: S13. Real: S13
7. S13: Reconocimiento F: Software: Experimentación: Variación de parámetros según el marco metodológico para el modelo con transferencia de aprendizaje. Previsto: S14. Real: S14
8. S13: Reconocimiento F: Software: Experimentación: Análisis y desarrollo de la variación del modelo con una clase adicional. Previsto: S14. Real: S14

D.14. Semana 14

1. S14: Sintonizador: Software: Refactorización: Implementación de memoria. Previsto: S14. Real: S14
2. S14: Sintonizador: Software: Refactorización: Implementación de heurísticos para simplificar la representación en partícula. Previsto: S15. Real: S15
3. S14: Sintonizador: Investigación: Consolidación: Actualización de tablas, contenidos, información en Notebooks. Previsto: S15. Real: S15

D.15. Semana 15

1. S15: Sintonizador: Software: Refactorización: Corrección e integración de el mejor modelo a la memoria. Previsto: S16. Real: S16
2. S15: Sintonizador: Software: Refactorización: Mejora de representación de la partícula. Previsto: S16. Real: S16
3. S15: Sintonizador: Software: Refactorización: Claridad de código y documentación. Previsto: S16. Real: S16

4. S15: Sintonizador: Software: Refactorización: Implementación y corrección de heurísticos. Previsto: S16. Real: S16

D.16. Semana 16

1. S16: Sintonizador: Software: Refactorización: De métodos sueltos a *framework*. Previsto: S17. Real: S17
2. S16: Sintonizador: Software: Validación: Resultados previos con MNIST y CIFAR10. Previsto: S17. Real: S17
3. S16: General: Presentación: Elaboración, preparación y práctica de presentación y diapositivas. Previsto: S17. Real: S17
4. S16: Reconocimiento F: Software: Mejoras recomendadas. Previsto: S17. Real: S17
5. S16: Reconocimiento F: Software: Carga de datos directamente y pruebas. Previsto: S17. Real: S17
6. S16: Reconocimiento F: Documento: Corrección de documento. Previsto: S17. Real: S17

D.17. Semana 17

1. S17: Sintonizador: Software: Refactorización: Finalización del refactorización, adaptado a arquitectura Keras Tuner. Previsto: S18. Real: S18
2. S17: Sintonizador: Software: Resultados: Obtención de resultados con MNIST y CIFAR10. Previsto: S18. Real: S18
3. S17: S16: Reconocimiento F:: Software: Refactorización: Implementación de hipermodelos. Previsto: S18. Real: S18
4. S17: Reconocimiento F: Software: Resultados: Pruebas con y sin Generadores. Previsto: S18. Real: S18
5. S17: Reconocimiento F: Software: Software: Preparación de la librería para la integración. Real: S18
6. S17: General: Presentación: Resumen: Preparación, corrección y establecimiento del resumen. Previsto: S18. Real: S18
7. S17: General: Presentación: Diapositivas: Preparación, corrección y establecimiento de diapositivas. Previsto: S18. Real: S18
8. S17: General: Presentación: Preparación: Práctica de presentación. Previsto: S18. Real: S18

D.18. Semana 18

1. S18: Sintonizador - Reconocimiento F: Software: Resultados: Obtención de resultados con VGGFace2. Previsto: S19. Real: S19
2. S18: Sintonizador: Documento: Documentación: Redacción de Capítulo del Libro sobre Sintonizadores. Previsto: S19. Real: S19
3. S18: Sintonizador: Documento: Documentación: Ajustes de Redacción y actualización de resultados. Previsto: S19. Real: S19
4. S18: Reconocimiento F: Documento: Documentación: Redacción de artículo. Previsto: S19. Real: S19

5. S18: Reconocimiento F: Documento: Documentación: Ajustes de Redacción y actualización de resultados. Previsto: S19. Real: S19

Índice de figuras

Índice de cuadros

Bibliografía

[1] Jason Brownlee. How to perform face recognition with vggface2in keras. `https://machinelearningmastery.com/how-to-perform-face-recognition-with-vggface2-convolutional-neural-network-in-keras/`, 08 2020.

[2] Andre Budiman, Fabian, Ricky Aryatama Yaputera, Said Achmad, and Aditya Kurniawan. Student attendance with face recognition (lbph or cnn): Systematic literature review. *Procedia Computer Science*, 216:31–38, 2023. 7th International Conference on Computer Science and Computational Intelligence 2022.

[3] Fernando Sancho Caparrini. Multilayer perceptron. `http://www.cs.us.es/~fsancho/Modelos/ANN.html`. Accessed on 2021-09-20.

[4] Md. Tahmid Hasan Fuad, Awal Ahmed Fime, Delowar Sikder, Md. Akil Raihan Iftee, Jakaria Rabbi, Mabrook S. Al-Rakhami, Abdu Gumaei, Ovishake Sen, Mohtasim Fuad, and Md. Nazrul Islam. Recent advances in deep learning techniques for face recognition. *IEEE Access*, 6:32, 07 2021.

[5] Guosheng Hu, Yongxin Yang, Dong Yi, Josef Kittler, William Christmas, Stan Z. Li, and Timothy Hospedales. When face recognition meets with deep learning: an evaluation of convolutional neural networks for face recognition, 2015.

[6] Franklin Johnson, Alvaro Valderrama, Carlos Valle, Broderick Crawford, Ricardo Soto, and Ricardo Ñanculef. Automating configuration of convolutional neural network hyperparameters using genetic algorithm, 2020.

[7] Shaoqing Ren Jian Sun Kaiming He, Xiangyu Zhang. Deep residual learning for image recognition. *arXiv*, 12 2015.

[8] Tom Lawrence, Li Zhang, Chee Peng Lim, and Emma-Jane Phillips. Particle swarm optimization for automatically evolving convolutional neural networks for image classification, 2021.

[9] James McDermott. Hands-on transfer learning with keras and the vgg16 model, 2015.

[10] Leandro Nunes. Fundamentals of natural computing, June 2 2006.

[11] Omkar M. Parkhi, Andrea Vedaldi, and Andrew Zisserman. Deep face recognition. In *British Machine Vision Conference*, 2015.

[12] DIMA RODIONOV. Vggface2, 2022.

[13] Daniel Smilkov and Shan Carter. Tensorflow playground. `https://playground.tensorflow.org/`. Accessed on 2021-09-20.

[14] Muneeb ul Hassan. Vgg16 – convolutional network for classification and detection, 2018.

[15] Vladyslav Yaloveha, Andrii Podorozhniak, and Heorhii Kuchuk. Convolutional neural network hyperparameter optimization applied to land cover classification, 2022.

[16] Tong Yu and Hong Zhu. Hyper-parameter optimization: A review of algorithms and applications, 2020.