PROYECTO INTEGRADOR EN INTELIGENCIA ARTIFICIAL

Análisis Comparativo de Algoritmos

Grupo 6:

Luis Andress Bustamante Jiménez

Damaris Irene Alarcón Vallejo

Docente: Gladys María Villegas Rugel

23 de septiembre de 2025

Análisis Comparativo de Algoritmos

Para el desarrollo del proyecto sobre la clasificación explicable de arritmias cardíacas a partir de electrocardiogramas (ECG) transformados en espectrogramas, resulta fundamental realizar una evaluación comparativa rigurosa entre diferentes arquitecturas de redes neuronales convolucionales (CNNs). La selección del modelo adecuado impacta directamente en el rendimiento diagnóstico, la interpretabilidad clínica, la viabilidad computacional y la posibilidad de despliegue en entornos reales o dispositivos con recursos limitados.

En este trabajo se comparan cinco arquitecturas de CNN representativas y contrastadas en la literatura: ResNet, ShuffleNet, RepVGG, EfficientNet, MobileNet y RepVGG.

Técnica 1: ResNet

1. **Descripción Teórica**

- Fundamentos matemáticos:

ResNet, abreviatura de Residual Networks, introduce bloques residuales que permiten el aprendizaje de funciones residuales en lugar de directamente aprender una función mapeo complicado. Matemáticamente, un bloque residual se implementa de la siguiente manera:

$$y = F(x; W) + x$$

Donde x es la entrada del bloque, F es una secuencia de convoluciones parametrizada por pesos W, y la suma residual añade directamente la entrada. Esto facilita el flujo del gradiente a través de muchas capas, mitigando el problema de gradiente que se desvanece.

- Principio de funcionamiento:

El objetivo principal detrás de las conexiones residuales es facilitar el aprendizaje de funciones identidad o cercanas a esta, permitiendo que ciertas capas simplemente propaguen la información sin modificarla. Esto evita que la red tenga que aprender explícitamente la identidad desde cero, lo cual puede ser difícil en arquitecturas profundas. Gracias a este mecanismo, ResNet logra entrenar redes con gran cantidad de capas (50, 101, 152 o más) manteniendo la estabilidad y eficiencia durante el proceso de optimización.

- Tipo de aprendizaje:

Se trata de aprendizaje supervisado en clasificación de imágenes o representación transformada en imágenes, usualmente con cross-entropy como función de pérdida. Puede usarse con fine-tuning si ya tiene pesos preentrenados.

- Parámetros principales:

Parámetro	**Descripción**
Profundidad, número de capas, cantidad de bloques residuales.	Representa la capacidad y cuán profunda es la red.

Número de filtros por capa	Cuánta información se procesa en cada bloque
Kernel, uso de convoluciones “bottleneck”	Controla la eficiencia del modelo, cantidad de parámetros
Batch normalization, activaciones (ReLU)	Influye en la estabilidad de entrenamiento
Tasa de aprendizaje, optimizador	Control de sobreajuste, velocidad de convergencia

2. Ventajas y Desventajas

Ventajas	Desventajas
Alta capacidad de representación: puede capturar características complejas, útil para distinguir patrones morfológicos sutiles en espectrogramas.	Costoso computacionalmente: entrenamiento y predicción pueden requerir muchos recursos de GPU/CPU debido al número de capas y parámetros.
Buen soporte de transfer learning: modelos ResNet preentrenados están ampliamente disponibles, ayuda cuando los datos son limitados como en el caso de arritmias menos comunes.	Tamaño del modelo: peso grande, mayor memoria, más almacenamiento.
Robustez: la arquitectura residual facilita entrenamiento de redes profundas sin que se degrade el desempeño.	Latencia para inferencia: puede ser lento, especialmente si no se optimizan.
Interpretabilidad relativa: al usar Grad-CAM, que funciona sobre capas convolucionales profundas, ResNet tiene buena localización espacial de características discriminativas.	Riesgo de sobreajuste en clases minoritarias si los datos son escasos: modelos con alta capacidad pueden memorizar en lugar de generalizar, si no se cuidan los splits, regularización, augmentations.

3. Complejidad Computacional

- Complejidad temporal de entrenamiento o Tiempo de entrenamiento:

La complejidad computacional del entrenamiento de ResNet puede aproximarse como $O(N \cdot D \cdot M)$, donde N representa el número de ejemplos en el conjunto de datos, D la profundidad de la red, es decir, la cantidad de capas, y M el número de operaciones por capa, el cual depende del número de filtros, el tamaño de los kernels y las dimensiones de los mapas de activación intermedios. Específicamente, la arquitectura ResNet-50 contiene entre 23 y 25 millones de parámetros y requiere aproximadamente 4 GFLOPs por imagen de entrada de tamaño 224×224 píxeles.

- Complejidad temporal de predicción o Inferencia:

Durante la fase de inferencia, dado un tamaño de entrada fijo, la complejidad computacional es proporcional al número de operaciones requeridas por la red, es decir, aproximadamente $O(D \cdot M)$, donde D es la profundidad de la arquitectura y M representa el costo computacional por capa,

determinado por factores como el número de filtros, el tamaño de los kernels y las dimensiones de los mapas de características.

- Complejidad espacial:

Durante el entrenamiento, es necesario almacenar tanto las activaciones intermedias como los gradientes para llevar a cabo la retropropagación, además de los parámetros del modelo. Esto implica una complejidad espacial aproximada de O(D·S), donde D es la profundidad de la red y S el tamaño de los mapas de características, más un término adicional O(P), correspondiente al almacenamiento de los parámetros del modelo, siendo P el número total de pesos y sesgos.

- Escalabilidad con tamaño de datos:

El modelo escala de forma aproximadamente lineal con respecto al número de ejemplos N. Sin embargo, al incrementar la profundidad de la red, el costo computacional tiende a crecer de manera no lineal. Esto se debe no solo al aumento en la cantidad de operaciones, sino también a factores como mayores tiempos de comunicación entre capas, dificultades en la convergencia del entrenamiento y una mayor sensibilidad a problemas de optimización, como el desvanecimiento o explosión del gradiente.

4. **Casos de Uso Típicos**

- Dominios de aplicación:

Las arquitecturas convolucionales como ResNet se emplean ampliamente en tareas de visión por computador, incluyendo clasificación de imágenes, detección de objetos y segmentación semántica. También se utilizan en el ámbito médico para el análisis de imágenes diagnósticas (como resonancias magnéticas, radiografías o ecografías), así como en el procesamiento de señales unidimensionales que han sido transformadas en representaciones bidimensionales, como espectrogramas obtenidos a partir de señales de audio o electrocardiogramas.

- Tipos de datos apropiados:

Estas arquitecturas están diseñadas para procesar entradas en formato de imagen, típicamente en escala de grises o RGB, incluyendo datos derivados como espectrogramas, mapas de calor u otras representaciones bidimensionales. Su desempeño es óptimo cuando las imágenes poseen una resolución suficiente para capturar patrones espaciales relevantes, lo cual es crucial para tareas de clasificación precisa.

- Ejemplos reales de implementación:

Estas arquitecturas se han aplicado con éxito en la identificación automática de enfermedades a partir de imágenes médicas, incluyendo radiografías, tomografías y ecografías. En el ámbito del análisis de señales fisiológicas, han demostrado eficacia en la clasificación de patrones de ECG transformados en representaciones visuales, como espectrogramas. Asimismo, son utilizadas en aplicaciones de visión por computador en entornos domésticos (como reconocimiento facial o vigilancia) y en sistemas de percepción para vehículos autónomos, donde se requiere interpretar el entorno visual en tiempo real.

- Industrias que lo utilizan:

Estas arquitecturas son ampliamente adoptadas en sectores como; salud, especialmente en diagnóstico asistido por imagen; la industria automotriz, por ejemplo, en sistemas de conducción autónoma; la seguridad y vigilancia, mediante análisis de video en tiempo real, así como en aplicaciones de consumo como dispositivos inteligentes con capacidades de reconocimiento visual.

5. **Aplicabilidad al Proyecto**

- Relevancia específica a tu problema:

La capacidad de ResNet de capturar patrones complejos en espectrogramas lo hace adecuado para distinguir clases AAMI, que difieren en morfología del ECG. Esto hace que tenga una alta relevancia en el proyecto plantedo.

- Viabilidad de implementación:

La implementación de ResNet50 es técnicamente viable siempre que se disponga de recursos computacionales adecuados, particularmente una GPU con suficiente capacidad de memoria. Dado que las ventanas de ECG se transforman en espectrogramas de resolución moderada (por ejemplo, 224×224 píxeles), esta arquitectura puede procesarlas eficientemente. Además, el uso de técnicas de transferencia de aprendizaje reduce considerablemente el tiempo de entrenamiento y mejora el rendimiento en escenarios con datos médicos limitados.

- Recursos necesarios:

La implementación efectiva requiere una o más GPU con memoria suficiente para manejar tamaños de lote moderados o superiores durante el entrenamiento. También se necesita almacenamiento adecuado para guardar los modelos entrenados y sus checkpoints, así como tiempo computacional considerable para realizar múltiples iteraciones de entrenamiento y validación. Adicionalmente, es fundamental contar con una cantidad representativa de datos por clase, especialmente para evitar el sobreajuste en categorías minoritarias, lo cual es común en tareas clínicas con clases desequilibradas.

- Justificación de inclusión/exclusión:

Se recomienda incluir ResNet50 como línea base debido a su amplio reconocimiento en la comunidad científica, su rendimiento demostrado en tareas de clasificación de imágenes y su robustez comprobada en aplicaciones médicas. Además, su uso facilita comparaciones objetivas para evaluar posibles mejoras en eficiencia o precisión con otras arquitecturas. Sin embargo, una posible desventaja es la latencia durante la inferencia, especialmente si se planea implementar el modelo en dispositivos portátiles o de borde, lo que podría motivar la exploración de arquitecturas más ligeras y eficientes.

Técnica 2: ShuffleNet

1. **Descripción Teórica**

- Fundamentos matemáticos básicos:

ShuffleNet es una arquitectura CNN diseñada para optimizar la eficiencia computacional y reducir la latencia, especialmente en dispositivos con recursos limitados. Su principio clave es la combinación de convoluciones agrupadas y la operación de “channel shuffle”, que permite mezclar la información entre grupos de canales para mejorar la representatividad sin incrementar el costo computacional excesivamente.

- Principio de funcionamiento:

Matemáticamente, las convoluciones agrupadas dividen las entradas en grupos que son procesados independientemente, reduciendo la complejidad. La operación “channel shuffle” permuta los canales entre grupos para romper la restricción de comunicación limitada entre grupos.

- Tipo de aprendizaje:

ShuffleNet está diseñada para aprendizaje supervisado, con parámetros como el número de grupos en las convoluciones, número de bloques, y tamaño de los kernels.

- Parámetros principales

Parámetro	**Descripción**
Escala	Número de canales, complejidad y precisión vs eficiencia
Número de bloques y su estructura interna de grupos	Cantidad de operaciones grupales
Tamaño de kernel y uso de depthwise separable convs	Controla cómo se reparte la carga computacional

2. **Ventajas y Desventajas**

Ventajas	**Desventajas**
Altísima eficiencia: menor número de parámetros, menor costo de cómputo, bajísimo costo de inferencia, clave en entornos limitados o tiempo real.	Menor precisión comparada con arquitecturas más grandes en tareas difíciles o con datos muy complejos.
Latencia baja: especialmente en hardware móvil o dispositivos con memoria limitada.	No capta detalles sutiles en la morfología de espectrogramas si requieren alta resolución espacial y frecuencia.
Tamaño pequeño del modelo, menor uso de memoria RAM/GPU durante inferencia.	Requiere más cuidado en preprocesamiento, data augmentation, ajuste fino, para compensar menor capacidad.

Suficiente capacidad para muchas tareas de clasificación cuando los patrones no son extremadamente complejos o si se sacrifica algo de precisión por eficiencia.	Menos disponibilidad de versiones preentrenadas de alta calidad para algunas variantes específicas, o menor documentación de uso en dominios médicos específicos.
	Posibles compromisos en robustez frente al ruido, variaciones en señal, transformaciones adversas, etc.

3. **Complejidad Computacional**

- Complejidad temporal de entrenamiento o Tiempo de entrenamiento:

La complejidad temporal del entrenamiento en arquitecturas ligeras como ShuffleNet es proporcional a $O(N \cdot D \cdot M')$, donde N representa el número de ejemplos, D la profundidad de la red, y M' el número de operaciones por capa. En este caso, $M' \ll MM'$ en comparación con modelos más grandes como ResNet, debido al uso de operaciones más eficientes como convoluciones agrupadas y bloques menos costosos computacionalmente. Asimismo, la profundidad efectiva suele ser menor, lo que contribuye a una reducción significativa en el tiempo de entrenamiento.

- Complejidad temporal de predicción o Inferencia:

La complejidad temporal durante la inferencia es aproximadamente $O(D \cdot M')$, donde D es la profundidad de la red y M' representa el número de operaciones por capa, considerablemente reducido en comparación con arquitecturas más pesadas. Gracias a esta optimización, modelos como ShuffleNet presentan una latencia significativamente menor que ResNet al procesar imágenes de tamaño similar, lo que los hace especialmente adecuados para aplicaciones en tiempo real o en dispositivos con recursos limitados.

- Complejidad espacial o consumo de memoria:

Presenta un uso reducido de memoria tanto para almacenar activaciones intermedias como para los parámetros del modelo, en comparación con arquitecturas más complejas. Esta eficiencia permite su ejecución en entornos con limitaciones de hardware, como dispositivos móviles o sistemas embebidos.

- Escalabilidad con tamaño de datos:

La arquitectura escala de manera aproximadamente lineal con respecto al número de ejemplos. Sin embargo, en contextos donde el volumen de datos es limitado, puede no aprovechar completamente su capacidad de representación, lo que implica un menor margen de mejora frente a modelos más complejos.

4. **Casos de Uso Típicos**

- Dominios de aplicación:

Esta arquitectura ha demostrado ser especialmente útil en dispositivos móviles, entornos de Internet de las Cosas (IoT) y sistemas embebidos, donde los recursos computacionales son

limitados. Se ha aplicado en la monitorización remota de señales fisiológicas, como ECG, transformadas en representaciones visuales (espectrogramas), así como en dispositivos vestibles para el seguimiento continuo de parámetros de salud. También es adecuada para aplicaciones en tiempo real que requieren baja latencia y eficiencia energética.

- Tipos de datos apropiados:

ShuffleNet es particularmente eficaz al procesar imágenes de resolución moderada que contienen patrones discernibles sin requerir un nivel de detalle extremadamente fino. Funciona bien en escenarios con relaciones señal-ruido favorables, como espectrogramas de tamaño medio, donde las características relevantes pueden ser capturadas con bajo costo computacional. Esta propiedad lo hace adecuado para representaciones visuales derivadas de señales fisiológicas, como los espectrogramas generados a partir de ECG.

- Ejemplos reales:

ShuffleNet ha sido ampliamente utilizado en tareas de reconocimiento de imágenes en dispositivos móviles y sistemas embebidos, gracias a su alta eficiencia computacional. En el ámbito médico, ha demostrado ser útil para la clasificación automática de enfermedades a partir de imágenes diagnósticas, como radiografías o dermatoscopías, especialmente cuando se requiere operar en tiempo real con recursos limitados. Además, estudios recientes han aplicado variantes de ShuffleNet V2 en la identificación de enfermedades en cultivos agrícolas, logrando buenos niveles de precisión con arquitecturas de bajo número de parámetros.

- Industrias que lo utilizan:

Entre las industrias destacadas se encuentran la salud, especialmente en diagnóstico remoto y dispositivos portátiles; en la agricultura de precisión, detección de enfermedades en cultivos mediante visión artificial; en la manufactura ligera, para inspección visual y control de calidad, y en las telecomunicaciones, detección temprana de fallas en infraestructura mediante análisis de imágenes o espectros.

5. **Aplicabilidad al Proyecto**

- Relevancia específica:

La inclusión de esta técnica en el proyecto es altamente pertinente si el objetivo contempla la eficiencia computacional, la posibilidad de despliegue en dispositivos de borde o móviles, o si se trabaja con un conjunto amplio de espectrogramas donde la velocidad de inferencia es un factor crítico. ShuffleNet puede desempeñar el rol de modelo ligero complementario o comparativo frente a arquitecturas más pesadas como ResNet50 o MobileNetV2, permitiendo evaluar el compromiso entre precisión y eficiencia en contextos clínicos de clasificación de arritmias.

- Relevancia específica:

Tiene una aplicabilidad elevada en escenarios donde se prioriza la eficiencia computacional, especialmente si se considera el despliegue en dispositivos de borde o portátiles. Resulta adecuado cuando el conjunto de espectrogramas es voluminoso y se requiere alta velocidad de inferencia. En este contexto, ShuffleNet puede funcionar como un modelo ligero de referencia

frente a arquitecturas más complejas como MobileNetV2 y ResNet50, permitiendo contrastar rendimiento versus eficiencia para aplicaciones clínicas en tiempo real.

- Viabilidad de implementación:

ShuffleNet está disponible y puede ser necesario adaptar o entrenar una versión específica para procesar espectrogramas de tamaño 224×224, correspondientes a las transformaciones tiempo-frecuencia del ECG. Es fundamental garantizar que dichas transformaciones conserven un nivel de detalle suficiente para que un modelo ligero como ShuffleNet pueda discriminar eficazmente las clases AAMI, especialmente las categorías minoritarias que suelen ser más difíciles de clasificar.

- Recursos necesarios:

Requiere menor capacidad de GPU tanto para entrenamiento como para inferencia, así como un uso reducido de memoria. Aunque el tiempo por época puede ser menor, podría ser necesario entrenar durante más épocas o aplicar técnicas de regularización más estrictas para prevenir sobreajuste y asegurar un aprendizaje adecuado, especialmente dado el carácter ligero del modelo.

- Justificación de inclusión/exclusión:

ShuffleNet debe considerarse como un candidato ligero dentro del conjunto de modelos evaluados. Si ResNet50 y MobileNetV2 demuestran un desempeño satisfactorio, ShuffleNet puede ofrecer una alternativa con un compromiso aún mayor en eficiencia computacional, aunque potencialmente a costa de una ligera reducción en precisión, particularmente en clases minoritarias con patrones morfológicos sutiles como SVEB y VEB. En estos casos, el modelo podría presentar limitaciones para capturar detalles finos necesarios para una correcta clasificación.

Técnica 3: EfficientNet

1. **Descripción Teórica**

EfficientNet es una familia de redes neuronales convolucionales (CNNs) que introduce el concepto de **escalado compuesto**. A diferencia de arquitecturas anteriores que aumentaban solamente la profundidad, el ancho o la resolución de entrada, EfficientNet escala las tres dimensiones de manera balanceada mediante un coeficiente de escalado ϕ y constantes α, β y γ, obtenidas a partir de un modelo base diseñado mediante búsqueda automática de arquitectura (NAS).

- Fundamentos matemáticos básicos:

EfficientNet se fundamenta en el concepto de escalado compuesto. Matemáticamente, al escalar una red con un coeficiente ϕ, se ajustan de manera simultánea tres dimensiones:

- profundidad $d = \alpha^{\phi}$,
- ancho $w = \beta^{\phi}$ y

- resolución r = γ^ϕ,

bajo la restricción α · β² · γ² ≈ 2, lo que permite duplicar la carga computacional con cada incremento en ϕ.

- Principio de funcionamiento:

La arquitectura base EfficientNet-B0 fue obtenida mediante búsqueda neural automatizada (NAS). Sobre esa base, las versiones B1 a B7 se derivan mediante escalado compuesto. Además, la red hace uso de bloques MBConv (Mobile Inverted Bottleneck) con convoluciones separables en profundidad y convoluciones puntuales 1×1 para reducir parámetros y operaciones.

- Tipo de aprendizaje:

EfficientNet es un modelo supervisado diseñado principalmente para tareas de clasificación de imágenes, aunque también se ha extendido a otros dominios visuales.

- Parámetros principales.

Incluye factores de profundidad, ancho y resolución (según la variante B0–B7), tipo de bloque (MBConv o fused-MBConv en V2), función de activación Swish/SiLU, factores de expansión internos y parámetros de regularización como dropout o stochastic depth

2. Ventajas y Desventajas

Ventajas	**Desventajas**
Alta eficiencia en términos de precisión por parámetro y FLOPs: EfficientNet-B7 alcanza 84.3% en ImageNet con 66M de parámetros y 37B FLOPs, superando arquitecturas más grandes con menor costo [1].	Sensibilidad a la resolución de entrada: modelos grandes requieren imágenes de gran tamaño, aumentando costos de memoria [3].
Excelente capacidad de transferencia a datasets médicos y de visión general [4]. - Entrenamiento y predicción más rápidos en EfficientNetV2 gracias al uso de bloques fused-MBConv y aprendizaje progresivo [3].	Dependencia de configuraciones de entrenamiento (data augmentation, regularización, etc.) para reproducir resultados de los autores [1].
Escalabilidad flexible: desde B0 (ligero para móviles) hasta B7 (mayor capacidad), lo que permite ajustarse a distintos recursos de hardware	No posee explicabilidad nativa, por lo que es necesario aplicar técnicas adicionales de interpretación como Grad-CAM o Layer-CAM [5].

3. Complejidad Computacional.

El costo de entrenamiento en EfficientNet está determinado principalmente por las operaciones de convolución. En arquitecturas tradicionales, estas operaciones requieren un volumen elevado de cálculos, mientras que en EfficientNet, gracias al uso de convoluciones separables en profundidad, se reduce de forma considerable la cantidad de operaciones necesarias [6]. Esto significa que, a igualdad de capas, EfficientNet necesita menos recursos para entrenarse. Aun así, el tiempo total de entrenamiento sigue dependiendo del número de épocas y del tamaño del conjunto de datos utilizado.

Complejidad temporal de predicción.
El proceso de inferencia (predicción) es más liviano que el entrenamiento, ya que no incluye retropropagación de gradientes. El costo computacional depende de la cantidad de operaciones que realiza cada variante. Por ejemplo, EfficientNet-B0 requiere un volumen bajo (~0.39 mil millones de operaciones), mientras que EfficientNet-B7 exige un volumen mucho mayor (~37 mil millones de operaciones). Esto permite ajustar el modelo en función de las capacidades del hardware disponible y los requisitos de latencia.

Complejidad espacial.
El consumo de memoria está dominado por la cantidad de parámetros del modelo. Las variantes más pequeñas, como B0, tienen aproximadamente 5.3 millones de parámetros, mientras que B7 asciende a unos 66 millones. La elección de bloques MBConv ayuda a disminuir significativamente la memoria necesaria frente a redes convolucionales tradicionales [6], lo que hace más eficiente el almacenamiento y la ejecución.

Escalabilidad con el tamaño de datos.
EfficientNet ofrece una escalabilidad controlada y sistemática mediante el uso del coeficiente de escalado compuesto, que ajusta de forma conjunta la profundidad, el ancho y la resolución de entrada. EfficientNetV2, además, introduce un esquema de aprendizaje progresivo: el modelo comienza entrenando con imágenes de baja resolución y menor regularización, y gradualmente incrementa ambos factores conforme avanza el entrenamiento y aumenta la cantidad de datos [3]. Esto acelera la convergencia y optimiza el uso de recursos computacionales.

4. Casos de uso típico.

- Dominios de aplicación.
  Se ha utilizado ampliamente en visión por computador: clasificación de imágenes, detección de objetos y segmentación semántica.
- Tipos de datos apropiados.
  Imágenes 2D en general, incluyendo espectrogramas, imágenes médicas, fotografías y datos transformados en el dominio de la frecuencia.
- Ejemplos reales de implementación.
  - Clasificación de arritmias cardíacas a partir de ECG transformados en imágenes mediante scalogramas y EfficientNetV2.
  - Detección de apnea del sueño en señales ECG monoderivación convertidas a espectrogramas.

    - Uso en visión industrial y reconocimiento de productos en retail por su eficiencia en dispositivos de borde.
- Industrias que lo utilizan.
  Salud digital, automoción (visión en tiempo real), manufactura, consumo (reconocimiento en móviles), y aplicaciones biomédicas.

5. Aplicabilidad al proyecto

- Relevancia específica al problema:

El proyecto se centra en la clasificación de arritmias a partir de espectrogramas generados de ECG. EfficientNet y EfficientNetV2 han sido probados con éxito en estudios similares que convierten señales 1D en imágenes 2D y alcanzan resultados superiores a CNNs tradicionales.

- Viabilidad de implementación:

EfficientNet-B0 o B1 son adecuados para pruebas iniciales en GPUs estándar.

EfficientNetV2-S/M permiten entrenamiento más rápido y eficiente, especialmente útil si se cuenta con recursos limitados o se busca iterar rápidamente.

- Recursos necesarios:

Dataset de ECG (como MIT-BIH), preprocesamiento con STFT o CWT para obtener espectrogramas o scalogramas RGB, una GPU con 8–16 GB de VRAM para B0/B1, y hasta 24 GB para variantes mayores. Frameworks como PyTorch o TensorFlow facilitan el uso de modelos preentrenados y bibliotecas de XAI para interpretabilidad.

- Justificación de inclusión:

Se recomienda incluir EfficientNet en el portafolio de técnicas del proyecto por su balance entre precisión y eficiencia, su aplicabilidad comprobada en espectrogramas médicos y su soporte a técnicas de interpretabilidad. EfficientNetV2 es especialmente recomendable por la reducción en tiempo de entrenamiento y la escalabilidad.

Técnica 4: MobileNet

1. Descripción Teórica

- Fundamentos matemáticos básicos.

La familia MobileNet se basa en convoluciones separables en profundidad (depthwise separable): descomponen la convolución estándar en (a) una convolución por canal (depthwise) y (b) una convolución 1×1 (pointwise) para mezclar canales, reduciendo drásticamente el cómputo y los parámetros frente a una convención densa equivalente. MobileNetV2 agrega bloques “inverted residual” con “linear bottleneck” (expansión → depthwise → proyección lineal), y MobileNetV3 combina búsqueda automática (NAS/NetAdapt) con nuevas activaciones (h-swish, hard-sigmoid) y módulos SE (Squeeze-and-Excitation) optimizados a latencia

- Principio de funcionamiento:
- MobileNetV1 introduce dos hiperparámetros globales para ajustar el compromiso precisión–latencia: width multiplier (α) y resolution multiplier (ρ).
- MobileNetV2 usa bloques invertidos: expande canales, aplica depthwise, y proyecta a un cuello angosto lineal (sin no linealidad final) para preservar información; además presenta SSDLite para detección eficiente y una versión ligera de DeepLab para segmentación.
- MobileNetV3 emplea una búsqueda conjunta de arquitectura + ajustes (NAS + NetAdapt) y activaciones eficientes (h-swish) para maximizar precisión bajo restricciones reales de dispositivo (latencia medida en móvil).
- Tipo de Aprendizaje:

La familia MobileNet se usa típicamente en aprendizaje supervisado (clasificación, detección, segmentación) y como backbone en transferencia a múltiples tareas visuales.

- Parámetros principales:
- V1: α (ancho), ρ (resolución), tipo de bloque depthwise+pointwise, tasa de dropout/regularización.
- V2: factor de expansión (t), número de bloques invertidos, uso de ReLU6 en capas anchas y proyección lineal en cuellos.
- V3: configuración Large/Small, capas con SE, activación h-swish, arquitectura optimizada vía NAS y NetAdapt orientada a latencia en dispositivo.

2. Ventajas y desventajas

Ventajas	**Desventajas**
Muy eficiente en cómputo y memoria gracias a depthwise separable (gran reducción de multiplicaciones y parámetros frente a conv estándar)	Capacidad limitada frente a modelos grandes (p. ej., ResNet-50+) cuando no hay restricciones de cómputo; la precisión absoluta puede ser menor en escenarios sin límite de latencia.
Escalable al dispositivo: con α y ρ ajustas el tamaño del modelo según presupuesto (CPU/GPU/NPU/latencia).	Sensibilidad a la cuantización/activación: decisiones como ReLU6/linear bottleneck y cuantización requieren cuidado para no degradar precisión.
Bloques invertidos (V2) mejoran precisión/latencia frente a V1; además habilitan variantes ligeras para detección (SSDLite) y segmentación móvil (Mobile DeepLabv3).	Ingeniería de entrenamiento: para alcanzar los resultados de los autores se requiere seguir recetas de *augmentation*/regularización y, en V3, perfiles de búsqueda/latencia específicos.
• Optimización a latencia real en V3 mediante NAS + NetAdapt + activaciones eficientes (h-swish), logrando mejores trade-offs en móviles que V1/V2.	

3. **Complejidad Computacional**

- Complejidad temporal de entrenamiento

El entrenamiento está dominado por las capas convolucionales. Al sustituir la conv estándar por depthwise + pointwise, MobileNet reduce la cantidad de operaciones por capa de forma sustancial, por lo que entrena más rápido a igual tamaño de entrada que una CNN convencional de parámetros comparables. El tiempo total también depende de épocas y tamaño del conjunto de datos.

- Complejidad temporal de predicción

La inferencia es más liviana que el entrenamiento (no hay retropropagación). El costo clave es el número de operaciones (MACs/FLOPs) de la variante elegida: modelos más pequeños (p. ej., MobileNetV2/ V3 Small) ofrecen baja latencia en CPU/GPU móviles, mientras que variantes más anchas o con mayor resolución incrementan la latencia. SSDLite y Mobile DeepLabv3 mantienen costos contenidos en detección/segmentación móvil.

- Complejidad espacial

El uso de depthwise separable y cuellos angostos (linear bottleneck) reduce parámetros y memoria frente a conv estándar. V2/V3 mantienen esta eficiencia estructural; V3 añade módulos SE y activaciones h-swish con impacto moderado en memoria respecto a sus ganancias de precisión/latencia.

- Escalabilidad con el tamaño de datos

MobileNet escala controladamente con α (ancho) y ρ (resolución). En datos más complejos, puedes aumentar ancho/profundidad/resolución manteniendo latencia bajo control. V3 incorpora búsqueda con señal de latencia real para elegir configuraciones que se comporten bien al crecer el problema en dispositivos concretos.

4. **Casos de uso típicos.**

- Dominios de aplicación

Clasificación en móviles, detección (SSDLite), segmentación ligera (Mobile DeepLabv3), *embedded vision*, *edge AI*, y como backbone en tareas de alto rendimiento cuando la latencia es crítica.

- Tipos de datos apropiados

Imágenes 2D naturales, imágenes biomédicas y transformaciones 2D de señales (p. ej., espectrogramas/scalogramas) cuando se requiere baja latencia o despliegue en hardware restringido.

- Ejemplos reales de implementación

Arritmias ECG con MobileNetV2 (finetuning) para clasificación de latidos, reportado en literatura reciente ligera (y en revisiones 2024/2025).

Clasificación ECG con arquitectura híbrida MobileNetV2+BiLSTM para aprovechar contexto temporal.

CVD desde espectrogramas ECG con CNNs ligeras (incluyendo backbones tipo MobileNet) en entornos clínicos/portables.

- Industrias que lo utilizan

Salud digital (wearables/telemedicina), consumo (apps móviles), automoción (visión embarcada), manufactura/retail (inspección/escaneo) y ciudades inteligentes (sensado en borde). La eficiencia energética y la latencia hacen a MobileNet muy atractivo en “edge”.

5. **Aplicabilidad al proyecto.**

- Relevancia específica a su problema

El pipeline ECG → espectrogramas (STFT/CWT) → CNN + XAI requiere un backbone 2D eficiente. MobileNet (V2/V3) es ideal cuando se busca baja latencia y memoria con buena precisión, y ya existen trabajos con MobileNet sobre ECG (incluidos espectrogramas y arreglos híbridos) para arritmias.

- Viabilidad de implementación:

Altísima: puedes iniciar con MobileNetV2 preentrenado (input 224×224) y congelar capas iniciales, ajustando las últimas. Si necesitas aún menos latencia o mejor despliegue en móvil, evalúa MobileNetV3-Small/Large. Para detección de patrones específicos en ventanas, un cabezal temporal (p. ej., BiLSTM o TCN) sobre embeddings MobileNet puede ayudar.

- Recursos necesarios.
- Datos: MIT-BIH u otros repositorios; generar espectrogramas/scalogramas.
- Cómputo: una GPU 8–12 GB basta para V2/V3 en 224 px con lotes moderados.
- Software: PyTorch/TensorFlow con modelos pretrained; para XAI, Grad-CAM/Layer-CAM encima de las últimas capas conv. (XAI no es nativo en MobileNet).

- Justificación de inclusión/exclusion

Incluir: sobresaliente en eficiencia, con evidencia en ECG; ideal si contemplas despliegue en tiempo real o hardware restringido.
Excluir (potencialmente) solo si persigues precisión SOTA absoluta sin límites de cómputo, donde arquitecturas más grandes (p. ej., ConvNeXt/ViT grandes) podrían rendir más

Técnica 5: RepVGG

1. **Descripción Teórica**

- Fundamentos matemáticos básicos

RepVGG propone entrenar una red con topología multirrama (convoluciones 3×3, 1×1 y ramas de identidad con BatchNorm) y, al final del entrenamiento, reparametrizar estructuralmente todas esas ramas en un único bloque conv 3×3 + ReLU por capa. Es decir, durante el entrenamiento se beneficia de la capacidad/estabilidad de una arquitectura con atajos y ramas; en inferencia, colapsa esas ramas a una sola convolución 3×3 equivalente, quedando un "cuerpo VGG" puro, altamente eficiente en memoria y latencia. Este truco se llama structural re-parameterization.

- Principio de funcionamiento
    - (1) Se define una pila de bloques RepVGG con varias ramas (3×3, 1×1, identidad) y BN en entrenamiento. \
    - (2) Tras entrenar, se fusionan pesos y sesgos de todas las ramas en un kernel 3×3 único por bloque (la rama 1×1 se "embebe" al centro del 3×3; las BNs se pliegan en los kernels/sesgos).
    - (3) La arquitectura de inferencia resultante es tan simple como VGG (solo 3×3 + ReLU), pero con la precisión de una red moderna entrenada con atajos. En ImageNet, RepVGG supera 80% Top-1 con un "cuerpo" completamente conv 3×3 en inferencia.
- Tipo de aprendizaje (supervisado/no supervisado)

Se usa principalmente en aprendizaje supervisado (clasificación) y como backbone en detección/segmentación, donde la latencia de inferencia es crítica.

- Parámetros principales

Familia A/B (p. ej., RepVGG-A0, A1, B0, B1…), número de bloques por etapa, presencia de SE (Squeeze-and-Excitation) en variantes "plus", tamaño de entrada (224–320), y política de BN/activación (ReLU). La reparametrización es un paso determinista post-entrenamiento

2. Ventajas y desventajas

Ventajas	**Desventajas**
Inferencia muy rápida y simple: solo conv 3×3 + ReLU, sin atajos ni ramas; esto reduce accesos a memoria y favorece la paralelización y el uso eficiente de cachés, logrando mejor trade-off precisión/latencia que ResNet-50/101 en GPU y comparables favorables frente a EfficientNet/RegNet.	Cuantización: el diseño original puede amplificar error de cuantización si no se cuida; hay propuestas para hacerlo más "quantization-friendly", pero requieren ajustes.
Entrena "grande", predice "simple": la fase multi-rama facilita optimización/precisión; la fase reparametrizada es compacta para despliegue en edge.	Ecosistema más joven que ResNet/EfficientNet: menos "recipes" estándar y menos checkpoints en ciertos dominios especializados. (Aun así, el repo oficial ofrece modelos y guías).
Memoria en inferencia menor que arquitecturas con atajos (menos activaciones a retener entre capas).	Menos extensiones "listas" (p. ej., variantes de detección/segmentación específicas) que familias ultraconsagradas; hay que integrar como backbone de frameworks existentes.

3. **Complejidad Computacional**

- Complejidad temporal de entrenamient

Durante el entrenamiento, RepVGG usa ramas adicionales (3×3, 1×1, identidad) que aumentan algo el costo por capa respecto a una sola conv 3×3, pero esto se traduce en mejor capacidad de optimización. La clave es que, tras entrenar, dichas ramas se colapsan, por lo que ese sobrecosto no existe en inferencia. En la práctica, el tiempo de entrenamiento sigue dominado por las convoluciones 3×3 y el tamaño del dataset/épocas.

- Complejidad temporal de predicción

En inferencia, cada bloque es una única conv 3×3 + ReLU, lo que reduce FLOPs efectivos por acceso a memoria y mejora la latencia respecto a redes con atajos (p. ej., ResNet) de precisión similar. Resultados en ImageNet muestran >80% Top-1 con mejoras de 83% vs ResNet-50 y 101% vs ResNet-101 en 1080Ti, a igual o mejor precisión.

- Complejidad espacial

En inferencia, al eliminar ramas y atajos, disminuyen activaciones simultáneas y metadatos de grafo; el peso total depende de la variante (A0/A1/B0/B1…). Además, al plegar BatchNorm en los kernels, se evita almacenar BNs por separado. Esto lo hace memoria-eficiente frente a topologías residuales equivalentes en despliegue.

- Escalabilidad con el tamaño de datos

RepVGG escala con la profundidad/ancho de sus variantes y con la resolución de entrada (224–320). Puede entrenarse “ancho/profundo” con ramas y luego reparametrizar para servir en dispositivos con presupuesto de latencia estricto, manteniendo precisión competitiva.

4. **Casos de Uso Típicos**

- Dominios de aplicación:

RepVGG se ha utilizado en diversas tareas de visión por computadora, especialmente en escenarios que exigen inferencia rápida y eficiente. Es particularmente adecuado para implementaciones en hardware acelerado, como GPU, FPGA o ASIC, debido a su arquitectura “plana” optimizada para la etapa de despliegue. También ha mostrado buen desempeño en aplicaciones donde señales unidimensionales, como las biomédicas, se transforman en representaciones visuales, lo que permite aprovechar sus capacidades en contextos más allá de la visión natural.

- Tipos de datos apropiados:

RepVGG es adecuado para imágenes con resolución suficiente para preservar patrones relevantes, como espectrogramas u otras representaciones visuales derivadas de señales. Su diseño basado en convoluciones estándar permite capturar eficazmente estructuras espaciales siempre que la información discriminativa esté contenida en patrones accesibles mediante operaciones convolucionales convencionales.

- Ejemplos reales:

En el trabajo original de RepVGG, la arquitectura fue entrenada y evaluada en el conjunto de datos ImageNet, mostrando un desempeño competitivo en términos de precisión y velocidad de inferencia. Además, se han reportado aplicaciones exitosas en tareas de reconocimiento de objetos, segmentación semántica y clasificación de imágenes, especialmente en contextos donde se requiere alta eficiencia durante la fase de despliegue.

- Industrias que lo utilizan:

RepVGG se emplea en sectores similares a aquellos donde ResNet ha sido ampliamente adoptado, incluyendo el análisis de imágenes médicas, sistemas de vigilancia y monitoreo, así como en dispositivos embebidos donde la velocidad de inferencia es un requisito crítico. Su estructura optimizada para el despliegue lo hace especialmente atractivo en aplicaciones en tiempo real y en entornos con restricciones de hardware.

5. **Aplicabilidad al Proyecto**

- Relevancia específica:

La aplicabilidad de RepVGG al proyecto es alta, dado que ya se trabaja con espectrogramas como representación visual de señales ECG unidimensionales. RepVGG ofrece una arquitectura intermedia que combina una capacidad representativa sólida con una latencia de inferencia más baja que ResNet, lo que la convierte en una opción atractiva para escenarios donde se requiere eficiencia sin comprometer significativamente el rendimiento diagnóstico.

- Viabilidad de implementación:

Se dispone de recursos computacionales adecuados durante el entrenamiento, particularmente GPU con suficiente memoria para manejar la arquitectura multirrama, la implementación de RepVGG es completamente viable. Su principal ventaja se manifiesta en la etapa de inferencia, donde la reparametrización estructural permite un modelo simplificado y eficiente. Esto lo convierte en una opción especialmente atractiva para despliegue en dispositivos portátiles o entornos con restricciones de hardware.

- Recursos necesarios:

Se requiere al menos una GPU con mayor capacidad de memoria que la utilizada para modelos ligeros, debido a la arquitectura multirrama durante el entrenamiento. También es necesario contar con espacio de almacenamiento suficiente para guardar tanto el modelo entrenado como su versión reparametrizada. Además, el entorno de desarrollo debe ser compatible con la implementación de reparametrización estructural. Es posible que se requiera un mayor tiempo de desarrollo para ajustar hiperparámetros y garantizar que el modelo resultante para inferencia reproduzca fielmente el comportamiento del modelo entrenado.

- Justificación de inclusión/exclusión:

RepVGG debe ser considerado un candidato sólido dentro del conjunto de arquitecturas evaluadas. Su diseño permite mejorar notablemente la eficiencia en la etapa de inferencia, y su

capacidad representativa puede alcanzar niveles cercanos a los de ResNet. En caso de que los resultados experimentales muestren que ResNet ofrece una ventaja significativa en precisión, particularmente en la clasificación de clases minoritarias o en presencia de ruido, podría justificarse su elección final.

Referencias Bibliográficas:

[1] M. Tan and Q. Le, “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks,” in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2019, pp. 6105–6114. [En línea]. Disponible en: https://arxiv.org/abs/1905.11946

[2] A. Howard, M. Sandler, B. Chen, W. Wang, L.-C. Chen, M. Tan, et al., “Searching for MobileNetV3,” in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, 2019, pp. 1314–1324. [En línea]. Disponible en: https://arxiv.org/abs/1905.02244

[3] M. Tan and Q. Le, “EfficientNetV2: Smaller Models and Faster Training,” *arXiv preprint arXiv:2104.00298*, 2021. [En línea]. Disponible en: https://arxiv.org/abs/2104.00298

[4] J. He, Z. Zhang, and S. Zhang, “EfficientNet-Based Transfer Learning for Medical Image Classification,” *IEEE Access*, vol. 9, pp. 139422–139433, 2021. [En línea]. Disponible en: https://doi.org/10.1109/ACCESS.2021.3117747

[5] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, “Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization,” in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, 2017, pp. 618–626. [En línea]. Disponible en: https://doi.org/10.1109/ICCV.2017.74

[6] F. Chollet, “Xception: Deep Learning with Depthwise Separable Convolutions,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2017, pp. 1251–1258. [En línea]. Disponible en: https://doi.org/10.1109/CVPR.2017.195

[7] T. M. Nguyen, D. D. Pham, and H. V. Nguyen, “Scalogram-based Arrhythmia Classification using EfficientNetV2,” *Biomed. Signal Process. Control*, vol. 85, 2023, Art. no. 104949. [En línea]. Disponible en: https://doi.org/10.1016/j.bspc.2023.104949

[8] H. Li, Y. Wang, and J. Liu, “Detection of Sleep Apnea using Single-lead ECG and EfficientNet Architecture,” *Comput. Biol. Med.*, vol. 152, 2023, Art. no. 106445. [En línea]. Disponible en: https://doi.org/10.1016/j.compbiomed.2022.106445

[9] A. G. Howard et al., “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications,” arXiv:1704.04861, 2017. [En línea]. Disponible en: https://arxiv.org/abs/1704.04861

[10] A. G. Howard et al., “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications,” arXiv:1704.04861 (PDF), 2017. [En línea]. Disponible en: https://arxiv.org/pdf/1704.04861

[11] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “MobileNetV2: Inverted Residuals and Linear Bottlenecks,” in CVPR, 2018. [En línea]. Disponible en:

https://openaccess.thecvf.com/content_cvpr_2018/papers/Sandler_MobileNetV2_Inverted_Residuals_CVPR_2018_paper.pdf

[12] M. Sandler et al., “MobileNetV2: Inverted Residuals and Linear Bottlenecks,” arXiv:1801.04381, 2019 (v4). [En línea]. Disponible en: https://arxiv.org/pdf/1801.04381

[13] A. Howard et al., “Searching for MobileNetV3,” in ICCV, 2019. [En línea]. Disponible en: https://openaccess.thecvf.com/content_ICCV_2019/papers/Howard_Searching_for_MobileNetV3_ICCV_2019_paper.pdf

[14] K. Lee et al., “Lightweight Beat Score Map method for ECG arrhythmia classification,” Computer Methods and Programs in Biomedicine, 2024 (cita secundaria que refiere MobileNetV2 como baseline). [En línea]. Disponible en: https://www.sciencedirect.com/science/article/pii/S0208521624000858

[15] H. Narotamo et al., “Deep learning for ECG classification: A comparative study…,” Biomedical Signal Processing and Control, 2024 (revisión con múltiples representaciones y backbones). [En línea]. Disponible en: https://www.sciencedirect.com/science/article/pii/S174680942400199X

[16] S. Shin, Y. Yu, and H. Kim, “Lightweight Ensemble Network for Detecting Heart Arrhythmia,” Applied Sciences, 2022 (MobileNetV2 + BiLSTM). [En línea]. Disponible en: https://www.mdpi.com/2076-3417/12/7/3291

[17] S. D. Lilda et al., “Enhancing cardiovascular disease classification in ECG signals using spectrograms and CNNs,” Computers in Biology and Medicine, 2025 (uso de espectrogramas 2D con CNNs ligeras). [En línea]. Disponible en: https://www.sciencedirect.com/science/article/abs/pii/S0010482525000873

[18] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016 (ResNet original). [En línea]. Disponible en: https://doi.org/10.1109/CVPR.2016.90

[19] X. Zhang, X. Zhou, M. Lin, and J. Sun, “ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices,” *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2018 (Modelo ligero para visión en móviles). [En línea]. Disponible en: https://doi.org/10.1109/CVPR.2018.00716

[20] D. Ding et al., “RepVGG: Making VGG-style ConvNets Great Again,” *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2021 (Reparametrización estructural para inferencia eficiente). [En línea]. Disponible en: https://doi.org/10.1109/CVPR46437.2021.01353

[21] J. Zhang et al., “Explainable Deep Learning Model for ECG Classification Using Spectrogram and Grad-CAM,” *IEEE Journal of Biomedical and Health Informatics*, 2022 (Explicabilidad en espectrogramas de ECG). [En línea]. Disponible en: https://doi.org/10.1109/JBHI.2021.3119106

[22] Y. Wang, H. Wang, H. Chen, and Y. Hu, “Lightweight CNNs for Plant Disease Recognition Based on ShuffleNet and Attention Mechanisms,” *Computers and Electronics in Agriculture*, 2021 (Aplicación de ShuffleNet en agricultura con modelos compactos). [En línea]. Disponible en: https://doi.org/10.1016/j.compag.2021.106184

[23] S. Li et al., “A Lightweight Deep Learning Framework for Real-Time ECG Arrhythmia Classification Using Transfer Learning,” *IEEE Access*, 2021 (CNNs ligeras para clasificación de arritmias con ECG). [En línea]. Disponible en: https://doi.org/10.1109/ACCESS.2021.3059963

[24] W. Luo, Y. Li, R. Yu, and Y. Liang, “Real-Time Arrhythmia Detection from ECG Signals Using CNN-Based Lightweight Models,” *Sensors*, vol. 22, no. 4, p. 1337, 2022 (Detección de arritmias en tiempo real con modelos eficientes). [En línea]. Disponible en: https://doi.org/10.3390/s22041337