Optimización Automatizada de Hiperparámetros en Pipelines de Procesamiento de Imágenes en Sistemas Empotrados mediante Técnicas Genéticas

José L. Mira, Jesús Barba, Julián Caba, José A. de la Torre, Fernando Rincón, y Juan Carlos López[1],

Resumen—**El procesamiento eficiente en tiempo real de imágenes es de vital importancia en diversas aplicaciones, como el reconocimiento de objetos, la segmentación de regiones y la extracción de características. Sin embargo, al implementar estos algoritmos en dispositivos empotrados, surge una limitación inherente a los recursos de cómputo y la autonomía energética, ya que generalmente estos dispositivos funcionan con alimentación por baterías. Además, la configuración manual de los hiperparámetros para lograr un rendimiento óptimo y obtener resultados de alta calidad puede implicar un proceso complejo.**

En este trabajo, se propone un sistema de selección y evaluación automática para encontrar una combinación óptima de hiperparámetros en los pipelines de procesamiento de imágenes en dispositivos empotrados. Se consideran tanto la calidad de los resultados como los recursos utilizados. Se emplean métricas para evaluar la calidad de los resultados del procesamiento de imágenes y para medir los recursos utilizados en la implementación sobre dispositivos basados en tecnología de lógica reconfigurable (FPGA). Esta evaluación es clave en el proceso de maximizar la calidad de los resultados mientras se minimiza el consumo de recursos en el dispositivo.

Para validar la funcionalidad del sistema propuesto, se han llevado a cabo experimentos utilizando imágenes reales de trampas cromáticas de la mosca del olivo para realizar la segmentación y el conteo automático de estos insectos en dispositivos de control de plagas. Los resultados obtenidos demuestran que el enfoque propuesto permite encontrar combinaciones de hiperparámetros que logran un equilibrio óptimo entre la calidad de los resultados y la eficiencia en la utilización de recursos en entornos con recursos limitados.

Palabras clave—**Algoritmos Genéticos, Sistemas Empotrados, Procesamiento de Imágen, Hiperparameter Search.**

I. Introducción

En el campo de la visión por computador, la elección de los hiperparámetros de los algoritmos suele basarse en un enfoque de prueba y error en la mayoría de los casos. A medida que los algoritmos se vuelven más sofisticados y el número de hiperparámetros aumenta, encontrar una combinación óptima de valores para estos parámetros se vuelve cada vez más desafiante [1]. La elección adecuada de los hiperparámetros puede influir en gran medida en la eficiencia y la precisión de las tareas de visión por computador. Por ejemplo, en algoritmos de segmentación de imágenes, la selección óptima de hiperparámetros, como el umbral de binarización, el tamaño de ventana, el valor de suavizado y la sensibilidad de detección, puede determinar la calidad de la segmentación resultante [2] [3]. En otros casos como en los algoritmos de clasificación, el número de capas ocultas y la tasa de aprendizaje en redes neuronales, puede influir en la precisión y la capacidad de generalización del modelo [4].

Además, la implementación de algoritmos de visión por computador en sistemas empotrados basados en FPGA presenta una serie de desafíos considerables. Estos sistemas enfrentan dificultades debido a las restricciones de potencia computacional, recursos y consumo energético que suelen caracterizarlos. La elección de los hiperparámetros en estos algoritmos tiene un impacto significativo en la utilización de estos recursos limitados [5]. Por tanto, la optimización adecuada en esta seleccion se convierte en un factor crítico para lograr un equilibrio óptimo entre el rendimiento del algoritmo y la eficiencia del sistema empotrado, permitiendo así un despliegue efectivo. Estas consideraciones se vuelven especialmente relevantes en aplicaciones en tiempo real, donde la eficiencia y la precisión son aspectos clave a ser abordados en el diseño y la implementación de estos algoritmos.

La implementación de estos algoritmos en el ámbito de la agricultura inteligente plantea diversos desafíos [6]. Uno de los retos principales es la variabilidad de las condiciones ambientales agrícolas, caracterizada por cambios en la iluminación, densidad de vegetación y diversidad de objetos y cultivos presentes en las imágenes capturadas [7]. Esta variabilidad dificulta el desarrollo de algoritmos robustos que puedan adaptarse y generalizar de manera efectiva en entornos agrícolas diversos [8]. Otro desafío importante radica en la necesidad de sistemas que operen de manera eficiente y autónoma, ya que su implementación y toma de decisiones se llevará a cabo en ubicaciones con limitada conectividad a internet y fuentes de energía fotovoltaicas.

En el presente trabajo, se ha llevado a cabo la construcción de una herramienta destinada a automatizar la búsqueda de hiperparámetros en pipelines de procesamiento de imagen mediante la utilización de plantillas que definen las combinaciones de parámetros involucrados en el proceso y las técnicas genéticas aplicadas para optimizar la selección de dichos

[1]Escuela Superior de Informática. Dpto. de Tecnologías y Sistemas de Información. Universidad de Castilla-La Mancha, Ciudad Real. e-mail: joseluis.mira@uclm.es

parámetros. El objetivo principal ha sido evaluar y validar la funcionalidad de dicha herramienta en términos de la calidad de la segmentación de imágenes y la utilización de recursos en su implementación en sistemas empotrados. Para ello se han utilizando imágenes de trampas cromáticas como base para desarrollar y configurar un algoritmo que delimitará de manera precisa la presencia de insectos, utilizando técnicas de segmentación. Además se ha tomado como referencia la plataforma de cómputo Ultra96-V2 y los recursos de los que dispone.

II. Arquitectura de la Herramienta

Aunque se han realizado aproximaciones a la optimización de hiperparámetros en diversos algoritmos, incluyendo los de visión por computador, existe una falta de enfoque en el ámbito de los algoritmos de visión por computador específicamente diseñados para dispositivos empotrados. El aporte principal de este trabajo radica en la implementación de una herramienta que aborda la optimización en la búsqueda de hiperparámetros en algoritmos de visión por computador destinados a dispositivos empotrados que incorporan tecnología reconfigurable como un recurso más de procesamiento. Esta implementación se ha realizado bajo el lenguaje de programación Python, debido a la versatilidad y flexibilidad que proporciona para la construcción de pipelines de procesamiento de imagen. En la Figura 1 se puede observar la arquitectura de la herramienta a alto nivel.

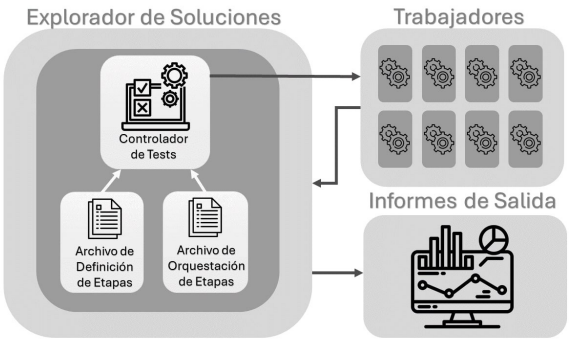

Fig. 1: Arquitectura de la herramienta

A. Archivos de configuración

La herramienta esta diseñada para componer un pipeline de procesamiento de imagen a través de tres archivos de configuración.

- **Archivo de definición de etapas**: En este archivo se encuentran definidas todas las etapas que se desean utilizar en el algoritmo. En cada una de estas etapas se definen los metadatos correspondientes a sus características. Se ha optado por el lenguaje *JSON* para la definición de este archivo ya que es considerado un formato referente en su uso en archivos de configuración, a la vez que su versatilidad en la manipulación del mismo.
  En el Listing 1 se puede observar la estructura del archivo de definición de las etapas que se van a utilizar, el ejemplo se corresponde con la definición de la función de erosión en imágenes.

Listing 1: Ejemplo de definición de la etapa de erosión

```
{
    "id":"60",
    "name":"Erode",
    "description":"Computes a local minimum over
         the area of given kernel.",
    "input parameters":[
        {
            "shape": ["n", "n"],
            "dtype": "int8"
        }
    ],
    "output parameters":[
        {
            "shape": ["n", "n"],
            "dtype": "int8"
        }
    ],
    "vitis_hls":{
        "function":"xf::erode<XF_BORDER_CONSTANT
             , XF_8UC1, @ROWS, @COLS, @K_SHAPE,
             @K_ROWS, @K_COLS, 1>(@input,
             @output, @kernel);",
        "includes":["#include \"xf_erosion.hpp
             \""],
        "references":
        {
            "@input":"input image stream from
                 previous step",
            "@output":"output image stream to
                 next step",
            "@ROWS":"Maximum height of input and
                  output image",
            "@COLS":"Maximum width of input and
                 output image",
            "@K_SHAPE":"Shape of the kernel .
                 The supported kernel shapes are
                  RECT,CROSS and ELLIPSE.",
            "@K_ROWS":"Height of the kernel.",
            "@K_COLS":"Width of the kernel.",
            "@kernel":"Erosion kernel of size
                 K_ROWS * K_COLS"
        },
        "resources":{
            "CLBs":96,
            "LUTs":392,
            "FFs":411,
            "DSPs":0,
            "BRAMs":3
        }
    }
},
```

- **Archivo de orquestación de etapas**: En este archivo se define la configuración de las etapas que se va a utilizar. Para ello, igual que en el archivo anterior se utiliza el lenguaje JSON. El archivo se puede entender como una lista de listas en las cuales están definidas las posibilidades combinatorias entre las diferentes etapas en el algoritmo. Este, se encarga de establecer el orden de las etapas que se van a utilizar gracias a su identificación a través del ID de etapa, además de los posibles valores con los que configurar cada una de estas etapas de manera individual. El archivo está diseñado de manera que permite la especificación directa de valores de configuración en las etapas, o bien establecer un rango de valores posibles para los hiperparámetros. En la Listing 2 se puede observar un ejemplo de archivo de orquestación de etapas.

Listing 2: Ejemplo de archivo de orquestación de etapas

```
    {
        "stages":[
            [
                {
                    "id":"00"
```

```
                },
                {
                    "id":"10",
                    "values":[1024, 512, 256]
                }
            ],
            [
                {
                    "id":"20"
                }
            ],
            [
                {
                    "id":"00"
                },
                {
                    "id":"30"
                }
            ],
            [
                {
                    "id":"51",
                    "min": 0,
                    "max": 50,
                    "step": 3
                },
                {
                    "id":"52"
                }
            ],
            [
                {
                    "id":"00"
                },
                {
                    "id":"60",
                    "min":[1, 3],
                    "max":[3, 7],
                    "step":[1, 2]
                },
                {
                    "id":"61",
                    "min":[1, 3],
                    "max":[3, 7],
                    "step":[1, 2]
                },
                {
                    "id":"62",
                    "min":[1, 3],
                    "max":[3, 7],
                    "step":[1, 2]
                },
                {
                    "id":"63",
                    "min":[1, 3],
                    "max":[3, 7],
                    "step":[1, 2]
                }
            ]
        ]
    }
```

- **Archivos de funciones**: Por último, están los archivos en los que se define la funcionalidad, en código ejecutable por la herramienta, de cada una de las etapas definidas en el archivo de definición. Este código ejecutable está diseñado para funcionar a modo de bloque el cual se configura con los hiperparametros establecidos. Este bloque a su vez es el encargado de ejecutar sobre la imagen de entrada la funcion que se ha definido y producir una imagen de salida como resultado.

B. Controlador de Tests

Esta parte de la arquitectura de la herramienta es la encargada de interpretar los ficheros de configuración del test. En primer lugar, se generan y recopilan todos los posibles valores de configuración de los hiperparámetros en cada etapa. A continuación, cada uno de estos valores se fusiona junto a los metadatos en el fichero de definición de las etapas. Una vez configuradas todas las etapas posibles, la herramienta se encarga de generar la combinación de etapas para dar lugar a cada uno de los tests posibles a realizar. Cabe destacar que el número de combinaciones posibles en un pipeline de procesamiento de imágenes de este estilo crece de manera exponencial en relación al número de etapas que se pretendan utilizar y la cantidad de valores de configuración posibles de esas etapas.

C. Hilos Productor y Consumidor

Otra de las características de la arquitectura es la capacidad de poder realizar la ejecución y evaluación de varios tests a la vez. Para ello, se ha implementado un modelo productor-consumidor basado en colas a través del cual un proceso se encarga de seleccionar un test del espacio de búsqueda posible para volcarlo al *buffer* de trabajo. Estos tests son consumidos por el número de hilos desplegados a petición del usuario, donde se ejecutan las etapas configuradas además del cálculo de las métricas relacionadas con los resultados obtenidos tras el proceso.

D. Explorador de soluciones

Se puede considerar esta parte de la arquitectura como la que implementa la heurística o el conjunto de reglas que guían el proceso de búsqueda en la selección de los hiperparámetros. En el caso de estudio de este trabajo, la heurística seleccionada ha sido la utilización de *algoritmos genéticos*. Este tipo de algoritmos presenta ventajas en términos de tiempo de computación para converger en una solución respecto a otro tipo de estrategias de búsqueda por fuerza bruta, como por ejemplo GridSearch o RandomSearch. Dado que la configuración de un test se basa en etapas, se puede entender la configuración de cada una de estas etapas como los genes que van a ser utilizados. Al igual que en otras implementaciones de este tipo de algoritmos, en primer lugar se establece una población aleatoria que tiene que evaluarse, para más adelante, en funcion de la calidad obtenida por cada uno de los miembros de la población (entiéndase los tests) se establecen unas probabilidades las cuales están directamente relacionadas con las métricas obtenidas. En base a estas probabilidades, se determina de manera aleatoria los cruces a partir de los cuales se va a realizar la configuración de parámetros a utilizar por la siguiente generación. Teniendo en cuenta que alguno de los parámetros generados cabe la posibilidad de que resulte alterado debido a un factor de mutación que se establece en el proceso de búsqueda para incrementar la efectividad de la heurística. Para finalizar, los resultados obtenidos son enviados a otra parte de la arquitectura donde son analizados y almacenados. En la Figura 2 se puede ver una ilustración sobre el proceso de selección de parámetros mediante estas técnicas genéticas.

E. Recolección de estadísticas

En este módulo se realiza la recolección de estadísticas tras la ejecución de cada uno de los tests.

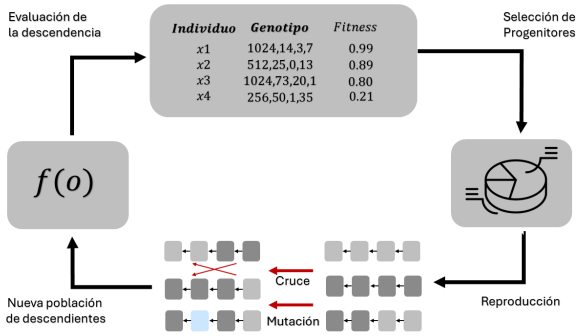

Fig. 2: Técnica genética en la exploración de hiperparámetros

Aquí quedan registrados los valores con los que ha sido ejecutado, además de las métricas que se han obtenido en los mismos. A partir de dichos registros, la herramienta es capaz de replicar cualquier test realizado. Se cuenta también con la generación de gráficos a modo de realimentación visual para el usuario, dichos gráficos permiten la observación sobre las configuraciones típicas mas repetidas entre los tests que han obtenido mejores métricas. Se puede entender esta parte como una ayuda a la toma de decisiones para el usuario a la hora de realizar la implementación del pipeline de procesamiento de imágenes sobre el dispositivo final.

III. Resultados

En esta sección se presentan los resultados obtenidos tras la evaluación de la herramienta utilizando como caso de estudio la configuración de la etapa de segmentación de imagen para una aplicación de detección de insectos en trampas cromáticas. El objetivo era determinar una configuración óptima de hiperparámetros en un pipeline de procesamiento de imágenes dedicado a estas tareas de segmentación. Sumado a eso, la utilización eficiente de recursos por la composición de etapas para una implementación sobre FPGA ha sido un foco de estudio. La incorporación de este punto de vista en las métricas que dirigen el proceso de búsqueda se ha visto que tiene un impacto en la calidad de la segmentación de las imágenes.

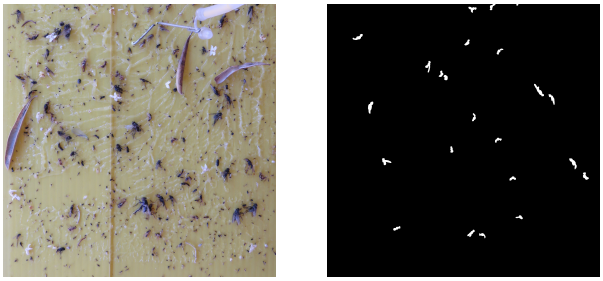

Fig. 3: Tipo de imagen y mascara del dataset

Con ese fin se ha utilizado un dataset compuesto por 13 imágenes provenientes de dichas trampas cromáticas, las cuales se hallan tomadas en diferentes condiciones ambientales de iluminación, en la Figura 3 se puede observar un ejemplo visual del tipo de imágenes que forman el dataset. Cada una de estas imágenes, viene acompañada de manera respectiva por una máscara de segmentación de los insectos realizada de forma manual para poder establecer una comparación con los resultados obtenidos. Para realizar la configuración de los hiperparámetros se ha utilizado una configuración similar a la que encontramos en el archivo visto de la Listing 2. La disposición de las etapas en la prueba, en orden de ejecución, ha quedado de la siguiente manera.

1. **No redimensionar las imágenes** o **Redimensionarlas**.
2. Convertir las imágenes a **Escala de Grises**.
3. **No Equalizar el Histograma** o **Equalizarlo**.
4. Hacer una binarización de las imágenes mediante la **Binarización Invertida** o mediante el metodo **Otsu**.
5. **No modificar la imagen** o **Dilatarla** o **Erosionarla** o **Apertura** o **realizar una Clausura**.
6. **No modificar la imagen** o **Dilatarla** o **Erosionarla** o **Apertura** o **realizar una Clausura**.
7. **No modificar la imagen** o **Dilatarla** o **Erosionarla** o **Apertura** o **realizar una Clausura**.
8. **No modificar la imagen** o **Dilatarla** o **Erosionarla** o **Apertura** o **realizar una Clausura**.

Esto sumado a los propios valores de configuración de cada etapa hace que el número de posibles combinaciones diferentes ascienda a 269879184. Para la prueba, se ha configurado la exploración de soluciones del algoritmo genético con los siguientes parámetros: 35 generaciones, población de 20 individuos y factor de mutación de 0,1.

Para la evaluación de la calidad de la segmentación se ha utilizado la métrica F1 que es la media armónica entre las métricas *precission* y *recall*. Estas métricas miden cómo de bien "clasificados" (esto es, discernir entre frente y fondo) están los píxeles resultantes de la ejecución del test.

Por otro lado, para medir la utilización de recursos se ha utilizado la placa de desarrollo AVNET Ultra96-V2, la cual incorpora el chip FPGA Xilinx Zynq UltraScale+ MPSoC ZU3EG A484. Los parámetros de este chip se han utilizado como referencia para estimar el porcentaje total de utilización de recursos que está ocupando cada test. En base a esta utilización se puede establecer una escala normalizada en la cual los valores cercanos a 1 indiquen una ocupación mínima de los recursos de la placa, mientras que los valores cercanos a 0 o incluso negativos indiquen una ocupación total o por exceso de los recursos disponibles en el chip. Para esta evaluación se han tenido en cuenta la estimación de recursos FFs, LUTs y BRAMs.

Se ha establecido una ponderación al 50 % entre las dos métricas, en la cual cada tanto la evaluación de la calidad de la segmentación como la estimación de recursos utilizados contribuyen de manera uniforme

a la función de evaluación utilizada en el proceso de exploración de soluciones.

Tras la ejecución de las pruebas se ha encontrado la siguiente distribución de configuraciones a partir de la selección de los 25 mejores tests ejecutados por la herramienta en base a sus métricas. En la Figura 4 se puede observar la distribución de las etapas que han obtenido mejores métricas en la realización de los experimentos. Tras el análisis de los resultados se puede observar que en las etapas que están comprendidas entre la 1 a la 4 hay una clara predominancia sobre la mejor configuración de hiperparámetros en estas. Es en el caso de las cuatro últimas etapas, las cuales están marcadas por la aplicación de operadores morfológicos sobre las imágenes, se puede observar algo mas de variabilidad en la selección óptima de estos hiperparámetros. Aun así es única y exclusivamente el caso de la última etapa donde se encuentra la mayor parte de esta variabilidad, lo cual indica la menor relevancia de esta etapa en la obtención de los resultados finales. Se demuestra así la versatilidad y utilidad de la herramienta al mostrar información relevante sobre el proceso que ayuda al diseñador del algoritmo y/o arquitectura a tomar decisiones poniendo el foco en aquellos aspectos del pipeline que más influyen en la calidad de los resultados.

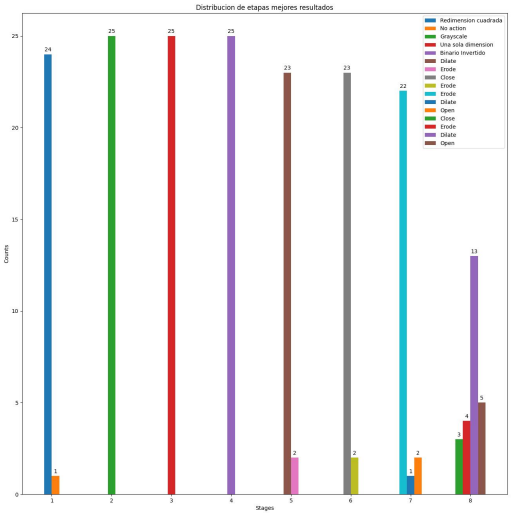

Fig. 4: Distribución típica de operaciones por etapas

También se puede observar la evolución de las métricas de la imagen a través de las generaciones en la utilización de los algoritmos genéticos en la Figura 5. En el caso de la métrica F1 se puede observar una tendencia de mejora a través de las generaciones. Por otro lado la métrica MAE (*Mean Absolute Error*) medida en la diferencia en el número de regiones independientes entre la mascara de referencia y la máscara obtenida en los tests muestra una ligera tendencia hacia el decrecimiento. Por último la métrica de SSIM, que indica la similitud morfológica entre los objetos segmentados por el algoritmo y la máscara de referencia, parece mantenerse estable.

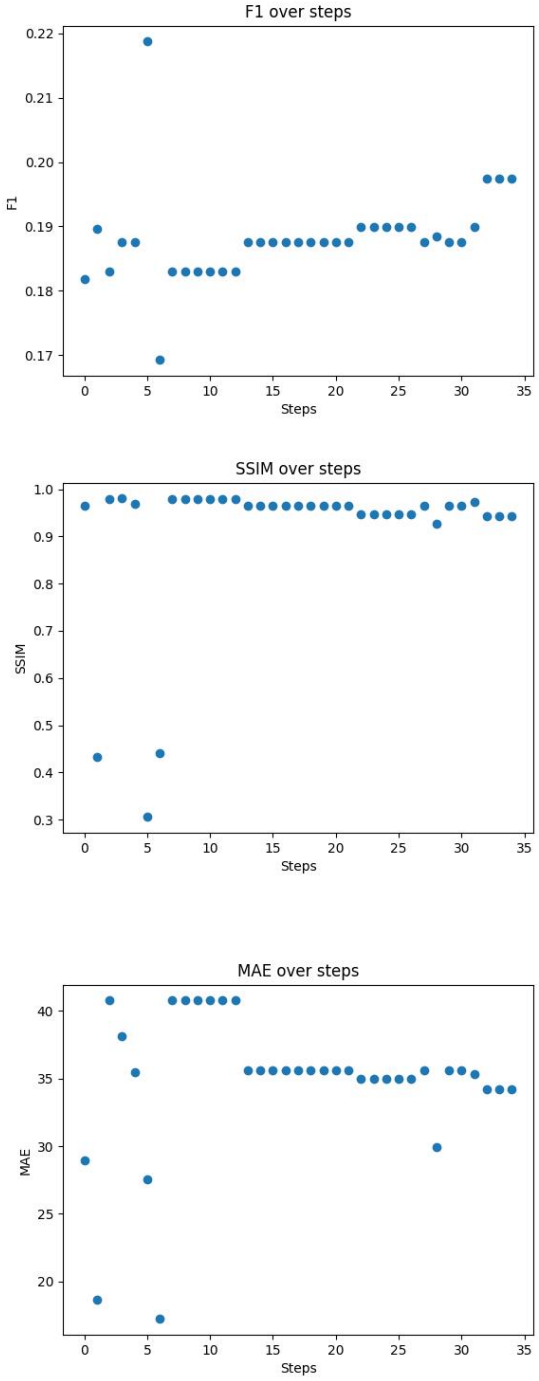

Fig. 5: Métricas de calidad de la imagen en relación a las generaciones

En cuanto a los resultados obtenidos en la evaluación del consumo de recursos, se puede observar en la Figura 6 una tendencia hacia la minimización de los recursos utilizados en cada una de las generaciones evaluadas por la herramienta.

Visualmente también se puede observar en la Figura 7 un ejemplo de segmentación obtenido por la configuración automática de los hiperparámetros en el pipeline de procesamiento de imágenes.

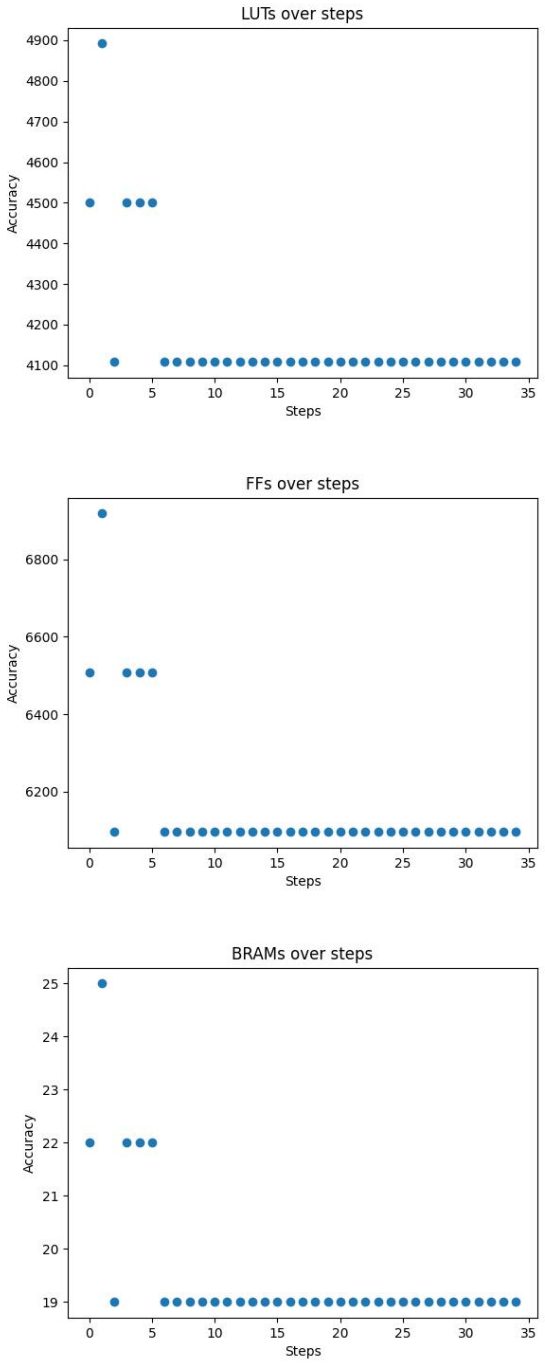

Fig. 6: Métricas de utilización de recursos en relación a las generaciones

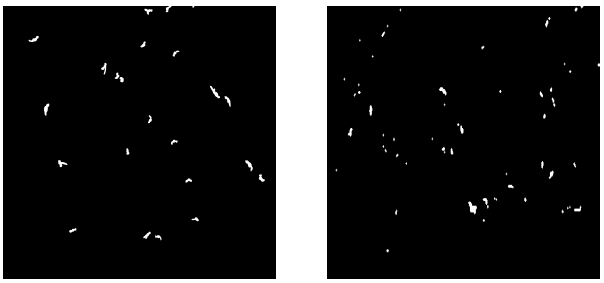

Fig. 7: Comparativa máscara real vs segmentación automática

IV. Conclusiones

En este trabajo se ha realizado el diseño e implementación de una herramienta para la selección de hiperparámetros en pipelines de procesamiento de imágenes. El diseño de la herramienta se ha realizado de manera que puedan definir de manera formal las etapas que van a ser utilizadas en un pipeline de procesamiento de imagen y el rango de parámetros que se desee probar en las mismas. La selección se ha llevado a cabo utilizando técnicas genéticas para guiar el proceso de búsqueda en el espacio de soluciones. Uno de los aspectos mas relevantes del estudio ha sido tratar de enfocar esta búsqueda en base a la asignación eficiente de recursos a la hora de seleccionar las etapas que componen el pipeline. Esto es debido a que el despliegue de estos pipelines tiene como objetivo plataformas de cómputo empotradas que estarán desplegadas en el borde de la infraestructura de adquisición, procesamiento y transmisión de la información (p.ej. en una plataforma IoT). En última instancia, el tipo de plataformas consideradas para la aplicación final consistirá en un dispositivo de procesamiento basado en un SoC-FPGA y estará alimentado por batería; de ahí la importancia de los recursos empleados.

Los resultados obtenidos muestran la capacidad de este tipo de técnicas de optimización de hiperparámetros para encontrar soluciones óptimas en terminos de calidad en la segmentación de las imagenes y eficiencia en la utilización de recursos en los dispositivos empotrados.

Agradecimientos

Esta investigación ha sido financiada por el Ministerio de Economía y Competitividad (MINECO) del Gobierno de España bajo los proyectos TALENT (PID2020-116417RB-C4, subproyecto 1) y MIRATAR (TED2021-132149B-C41), así como por la Universidad de Castilla-La Mancha por medio de la ayuda grupos de investigación consolidados (2022-GRIN-34489).

Referencias

[1] Moshe Sipper, "High per parameter: A large-scale study of hyperparameter tuning for machine learning algorithms," *Algorithms*, vol. 15, no. 9, 2022.

[2] Aritra Chowdhury, Malik Magdon-Ismail, and Bülent Yener, "Quantifying error contributions of computational steps, algorithms and hyperparameter choices in image classification pipelines," *CoRR*, vol. abs/1903.02521, 2019.

[3] Gabriel Fillipe Centini Campos, Saulo Martiello Mastelini, Gabriel Jonas Aguiar, Rafael Gomes Mantovani, Leonimer Flávio de Melo, and Sylvio Barbon, "Machine learning hyperparameter selection for contrast limited adaptive histogram equalization," *EURASIP Journal on Image and Video Processing*, vol. 2019, no. 1, pp. 59, 05 2019.

[4] K. Shankar, Yizhuo Zhang, Yiwei Liu, Ling Wu, and Chi-Hua Chen, "Hyperparameter tuning deep learning for diabetic retinopathy fundus image classification," *IEEE Access*, vol. 8, pp. 118164–118173, 2020.

[5] Sergei Alyamkin, Matthew Ardi, Alexander C. Berg, Achille Brighton, Bo Chen, Yiran Chen, Hsin-Pai Cheng, Zichen Fan, Chen Feng, Bo Fu, Kent Gauen, Abhinav Goel, Alexander Goncharenko, Xuyang Guo, Soonhoi Ha, Andrew Howard, Xiao Hu, Yuanjun Huang, Donghyun Kang, Jaeyoun Kim, Jong Gook Ko, Alexander Kondratyev, Junhyeok Lee, Seungjae Lee, Suwoong Lee, Zichao Li, Zhiyu Liang, Juzheng Liu, Xin Liu, Yang Lu, Yung-Hsiang

Lu, Deeptanshu Malik, Hong Hanh Nguyen, Eunbyung Park, Denis Repin, Liang Shen, Tao Sheng, Fei Sun, David Svitov, George K. Thiruvathukal, Baiwu Zhang, Jingchi Zhang, Xiaopeng Zhang, and Shaojie Zhuo, "Low-power computer vision: Status, challenges, and opportunities," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 2, pp. 411–421, 2019.

[6] Diego Inácio Patrício and Rafael Rieder, "Computer vision and artificial intelligence in precision agriculture for grain crops: A systematic review," *Computers and Electronics in Agriculture*, vol. 153, pp. 69–81, 2018.

[7] Yuzhen Lu and Sierra Young, "A survey of public datasets for computer vision tasks in precision agriculture," *Computers and Electronics in Agriculture*, vol. 178, pp. 105760, 2020.

[8] José L. Mira, Jesús Barba, Francisco P. Romero, M. Soledad Escolar, Julián Caba, and Juan C. López, "Benchmarking of computer vision methods for energy-efficient high-accuracy olive fly detection on edge devices," *Multimedia Tools and Applications*, 2024.