Aprendizaje Estadístico 2025

Determinación del Sistema Cristalino con MobileNetV2

Juliana Del Valle[1],

Instituto de Física, Universidad de Antioquía, Medellín, Colombia.

Resumen

El presente trabajo explora el uso de redes neuronales convolucionales (CNN) para la determinación del grupo. Se utilizaron 87,989 datos del proyecto de materiales, procesados con la biblioteca Pymatgen y transformados en imágenes mediante Matplotlib. Se implementó la arquitectura MobileNetV2 con la técnica de aprendizaje por transferencia y ajuste fino (fine-tuning). La evaluación del modelo arrojó una precisión global del 54 %, sugiriendo posibles mejoras relacionadas con el balance de datos y la calidad del preprocesamiento.

1 Introducción

El principio de Neumann establece que las propiedades físicas de los cristales exhiben las simetrías de la estructura cristalina [1]. Por lo tanto, su determinación es fundamental para el analisis de los sólidos. Una de las técnicas más comunes para su estudio es la difracción de rayos X. En terminos generales, los difractómetros de rayos X se componen de 3 partes: un tubo de rayos X, un porta-muestras y un detector [2]. Donde se varía el ángulo de incidencia de los rayos, para obtener una gráfica de intensidades como se muestra en la Figura 1.

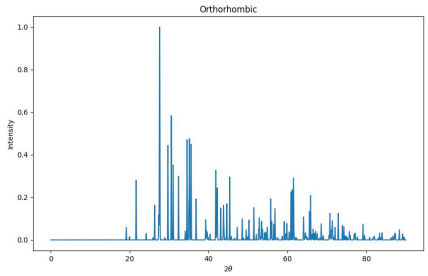

Figura 1: Patrón de Difracción de Rayos X

Este patrón puede entenderse considerando que los rayos X incidentes son reflejados por planos equidistantes [3]. De manera que la diferencia entre las fases de las ondas reflejadas dependen de la distancia entre los planos del sólido. En consecuencia, se puede obtener una condición geométrica para la interferencia constructiva, conocida como ley de Bragg que explica la aparición de los picos en el patrón de difracción (Figura 1). Sin embargo, la ecuación de Bragg no es una condición suficiente para la aparición de picos en el patrón. Otros factores como la simetría del cristal deben ser tomados en cuenta. Por esto, se puede usar el patrón de XRD para determinar la estructura [4]. Proceso que se conoce como "Indexado" del patrón, en el cual se debe comparar el patrón obtenido con el de materiales conocidos [5].

Aun así, el indexado de patrones es una tarea compleja y puede fallar en materiales de baja simetría [6]. En este contexto, Park et al. desarrollaron una Red Neuronal Convolucional (CNN) para identificar el sistema cristalino, el grupo de extinción y el grupo espacial, logrando exactitudes del 94,99 %, 83,83 % y 81,14 %, respectivamente [6]. Por su parte, Oviedo et al. emplearon CNN para predecir la dimensión cristalográfica y el grupo espacial a partir de patrones de XRD de películas delgadas, alcanzando exactitudes del 93 % y 89 %, respectivamente [5].

Siguiendo esta línea, el presente trabajo explora el uso de CNN para la determinación del grupo cristalino, el cual define las simetrías del cristal sin considerar las traslaciones ni las simetrías de los átomos o moléculas que lo componen. Para ello, se reentrena una red neuronal diseñada por Google, conocida como MobileNetV2.

2 Metodología

2.1 Conjunto de Datos

Los datos fueron obtenidos del proyecto de materiales [7], una base de datos online de acceso libre. Lanzada desde el 2011 por el Laboratorio Nacional Lawrence Berkeley (LBNL). Es soporta-

[1]E-mail: juliana.delvalle1@udea.edu.co

da además por el departamento de energía de los estados unidos (DOE), la fundación nacional de ciencias (NSF), entre otros.
A pesar de que esta base de datos tiene alrededor de 144.595 materiales inorgánicos [7]. En este trabajo se utilizaron 87989 materiales, de los cuales 83989 fueron obtenidos de una librería de python asociada al proyecto de materiales llamada matminer [8]. En concreto estos 83989 datos corresponde al conjunto llamado $mp_all_2018018$, es decir a los materiales disponibles en el proyecto en el año 2018. El resto de los materiales obtenidos se obtuvieron a través de la API del proyecto de materiales [7]. Con el fin de incrementar el número de materiales de los sistemas cristalinos con menos materiales en $mp_all_2018018$, lo cuales fueron; el cubico, hexagonal, trigonal y tetragonal.

Tanto en el conjunto de datos de matminer como en el obtenido a través de la API, los materiales se representaban mediante un objeto structure de la biblioteca Pymatgen [9]. Este objeto permite calcular el patrón de difracción del material y determinar su grupo cristalino, para lo cual se requieren dos clases adicionales: SpaceGroupAnalyzer y XRDCalculator, con este último se especifica el tipo de rayos X utilizado para generar el patrón, que en este caso fue CuKa.

Sin embargo, el patrón obtenido con las clases de Pymatgen solo contiene las coordenadas de los picos, por lo que es necesario definir un rango común de ángulos. Para este trabajo, se seleccionó el rango $[0, 90]$ de 2θ y se asignó una intensidad de 0 a los ángulos sin picos asociados. Posteriormente, se aplicó un filtro gaussiano a los datos de intensidad para suavizar los picos. Esto se realizó utilizando *Gaussian_filter1d* de la biblioteca *scipy.ndimage* [10], con un valor de $\sigma = 5$, elegido para garantizar que los picos fueran lo suficientemente suaves sin solaparse con picos vecinos. Finalmente, las intensidades fueron renormalizadas para que estuvieran en el intervalo $[0, 1]$. Finalmente, estos vectores de intensidad fueron utilizados para construir las imágenes asociadas al patrón de difracción mediante *imshow* de la biblioteca de Python *Matplotlib.pyplot* [11]. Cabe resaltar que las imágenes generadas deben ser cuadradas para ser utilizadas en la red neuronal. Posteriormente, estas fueron almacenadas localmente en carpetas asociadas a su sistema cristalino.

2.2 Arquitectura de la CNN

La CNN utilizada fue *MobileNetV2*, la cual pertenece a una clase de modelos diseñados por Google conocidos como *MobileNets* [12]. Estos modelos priorizan la eficiencia de la red, pues están pensados para ser usados en aplicaciones móviles o de visión embebida [13].

La arquitectura de estas redes se basa en las capas de convolución separable en profundidad (*depthwise separable convolution*), que, a diferencia de las capas convolucionales estándar, las cuales aplican los filtros y los combinan para formar la salida en un solo paso, dividen este proceso en dos etapas. Además, estas capas son seguidas de capas *Batch Normalization*, para normalizar las entradas y *Dropout* [13] [12].

En particular, *MobileNetV2* es una versión mejorada de esta arquitectura, propuesta por Google en 2019. Donde se implementa un nuevo tipo de bloque denominado *inverted residual with linear bottleneck*, el cual es una variante de los bloques residuales. En concreto, la arquitectura de *MobileNetV2* contiene una capa de convolución por profundidad con 32 filtros, seguida de 19 bloques residuales con cuello de botella. La función de activación utilizada fue *ReLU6*, y los *kernels* empleados fueron de tamaño 3×3 [12].

Para implementar este algoritmo, se utilizó la técnica de *transfer learning*, que consiste en tomar una red preentrenada, eliminar las últimas capas de clasificación para redefinirlas y posteriormente entrenar el modelo con nuevos. En concreto, *MobileNetV2* puede ser accedido desde el módulo *Keras Applications* [14]. Sin embargo, esta técnica requiere que los datos sobre los que previamente se entrenó la red, en este caso *ImageNet* [15], sean similares a los nuevos datos, lo cual no ocurre con los patrones de difracción. En consecuencia, se optó por entrenar todas las capas del modelo utilizando *fine-tuning*.

3 Resultados

Para cada sistema cristalino se lograron obtener los siguientes materiales de la base de datos (Figuras 3 y 3).

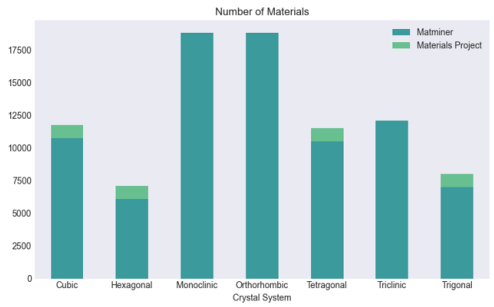

Figura 2: Número de materiales por sistema cristalino.

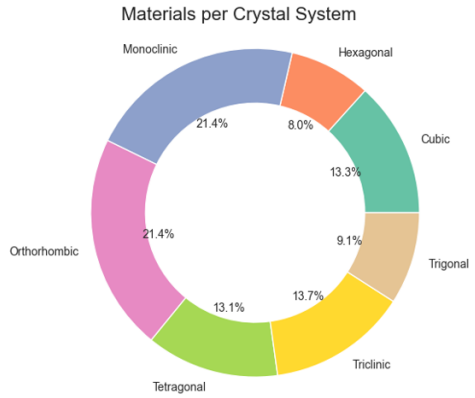

Figura 3: Proporción de materiales por sistema cristalino.

Después del procesamiento de datos, se obtuvieron patrones de difracción como los que se muestran en la Figura 3.

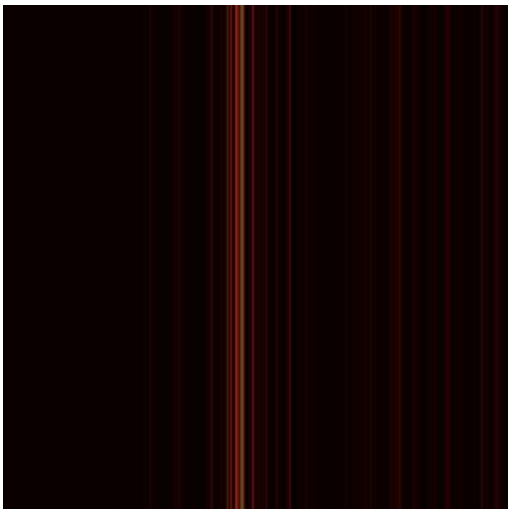

Figura 4: Ejemplo de un patrón de difracción obtenido.

Por último, la matriz de confusión obtenida tras el entrenamiento de *MobileNetV2* con estos patrones se muestra en la Figura **??**.

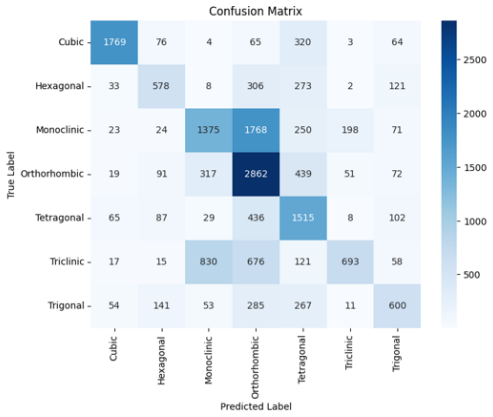

Figura 5: Matriz de confusión del modelo entrenado.

También se obtuvieron las siguientes métricas de evaluación:

System	Precision	Recall	F1-score
Cubic	0.89	0.77	0.83
Hexagonal	0.57	0.44	0.50
Monoclinic	0.53	0.37	0.43
Orthorhombic	0.45	0.74	0.56
Tetragonal	0.48	0.68	0.56
Triclinic	0.72	0.29	0.41
Trigonal	0.55	0.43	0.48
Accuracy	0.54	0.54	0.54

A partir de las anteriores tablas, se observa que la exactitud de la red evaluada sobre el conjunto de validación fue del 54 %. Este resultado puede deberse a múltiples razones. Una posible causa es un error en el código encargado de analizar los datos. También, puede deberse al sesgo en la base de datos, ya que más del 40 % de los datos pertenecen a solo dos sistemas cristalinos, mientras que sistemas como el hexagonal solo contienen un 8 % del total (Figura 3). Esto se refleja en la matriz de confusión (Figura **??**), donde el sistema hexagonal tuvo un desempeño deficiente.

Sin embargo, también se observa que el sistema monoclínico obtuvo un bajo rendimiento a pesar de ser uno de los sistemas con mayor cantidad de datos. Por otro lado, el mejor desempeño se obtuvo con el sistema cúbico, que representaba solo el 13,3 % de los datos. Por lo tanto, la falta de datos por sí sola no explica completamente el rendimiento del modelo.

Otra posible razón es que algunos grupos cristalinos pueden compartir características similares en sus patrones de difracción debido a que presentan simetrías semejantes. Además, podría ser que este tipo de red no sea el más adecuado para este problema. Por ejemplo, [6] utilizó una red convolucional 1D en lugar de una 2D.

Finalmente, el formato de las imágenes generadas podría estar afectando el desempeño del mo-

delo. Es posible que sean demasiado oscuras, omitiendo algunos picos importantes, o que el mapa de colores elegido no sea el más adecuado.

4 Conclusiones

De los resultados obtenidos, se evidencia que este proyecto no logró desarrollar un algoritmo capaz de determinar el sistema cristalino a partir del grupo espacial. Sin embargo, existen precedentes de otros autores, como [6] y [5], que sí han logrado hacerlo, incluso en tareas más complejas, como la determinación del grupo espacial. Por ello, es necesario revisar cada etapa del proyecto y, posiblemente, reducir la complejidad del problema, por ejemplo, acotándolo a ciertos tipos de materiales.

Referencias

[1] Robert E. Newnham. *Properties of Materials: Anisotropy, Symmetry, Structure*. OUP Oxford, 2005. Google-Books-ID: gSkTDAAAQBAJ.

[2] Elena gabriela Udrişţioiu Andrei A. Bunaciu and Hassan Y. Aboul-Enein. X-ray diffraction: Instrumentation and applications. *Critical Reviews in Analytical Chemistry*, 45(4):289–299, 2015. PMID: 25831472.

[3] Jenö Sólyom. *Fundamentals of the Physics of Solids: Volume II: Electronic Properties*. Springer Science & Business Media, November 2008.

[4] Marc De Graef and Michael E. McHenry. *Structure of Materials: An Introduction to Crystallography, Diffraction and Symmetry*. Cambridge University Press, October 2012. Google-Books-ID: NMUgAwAAQBAJ.

[5] Felipe Oviedo, Zekun Ren, Shijing Sun, Charles Settens, Zhe Liu, Noor Titan Putri Hartono, Savitha Ramasamy, Brian L. DeCost, Siyu I. P. Tian, Giuseppe Romano, Aaron Gilad Kusne, and Tonio Buonassisi. Fast and interpretable classification of small X-ray diffraction datasets using data augmentation and deep neural networks. *npj Computational Materials*, 5(1):60, May 2019.

[6] Woon Bae Park, Jiyong Chung, Jaeyoung Jung, Keemin Sohn, Satendra Pal Singh, Myoungho Pyo, Namsoo Shin, and Kee-Sun Sohn. Classification of crystal structure using a convolutional neural network. *IUCrJ*, 4(Pt 4):486–494, July 2017.

[7] Kristin Persson. Materials Project.

[8] Logan Ward, Alexander Dunn, Alireza Faghaninia, N. E. R. Zimmermann, Saurabh Bajaj, Qi Wang, Joseph H. Montoya, Jason Chen, Kyle Bystrom, Maxwell Dylla, Kyle Chard, Mark Asta, Kristin Persson, G. Jeffrey Snyder, Ian Foster, and Anubhav Jain. Matminer: An open source toolkit for materials data mining, 2018.

[9] Shyue Ping Ong, William Davidson Richards, Anubhav Jain, Geoffroy Hautier, Michael Kocher, Shreyas Cholia, Dan Gunter, Vincent Chevrier, Kristin A. Persson, and Gerbrand Ceder. Python materials genomics (pymatgen): A robust, open-source python library for materials analysis. *Computational Materials Science*, 68:314–319, 2013.

[10] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[11] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[12] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks, 2019.

[13] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications, 2017.

[14] François Chollet et al. Keras. https://keras.io, 2015.

[15] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.