



Máster en Ingeniería Informática
Aplicaciones Avanzadas de la Inteligencia
Artificial 2024-2025
Grupo 1

Práctica 2
“IA en la salud”

Jorge Ramos Santana – 100451001
Luis Daniel Casais Mezquida – 100429021
Carlos Salinas Gancedo – 100549334

Equipo 1

Profesor
María Paula de Toledo

Índice

1. Introducción y enfoque del problema	3
1.1. Antecedentes	3
2. Metodología	3
2.1. Decisiones de diseño e implementación	3
2.2. Tipo de clasificador	5
2.3. Arquitectura de la red.	5
2.4. Estrategia de entrenamiento	6
3. Implementación y recursos	6
3.1. Hardware utilizado	6
4. Experimentos y resultados	7
4.1. Proceso de experimentación	7
4.2. Tiempo de entrenamiento	7
4.3. Análisis de contaminación de datos	8
4.4. Resultados de clasificación.	8
4.5. Análisis de interpretabilidad (GradCAM)	9
5. Modelo Multiclasé	9
6. Conclusiones	10
6.1. Trabajos futuros	10
I Bibliografía	12

Índice de figuras

1	Ejemplo de preprocesamiento de imágenes.	4
2	Ejemplo de aumentación de datos.	5
3	Curvas de entrenamiento y validación del modelo.	7
4	Ejemplo de visualización GradCAM en una imagen de rayos X (Clase Normal).	9

1. Introducción y enfoque del problema

En esta práctica abordamos el desarrollo de un clasificador de imágenes de rayos X para la detección de COVID-19. El objetivo de nuestro trabajo es desarrollar un modelo de clasificación basado en deep learning capaz de diferenciar entre radiografías torácicas de pacientes con COVID-19 y pacientes sanos (normales). Para ello, utilizaremos técnicas de redes neuronales convolucionales (CNN) que han demostrado un gran rendimiento en tareas similares de clasificación de imágenes médicas.

1.1. Antecedentes

Desde el inicio de la pandemia, numerosos grupos de investigación han explorado el uso de técnicas de deep learning para la detección de COVID-19 en imágenes médicas [1] [2]. Los enfoques basados en transfer learning con arquitecturas pre-entrenadas como ResNet, DenseNet o EfficientNet han sido ampliamente utilizados. Estos modelos aprovechan el conocimiento adquirido en grandes conjuntos de datos de imágenes generales y lo adaptan a la tarea específica de clasificación de radiografías.

2. Metodología

2.1. Decisiones de diseño e implementación

Para el desarrollo de nuestro clasificador de COVID-19 en radiografías, hemos tomado las siguientes decisiones de diseño:

2.1.1. Lenguaje de programación y frameworks

Seleccionamos Python como lenguaje de programación debido a su popularidad y amplio ecosistema en el ámbito del aprendizaje automático. Entre las principales bibliotecas y frameworks utilizados destacan:

- **PyTorch:** Framework de deep learning muy popular desarrollado por Facebook. Lo elegimos por su flexibilidad, facilidad de depuración y soporte para la computación acelerada por hardware (GPU/MPS).
- **scikit-learn:** Para métricas de evaluación y herramientas de análisis de resultados.
- **Matplotlib y NumPy:** Para visualización de datos y operaciones matriciales eficientes.

2.1.2. Preprocesamiento de imágenes

El preprocesamiento de las imágenes de rayos X es crucial para mejorar el rendimiento del modelo. En nuestro caso, aplicamos las siguientes transformaciones:

- **Conversión a escala de grises:** Al tratarse de radiografías, trabajamos con un único canal para reducir la complejidad del modelo.
- **Recorte central:** Aplicamos un recorte centrado al 70 % de la imagen para eliminar bordes y centrarnos en la región torácica.
- **Normalización:** Los valores de píxeles se normalizan en el rango $[-1, 1]$.
- **Aumentación de datos** (solo para entrenamiento): Incluimos transformaciones aleatorias como rotaciones (± 15 grados) y cambios de escala (5-15 %) para mejorar la capacidad de generalización del modelo. No incluimos flip horizontal ya que no tiene sentido en el contexto médico.

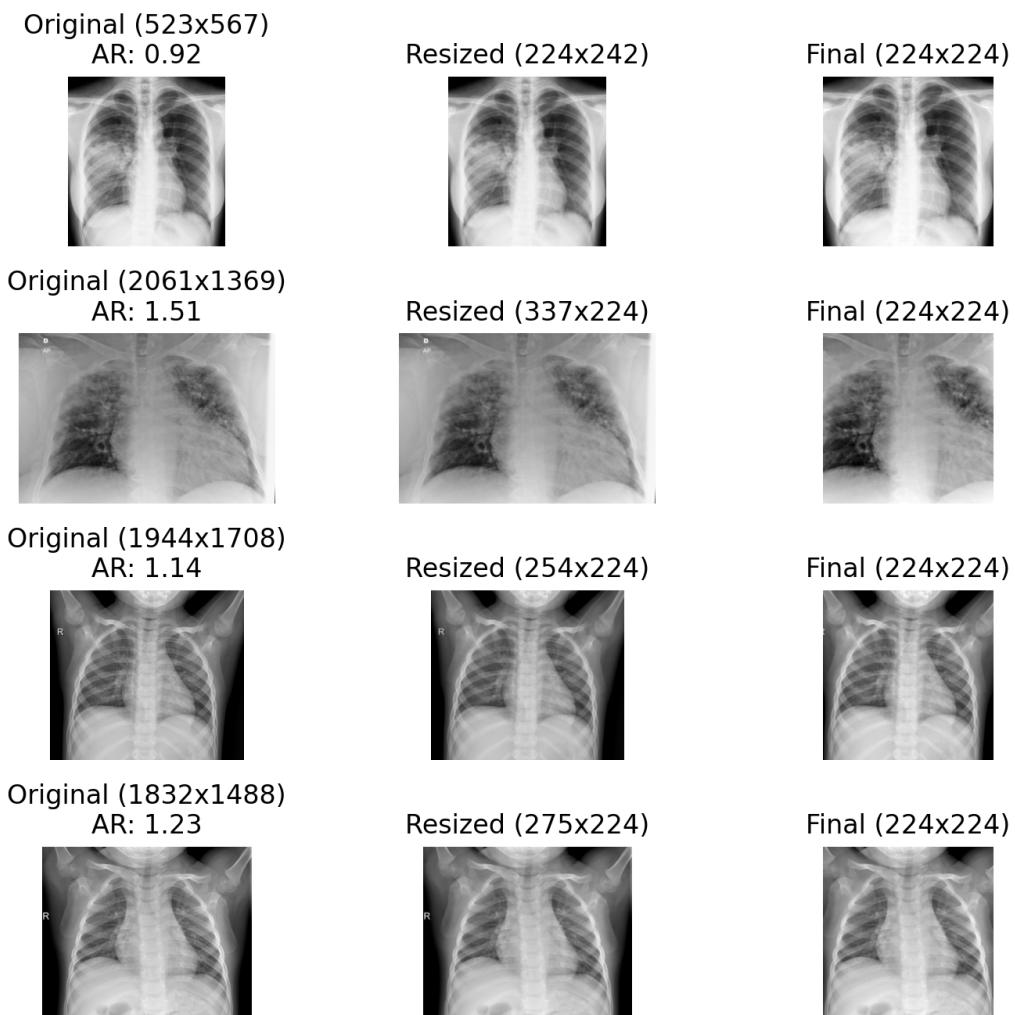


Fig. 1: Ejemplo de preprocesamiento de imágenes.

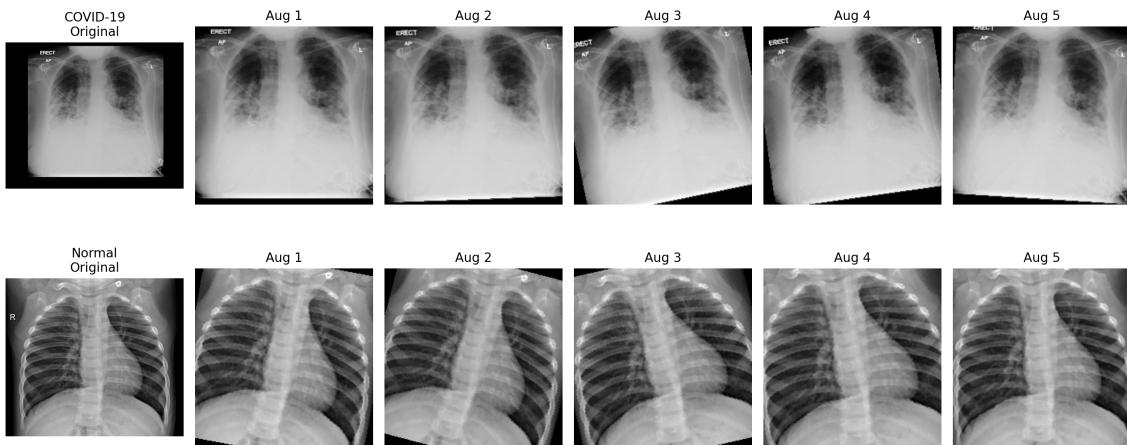


Fig. 2: Ejemplo de aumento de datos.

2.2. Tipo de clasificador

Después de experimentar con diversas arquitecturas, optamos por una Red Neuronal Convolutacional (CNN) personalizada para nuestro clasificador. Este enfoque se basó en varios factores. Las pruebas iniciales con modelos preentrenados como DenseNet-169 y ResNet-18 mediante transfer learning revelaron que estos modelos eran excesivamente complejos para la tarea, lo que resultaba en un aprendizaje demasiado rápido y potenciales problemas de overfitting. Por tanto, decidimos implementar una CNN más simple y entrenarla desde cero para adaptarla mejor a las características del problema.

2.3. Arquitectura de la red

Nuestra CNN consta de los siguientes componentes:

- **Capas convolucionales:** Cinco bloques convolucionales, cada uno compuesto por:
 - Una capa de convolución 2D con kernel de tamaño 3×3 y padding de 1
 - Batch Normalization [3] para estabilizar el aprendizaje
 - Función de activación ReLU
 - Max Pooling con kernel de tamaño 2×2 y stride de 2
- **Canales de características:** La red aumenta progresivamente el número de filtros:
 $1 \rightarrow 32 \rightarrow 64 \rightarrow 128 \rightarrow 256 \rightarrow 256$
- **Clasificador:** Tras las capas convolucionales, incluimos:
 - Aplanamiento (Flatten) de las características extraídas
 - Una capa completamente conectada con 512 neuronas y activación ReLU

- Capa de Dropout con probabilidad de 5 para reducir el overfitting
 - Capa de salida con 2 neuronas (una por clase)
- **Función de pérdida:** Cross Entropy Loss, adecuada para problemas de clasificación multiclase
 - **Regularización:** Además del Dropout mencionado, utilizamos Batch Normalization en cada bloque convolucional para estabilizar el entrenamiento y reducir el overfitting

2.4. Estrategia de entrenamiento

Implementamos las siguientes estrategias para el entrenamiento de nuestro modelo:

- **Número máximo de epochs:** 30, con early stop basada en la métrica F1 del conjunto de validación
- **Tamaño de mini-batch:** 32 imágenes
- **Inicialización de pesos:** Inicialización aleatoria por defecto de PyTorch
- **Optimizador:** Adam con learning rate de 0.001
- **Parada temprana:** Implementamos un mecanismo de early stop con paciencia de 7 epochs y una delta mínima de 0.001 en la puntuación F1 de validación
- **División de datos:** 80 % para entrenamiento y 20 % para validación

Utilizamos Adam como optimizador debido a su capacidad para adaptarse a diferentes tasas de aprendizaje y su eficiencia en la convergencia frente a otros optimizadores más clásicos como SGD.

3. Implementación y recursos

3.1. Hardware utilizado

El entrenamiento del modelo se realizó en un MacBook Pro de 14 pulgadas con las siguientes especificaciones:

- Procesador: Apple M3 Max 16 cores (12 performance and 4 efficiency). GPU de 40 cores.
- Memoria RAM unificada: 128 GB
- Aceleración por hardware: Metal Performance Shaders (MPS)

4. Experimentos y resultados

4.1. Proceso de experimentación

Nuestro proceso de experimentación siguió un enfoque iterativo:

1. **Transfer learning:** Evaluamos modelos preentrenados mediante transfer learning, concretamente DenseNet-169 y ResNet-18. Observamos que estos modelos convergían demasiado rápido, lo que sugería que eran excesivamente complejos para la tarea o que podían existir problemas en los datos.
2. **Red personalizada:** Decidimos implementar una CNN personalizada más simple y entrenarla desde cero, lo que nos dio mayor control sobre la arquitectura y permitió ajustarla mejor a las características del problema. Sin embargo, también observamos un aprendizaje rápido, lo que nos llevó a sospechar de la calidad de los datos.
3. **Fase de análisis:** Implementamos técnicas de interpretabilidad visual (GradCAM) para entender en qué regiones de la imagen se estaba centrando el modelo para tomar sus decisiones.

4.2. Tiempo de entrenamiento

El entrenamiento del modelo final se completó en 16 minutos y 19 segundos. El early stop se activó después de 27 epochs al no observarse mejoras significativas en la métrica F1 del conjunto de validación durante 7 epochs consecutivos.

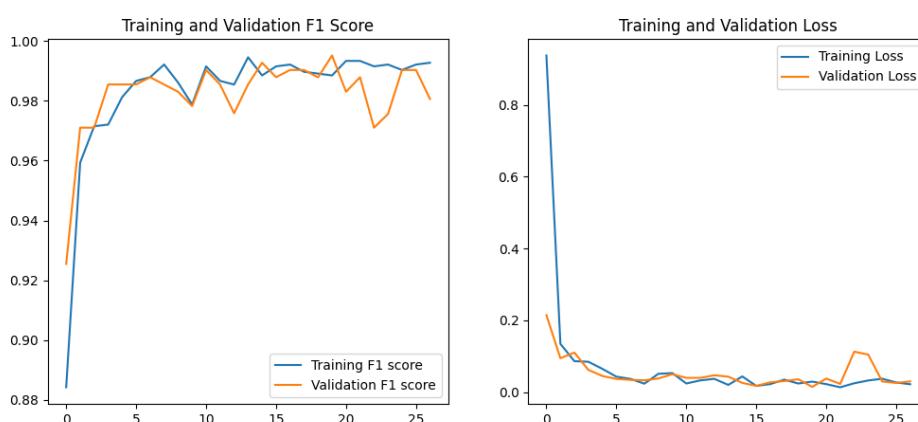


Fig. 3: Curvas de entrenamiento y validación del modelo.

4.3. Análisis de contaminación de datos

Durante la evaluación de checkpoints intermedios, notamos que los resultados eran excepcionalmente altos, lo que nos llevó a investigar la calidad del conjunto de datos. A través de un análisis visual de las imágenes, descubrimos que en el conjunto de COVID-19, algunas contenían anotaciones o marcas indicando la presencia de áreas de interés.

Ya que si limpiabamos estas imágenes con anotaciones el conjunto de COVID-19 se reduciría considerablemente, decidimos utilizar no el dataset reducido que se nos proporcionó, sino el dataset completo, que contiene un mayor número de imágenes. Sin embargo, este conjunto de datos también presentaba problemas de calidad. Por ejemplo, algunas imágenes estaban borrosas o mal alineadas, lo que podría afectar negativamente al rendimiento del modelo. Había imágenes con diferentes resoluciones y formatos, lo que complicaba el preprocesamiento y la normalización de los datos. Además, las radiografías del tórax estaban tomadas en dos posiciones diferentes (frontal y lateral), mientras que las del conjunto normal eran todas frontales. También había alguna imagen que no se trataba de una radiografía, sino de una resonancia magnética, que podría haber influido en el rendimiento del modelo.

Viendo esto, decidimos hacer una limpieza del conjunto de datos, eliminando **manualmente** las imágenes contaminadas y aquellas que no cumplían con los criterios de calidad. Con estos cambios se redujo el número de imágenes del conjunto de COVID-19, pero aumentó la calidad del conjunto de datos.

4.4. Resultados de clasificación

El modelo final obtuvo los siguientes resultados en el conjunto de validación:

- **Exactitud (Accuracy):** 98 %
- **F1-Score:** 99.51 %
- **Área bajo la curva ROC (AUC-ROC):** 0.9996
- **Sensibilidad (Recall para la clase COVID-19):** 97.99 %
- **Matriz de confusión:**

	Predicho COVID-19	Predicho Normal
Real COVID-19	244	5
Real Normal	3	160

Estos resultados muestran un rendimiento excepcionalmente alto, con casi una clasificación perfecta, lo cual podría sugerir un posible overfitting o problemas en el conjunto de datos.

4.5. Análisis de interpretabilidad (GradCAM)

Dado que tras el análisis de los resultados iniciales y la limpieza del conjunto de datos, el modelo parecía aprender patrones no deseados, decidimos implementar técnicas de interpretabilidad para comprender mejor su comportamiento. En este caso, utilizamos GradCAM, que nos permite visualizar las regiones de la imagen que más influyen en la decisión del clasificador.

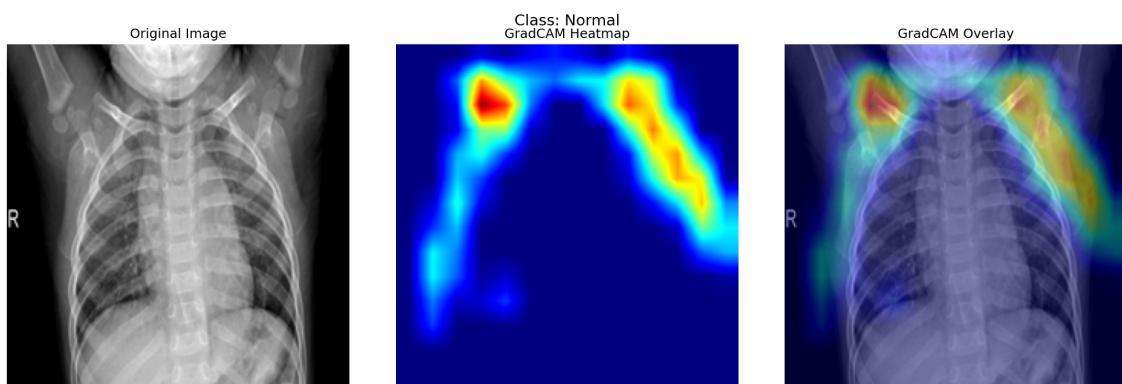


Fig. 4: Ejemplo de visualización GradCAM en una imagen de rayos X (Clase Normal).

En la figura 4 se muestra un ejemplo de la visualización GradCAM aplicada a una imagen normal. La región resaltada en rojo indica la zona de la imagen que el modelo considera más relevante para su decisión. En este caso, el modelo parece centrarse en el área del brazo del paciente, lo que no tiene relación con la patología que se busca clasificar.

Tras analizar varias imágenes, observamos que el modelo tiende a enfocarse en áreas no relevantes, como los brazos de los pacientes o marcas en las radiografías. Esto refuerza la hipótesis de que el modelo ha aprendido patrones no deseados debido a la contaminación del conjunto de datos.

Haciendo un análisis más profundo, nos percatamos de otro problema: en el conjunto de COVID-19, los pacientes tenían los brazos hacia abajo, mientras que en el conjunto normal, los brazos estaban hacia arriba. Esto llevó al modelo a aprender a clasificar las imágenes en función de la posición de los brazos, en lugar de la presencia de COVID-19.

5. Modelo Multiclasificación

Dado que el conjunto de datos estaba contaminado y el modelo no estaba aprendiendo a clasificar correctamente las imágenes, decidimos no continuar con el modelo de clasificación multiclasificación, que incluía además de COVID-19 y normal, la clase de neumonía bacteriana. Los resultados obtenidos en el modelo binario no eran confiables, por lo que

no tenía sentido continuar con un modelo más complejo. Sin embargo, si se hubiera continuado, el modelo multiclase habría seguido la misma arquitectura y metodología que el modelo binario, pero con una capa de salida de 3 neuronas y una función de activación softmax.

6. Conclusiones

El clasificador de COVID-19 basado en imágenes de rayos X desarrollado en este trabajo muestra un rendimiento aparentemente excepcional, pero este resultado extremadamente positivo debe interpretarse con cautela por varios motivos.

El análisis de interpretabilidad mediante GradCAM y el estudio detallado del conjunto de datos reveló problemas de contaminación en los datos, lo que significa que el modelo aprendió patrones artificiales o marcas no relacionadas con la patología. Asimismo, el hecho de que modelos complejos como DenseNet-169 y ResNet-18 aprendieran demasiado rápido fue una señal temprana de posibles problemas con el conjunto de datos. A pesar de las métricas casi perfectas, la utilidad clínica real de este modelo es nula debido a los problemas identificados.

Como bien se dice en el deep learning, los datos son lo más importante. Al tener un conjunto de datos contaminado, el modelo aprendió no a indentificar la presencia de COVID-19, sino a identificar la posición de los brazos de los pacientes. También destaca el valor de las técnicas de interpretabilidad como GradCAM, que nos permiten no solo evaluar el rendimiento numérico de un modelo, sino también comprender su comportamiento y detectar posibles fallos o sesgos en su proceso de toma de decisiones. Gracias a estas técnicas, pudimos identificar que el modelo no estaba aprendiendo a clasificar las imágenes en base a la presencia de la patología, sino de características irrelevantes como la posición de los brazos de los pacientes o marcas en las radiografías.

6.1. Trabajos futuros

Para futuros trabajos, sería recomendable realizar un análisis más exhaustivo del conjunto de datos antes de entrenar el modelo. En nuestro trabajo, la limpieza del conjunto de datos se realizó de manera manual, lo que puede ser poco eficiente y propenso a errores. Una opción sería implementar un proceso automatizado para detectar y eliminar imágenes contaminadas o de baja calidad. Un posible enfoque sería utilizar técnicas de segmentación de imágenes [4] [1] para filtrar únicamente las regiones relevantes de las radiografías, eliminando áreas no relacionadas con la patología.

Además, sería interesante explorar el uso de técnicas de data augmentation más avanzadas, como la generación de imágenes sintéticas mediante GANs (Generative Adversarial Networks) [5], para aumentar la diversidad del conjunto de datos y mejorar la capacidad de generalización del modelo. Utilizando esta técnica, podríamos generar imágenes

de rayos X sintéticas que simulen diferentes condiciones patológicas a partir de radiografías reales, lo que podría enriquecer el conjunto de datos y ayudar al modelo a aprender patrones más robustos.

Otra opción sería utilizar GPTs (Generative Pre-trained Transformers) multimodales para realizar la clasificación de imágenes. Estos modelos han demostrado un rendimiento excepcional en tareas de clasificación y segmentación de imágenes, y podrían ser una alternativa interesante a las CNN tradicionales. Sin embargo, su implementación es más compleja y requiere un mayor esfuerzo computacional. Uno de estos modelos es Qwen2.5-VL, que ha demostrado un rendimiento sobresaliente en tareas de clasificación y segmentación de imágenes [6].

Parte I

Bibliografía

- [1] E. Tartaglione, C. A. Barbano, C. Berzovini, M. Calandri y M. Grangetto, «Unveiling COVID-19 from CHEST X-Ray with Deep Learning: A Hurdles Race with Small Data,» *International Journal of Environmental Research and Public Health*, vol. 17, n.º 18, pág. 6933, sep. de 2020. DOI: [10.3390/ijerph17186933](https://doi.org/10.3390/ijerph17186933). dirección: <http://dx.doi.org/10.3390/ijerph17186933>.
- [2] L. Lara, L. E. Berger y R. Raju, *Diagnosing COVID-19 Severity from Chest X-Ray Images Using ViT and CNN Architectures*, 2025. arXiv: [2502.16622 \[eess.IV\]](https://arxiv.org/abs/2502.16622). dirección: <https://arxiv.org/abs/2502.16622>.
- [3] S. Ioffe y C. Szegedy, *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*, 2015. arXiv: [1502.03167 \[cs.LG\]](https://arxiv.org/abs/1502.03167). dirección: <https://arxiv.org/abs/1502.03167>.
- [4] N. Ravi et al., *SAM 2: Segment Anything in Images and Videos*, 2024. arXiv: [2408.00714 \[cs.CV\]](https://arxiv.org/abs/2408.00714). dirección: <https://arxiv.org/abs/2408.00714>.
- [5] I. J. Goodfellow et al., *Generative Adversarial Networks*, 2014. arXiv: [1406.2661 \[stat.ML\]](https://arxiv.org/abs/1406.2661). dirección: <https://arxiv.org/abs/1406.2661>.
- [6] S. Bai et al., *Qwen2.5-VL Technical Report*, 2025. arXiv: [2502.13923 \[cs.CV\]](https://arxiv.org/abs/2502.13923). dirección: <https://arxiv.org/abs/2502.13923>.