Comparación de Vitis-AI y FINN para implementar redes neuronales convolucionales en FPGA

1er Nicolás Urbano Pintos
DRL - CITEDEF
GRUPO TAMA - UTN FRH
Prov. Bs. As., Argentina
urbano.nicolas@gmail.com

2do Héctor Alberto Lacomi
DRL - CITEDEF
GRUPO ASE - UTN FRH
Prov. Bs. As., Argentina
hlacomi@citedef.gob.ar

3er Mario Blas Lavorato
GRUPO TAMA
UTN FRH
Haedo, Argentina
mlavorato@frh.utn.edu.ar

***Resumen*—Las redes neuronales convolucionales (CNN - Convolutional Neural Networks) son esenciales para la clasificación y detección de imágenes, y su implementación en sistemas embebidos es cada vez más deseable, ya que estos dispositivos ofrecen un tamaño compacto y bajo consumo de energía. Los FPGA han surgido como una opción atractiva debido a su baja latencia y alta eficiencia energética.**

Vitis AI y FINN son dos entornos de desarrollo que automatizan la implementación de CNN en FPGA. Vitis AI utiliza una unidad de procesamiento de aprendizaje profundo (DPU- Deeplearning Processing Unit) y aceleradores de memoria, mientras que FINN se basa en una arquitectura tipo streaming y ajusta la paralelización. Ambos entornos utilizan la cuantización de parámetros para reducir la memoria necesaria.

Si bien ya existen trabajos que comparan estos entornos, en este trabajo se amplía la comparación implementando tres modelos de CNN y un modelo de redes neuronales totalmente conectado para la clasificación de imágenes en la placa de desarrollo FPGA Kria KV260 de Xilinx utilizando ambos entornos. Se detalla el proceso completo, desde el entrenamiento hasta la evaluación en FPGA, incluyendo la cuantización y la implementación en hardware.

Los resultados muestran que FINN ofrece menor latencia, mayor rendimiento y mejor eficiencia energética que Vitis AI. Sin embargo, Vitis AI presenta ventajas en términos de facilidad de entrenamiento del modelo y ofrece un proceso de implementación en FPGA más simple.

***Index Terms*—FPGA, CNN, FINN, Vitis-AI, Cuantización**

I. Introducción

Las CNN son fundamentales en aplicaciones como la clasificación y detección de imágenes [1]. Estas redes neuronales utilizan una gran cantidad de capas y filtros para obtener precisiones de clasificación alta. Esto trae aparejado una utilización de memoria considerable y gran cantidad de operaciones de punto flotante. En aplicaciones donde se requiere inferir estos modelos en dispositivos portátiles, los sistemas embebidos surgen como una opción. Sin embargo, la cantidad de memoria y capacidad de cálculo necesaria para inferir en tiempo real, dificultan esta tarea. Una de las técnicas más utilizada para reducir la memoria, es la cuantización de los parámetros del modelo. Entre los enfoques actuales se destacan el uso de enteros de 8 bits [2] y las cuantizaciones binarias [3].

Los dispositivos FPGA, son una alternativa fundamental para la inferencia de estos modelos de aprendizaje profundo, debido a su baja latencia y gran eficiencia energética.

Vitis AI [4] y FINN [5] son dos entornos de desarrollo concebidos con el objetivo de automatizar la implementación de modelos de aprendizaje profundo en FPGA, ambos son de la firma Xilinx. Vitis AI, implementa un IP optimizado para la inferencia de redes neuronales de convolución, basado en un DPU y aceleradores de memoria. El DPU está formado por un planificador de trabajo, un módulo matricial de cómputo híbrido, y un módulo de memoria tipo pool [6]. En este DPU se ejecuta un microcodigo a través de un archivo con extensión xmodel generado por el compilador del Vitis AI. En cambio, FINN utiliza una arquitectura tipo streaming, e implementa las CNN a partir de 3 componentes, una Unidad de umbral vectorial de matriz, una unidad de ventana deslizante y una unidad de agrupamiento [7]. A través de factores de plegado o folding se determina la cantidad de elementos de procesamiento y carriles que realizan en paralelo las operaciones.

Machura et al. [8] implementa dos modelos de detección basados en CNN, LittleNet y YoloFINN, utilizando FINN y Vitis AI. Evalúan la precisión de detección de objetos, el rendimiento y el uso de energía en una placa de desarrollo Avnet Ultra96-V2.

Hamanaka et. al [9] realizan una comparación de FINN y Vitis AI con un modelo ResNet-8 implementado en una FPGA SOM K26 de Xilinx. Los resultados muestran FINN supera al acelerador basado en Vitis AI, en latencia, rendimiento y eficiencia energética.

El objetivo de este trabajo es ampliar la comparación de FINN y VITIS AI existente en la bibliografía. Para ello, se realiza un estudio detallado de estos entornos mediante la implementación de cuatro modelos de clasificación de imágenes. Tres de los modelos consisten en capas convolucionales y capas totalmente conectadas, variando en la cantidad de capas y tamaños de filtros. El cuarto modelo solo contiene capas totalmente conectadas.

Este estudio describe los pasos necesarios para implementar cada modelo en ambos entornos, abarcando desde la arquitectura, el entrenamiento, la cuantización y la implementación en una placa de desarrollo de FPGA Kria KV260. Además, se lleva a cabo un análisis haciendo hincapié en las métricas obtenidas en función de la complejidad del modelo. También se detalla el flujo de desarrollo de cada entorno, destacando las fortalezas y debilidades específicas de FINN y VITIS AI.

II. Metodología

Para este trabajo se implementaron 3 modelos convolucionales, CNV, VGG11 y mCNN, y un modelo con capas totalmente conectadas LFC. Los modelos CNV y LFC fueron desarrollados por Umuroglu et. al [5], VGG11 una versión reducida de VGG16 [10] y mCNN (mini CNN) es un modelo con 9 capas implementado por los autores. Si bien las redes totalmente conectadas no tienen filtros de convolución, son una parte fundamental en la etapa de clasificación de imágenes con CNN, por este motivo se incluyó a la comparación el modelo LFC.

A continuación se especifica los conjuntos de datos utilizados, la arquitectura de cada modelo, los hiperparámetros de entrenamiento, los procedimientos en FINN y VITIS-AI, los detalles para la preparación de la placa Kria KV-260, y las métricas empleadas en la comparación.

II-A. Conjuntos de datos

Se utilizaron los conjuntos de datos CIFAR10, MNIST y SVHN. CIFAR10 contiene 60 mil imágenes de 32x32 píxeles, color de objetos y animales en 10 clases. MNIST contiene 60 mil números manuscritos del cero al nueve en escala de grises con un tamaño de 28x28 píxeles. Y SVHN, contiene más de 120 mil imágenes de números de direcciones de casas recortadas, del cero al nueve en color con un tamaño de 32x32 píxeles.

II-B. Arquitecturas

En las figuras 1a, 1b, 1c y 1d se observan los tipos y cantidad de capas, y el tamaño de filtros de cada una. Siendo "Conv" capas de convolución y activación en color azul, "MaxPool" capas de pooling en color rojo y "TC" capas totalmente conectadas en color verde.

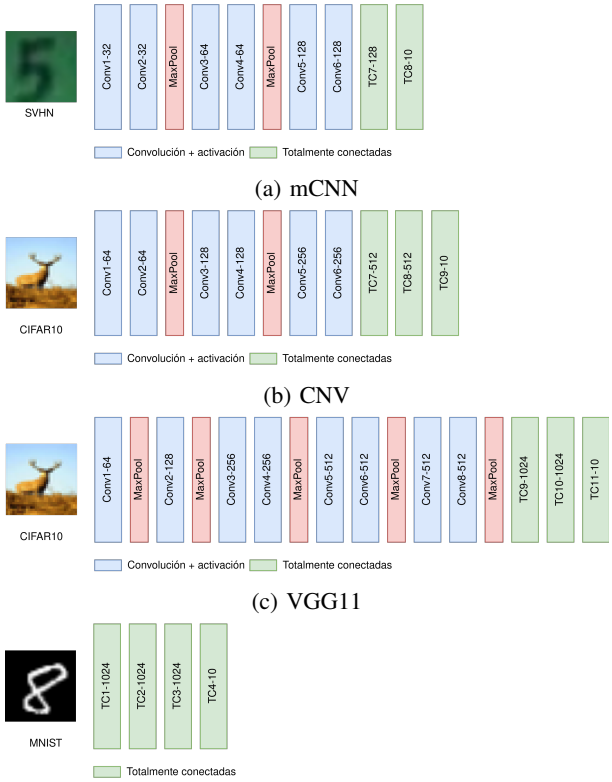
II-C. Entrenamiento

Los modelos desarrollados para FINN fueron implementados a partir de las capas de Brevitas realizando una cuantización binaria consciente del entrenamiento, con los hiperparámetros que se detallan en la tabla I. Los modelos

	Brevitas/FINN			
	CNV	mCNN	VGG11	LFC
Filtros pre-entrenados	No	No	No	No
Tipo de capas	QuantLayers	QuantLayers	QuantLayers	QuantLayers
Épocas	1000	1000	1000	1000
Tasa de Aprendizaje	0.02	0.02	0.02	0.02
Tamaño de lote	100	100	100	100
Pérdidas	CrossEntropy	CrossEntropy	CrossEntropy	CrossEntropy
Activaciones	Identidad	Identidad	Identidad	Identidad
Optimizador	ADAM	ADAM	ADAM	ADAM
Bits	1	1	1	1

Tabla I: Hiperparámetros utilizados en Brevitas/FINN

desarrollados para Vitis AI fueron implementados en PyTorch, y entrenados con los hiperparámetros que se detallan en la tabla II. Se utilizaron pesos pre-entrados en VGG11 en los 3 modelos basados en CNN, por lo tanto, no fue necesario entrenarlo más de 50 épocas.



(a) mCNN

(b) CNV

(c) VGG11

(d) LFC

Figura 1: Arquitecturas

	pytorch/vitis-ai			
	CNV	mCNN	VGG11	LFC
Filtros pre-entrnados	Si (VGG11)	Si (VGG11)	Si (VGG11)	No
Tipo de capas	nn.Layers	nn.Layers	nn.Layers	nn.Layers
Épocas	50	50	50	50
Tasa de Aprendizaje	0.2	0.2	0.2	0.2
Tamaño de lote	100	100	100	100
Función de pérdidas	CrossEntropy	CrossEntropy	CrossEntropy	CrossEntropy
Función de activación	ReLu	ReLu	ReLu	ReLu
Optimizador	SGD	SGD	SGD	SGD
Bits	32	32	32	32

Tabla II: Hiperparámetros utilizados en Pytorch/Vitis-AI

II-D. Procedimiento en FINN

El procedimiento en FINN se llevó a cabo en el contenedor Docker FINN 0.9 dev en Ubuntu 18.04. Se utilizó la versión 2022.2 de Vivado y Vitis HLS. A continuación se describen los pasos:

- Construcción y entrenamiento del modelo utilizando las capas cuantizadas de Brevitas. De acuerdo a las recomendaciones de la documentación de FINN se entrenó 1000 épocas. El modelo se almacena en un archivo .tar y es exportado al formato ONNX con Brevitas.
- Estimación de los recursos (LUTs, BRAM, DSP y URAM) y del rendimiento a partir del constructor $build\_dataflow$. Para este paso es necesario definir los parámetros de diseño, cómo el tiempo de reloj, el dispositivo objetivo, y los FPS objetivo. Para todos los casos

se utilizó 10 ns como tiempo de reloj, $KV260\_SOM$ como dispositivo y 10000 FPS. Vale la pena aclarar que los FPS objetivo es solo una condición de diseño, los FPS resultantes del modelo dependerán de la optimización de los recursos. Para este trabajo, se utilizó el auto plegado o auto folding que ofrece el constructor. Aunque es posible mejorar el rendimiento de estos aceleradores ajustando manualmente el plegado, este trabajo se centra en el estudio de entornos de desarrollo automatizado.
- Construcción del modelo con $build\_dataflow$ que consta de generación de código hls, generación del IP, síntesis del archivo con extensión bit y generación del driver. Estos últimos archivos son copiados a la memoria SD de la placa de desarrollo Kria KV260.

II-E. Procedimiento en Vitis-AI

En Vitis-AI, el procedimiento se llevó a cabo en el contenedor Docker Vitis-AI-2.5.0 en Ubuntu 18.04 y constó de los siguientes pasos:

- Carga del modelo de punto flotante y cuantización a INT8 con la función $torch\_quantization$ de PyTorch.
- Evaluación de la precisión del modelo cuantizado.
- Compilación del modelo cuantizado para generar las instrucciones de la DPU utilizando vai_c_xir, es necesario definir el modelo del DPU, en este caso se utilizó DPUCZDX8G_ISA1_B4096_MAX [6] porque es compatible con la versión 2.5 de Vitis AI, y está optimizada para la familia Zynq UltraScale+ MPSoC de Xilinx. En este punto se genera un archivo extensión xmodel y se copia en la memoria SD de la placa Kria KV260.

Si bien Vitis AI ofrece Vitis AI Optimizer [11] para reducir la complejidad del modelo, a partir de técnicas de poda, reduciendo pesos redundantes y manteniendo la pérdida de precisión lo más baja posible. Esta herramienta no fue utilizada en este trabajo de comparación, ya que al podar el modelo se está cambiando la cantidad de filtros, pesos y activaciones.

II-F. Preparación de placa de desarrollo Kria KV260

Para evaluar ambos métodos en un mismo sistema operativo, se optó por instalar en la tarjeta SD de la placa de desarrollo Xilinx Kria KV260 el sistema operativo Ubuntu 20.04.3 LTS, especialmente desarrollado para estas placas. Además, se instalaron las bibliotecas KRIA-PYNQ, el cual contiene PYNQ-DPU, que permite instanciar una DPU, cargando primero el overlay de la DPU y el archivo con extensión xmodel compilado por el Vitis AI. Y por último, se instaló la biblioteca FINN-EXAMPLES, la cual contiene los drivers para cargar los overlays en extensión bit generados por FINN.

II-G. Métricas

Se utilizaron las siguientes métricas:

- Precisión TOP1: Se calcula como el porcentaje de imágenes para las cuales el modelo predice correctamente la clase correcta como la primera opción entre todas las clases posibles.
- Latencia: El tiempo desde el inicio hasta el final de la inferencia para una sola imagen.
- Rendimiento: Es la cantidad de imágenes que se infieren en un segundo.
- Potencia media: Se calcula la potencia instantánea a partir de la lectura de los sensores de corriente y tensión de la placa de desarrollo. Luego, se realiza la inferencia completa del conjunto de datos de evaluación correspondiente a cada modelo analizado y se calcula la potencia media.
- Eficiencia Energética: Se calcula a partir del cociente entre el rendimiento y la potencia media.

III. Resultados

En la tabla III se observan los recursos utilizados para cada modelo. En Vitis AI, todos los modelos son ejecutados en el DPU (4096), en este caso se utiliza el 44 % de los LUTs, disponible, esta versión de DPU no utiliza BRAM, el 45 % de los DPS y el 100 % de la URAM. En la misma tabla se observan los recursos utilizados para cada modelo, en el caso de los modelos mCNN, CNV y VGG11 se nota que a medida que aumentan la cantidad de capas (8,9 y 11 respectivamente) aumenta el uso de LUTs.

	Disp.	Vitis-AI	FINN			
	KV260	DPU	mCNN	CNV	VGG11	LFC
LUTs	117120	51843	25492	39589	71574	18445
BRAM	144	0	34	142	127	112
DSP	1248	566	0	0	0	0
URAM	64	64	0	0	33	0

Tabla III: Recursos disponibles y utilizados por cada modelo

En la tabla IV se observa los resultados obtenidos en FINN y en la tabla V se observan los resultados obtenidos en Vitis AI. En ambos casos se muestran la precisión TOP1, la latencia, el rendimiento y la eficiencia energética.

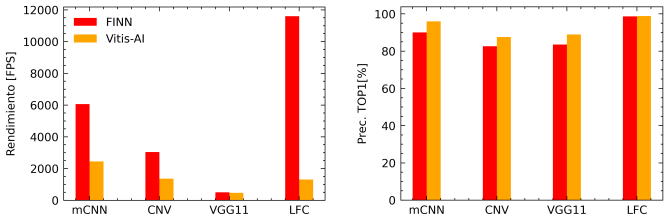
Modelo	Dataset	Prec. TOP1 [%]	Latencia [ms]	Rend. [FPS]	P [W]	Efic. [FPS/W]
mCNN	SVHN	90.07	0.16496	6062	3.8	1595
CNV	CIFAR10	82.62	0.3291	3038	3.9	779
VGG11	CIFAR10	83.51	1.9801	505	4.3	116
LFC	MNIST	98.69	0.0862	11597	3.7	3134

Tabla IV: Resultados obtenidos en FINN

Modelo	Dataset	Prec. TOP1 [%]	Latencia [ms]	Rend. [FPS]	P [W]	Efic. [FPS/W]
mCNN	SVHN	96.05	0.922	1315	3.7	355
CNV	CIFAR10	87.57	0.648	2455	5.9	413
VGG11	CIFAR10	88.92	2.400	476	7.2	66
LFC	MNIST	98.83	0.9011	1368	6.5	210

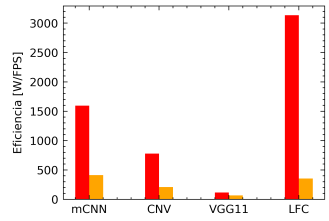
Tabla V: Resultados obtenidos en Vitis-AI

En la figura 2 se observan gráficas de barra que comparan las métricas obtenidas por la implementación en FINN y en Vitis AI de cada modelo, siendo FINN representado por el color rojo y Vitis AI por el color naranja.

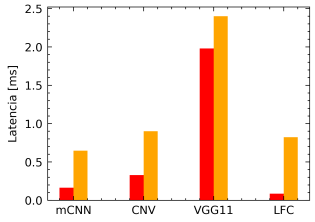


(a) Rendimiento

(b) Precisión TOP1



(c) Eficiencia



(d) Latencia

Figura 2: Comparación de resultados FINN versus Vitis AI

En la figura 2b se observa que en la mayoría de los casos, los modelos desarrollados en Vitis AI obtienen mayor precisión, esto es debido a que en FINN se utilizó una cuantización binaria, y esto produce una perdida en la precisión, respecto a la cuantización INT8 de Vitis AI.

En la figura 2a se observa que FINN tiene un mayor rendimiento que Vitis AI, pero también se nota que a medida que aumentan la cantidad de capas de convolución, la diferencia de rendimiento diminuye, como el caso de VGG11. En el caso de LFC, modelo que solo contiene capas totalmente conectadas, el rendimiento mejora en más de 8 veces. En la figura 2c se observa que la eficiencia es notablemente mayor en los modelos implementados en FINN, pero en los modelos convolucionales a medida que aumentan las capas, las diferencias se acortan.

En la figura 2d, se observa que en todos los casos FINN obtiene menor latencia que Vitis AI. En el caso de las CNN, a medida que aumentan las capas, se disminuye la diferencia.

IV. Conclusiones

En este trabajo, se implementaron tres modelos de redes nueronales convolucionales y uno solamente con capas totalmente conectaras para clasificación de imágenes en la placa de desarrollo de FPGA Kria KV260. Se utilizaron FINN y Vitis AI como entornos de desarrollo automatizados y cada modelo se implementó desde el entrenamiento hasta la evaluación en la FPGA, analizando pasos, recursos, rendimiento, latencia y eficiencia energética. Se observó que los modelos implementados con FINN presentaron menor latencia, mejor rendimiento y mayor eficiencia energética que los implementados con Vitis AI. Este trabajo añade a la literatura comparativa al analizar tres modelos basados en CNN con diferente complejidad (8, 9 y 11 capas) y un modelo con redes completamente conectadas.

La eficiencia energética y el uso de recursos difieren significativamente entre FINN y Vitis AI, especialmente con modelos de pocas capas, donde FINN adapta el hardware al recurso, mientras que Vitis AI implementa el DPU. En todos los casos, FINN presenta una mayor eficiencia energética que Vitis AI, aunque, con un mayor número de capas de convolución, estas diferencias se reducen.

Si bien los modelos implementados en Vitis AI superan en precisión a los implementados en FINN, esto se debe a que se utilizan diferentes cuantizaciones, INT8 y binaria, respectivamente. Pero Vitis AI ofrece la ventaja de ser entrenado con capas de PyTorch, lo que permite utilizar pesos y activaciones preentrenados o aplicar transferencia de aprendizaje, reduciendo la cantidad de épocas necesarias para entrenar.

Aunque FINN ofrece un constructor automatizado, para optimizar el rendimiento es necesario realizar un plegado a medida, este proceso no se encuentra automatizado, y modifica los recursos de hardware utilizados. En cambio, Vitis AI ofrece una herramienta Vitis AI Optimizer, la cual realiza una poda del modelo, optimizando sin modificar los recursos de hardware.

Por lo tanto, para aplicaciones que requieran alta eficiencia energética y alto rendimiento, se recomienda el desarrollo con FINN, aunque se necesita un proceso de plegado a medida, y en algunos casos será necesario adaptar el modelo a los recursos de hardware. Para aplicaciones donde el rendimiento y la eficiencia energética no sean tan críticos, se recomienda utilizar Vitis AI debido a que presenta un flujo de desarrollo más sencillo y herramientas de optimización.

Referencias

[1] X. Zhao, L. Wang, Y. Zhang, X. Han, M. Deveci, and M. Parmar, "A review of convolutional neural networks in computer vision," *Artificial Intelligence Review*, vol. 57, no. 4, pp. 1–43, 2024.

[2] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," *Journal of Machine Learning Research*, vol. 18, pp. 1–30, 2018.

[3] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized Neural Networks: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1," *arXiv: Learning*, 2016. [Online]. Available: http://arxiv.org/abs/1602.02830

[4] XILINX, "Vitis ai - adaptable & real-time ai inference acceleration," 2022. [Online]. Available: https://github.com/Xilinx/Vitis-AI

[5] Y. Umuroglu, N. J. Fraser, G. Gambardella, M. Blott, P. Leong, M. Jahre, and K. Vissers, "FINN: A framework for fast, scalable binarized neural network inference," *FPGA 2017 - Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, no. February, pp. 65–74, 2017, doi: 10.1145/3020078.3021744.

[6] Xilinx, "Dpuczdx8g for zynq ultrascale+ mpsocs product guide (pg338)," 2023. [Online]. Available: https://docs.xilinx.com/r/en-US/pg338-dpu/Core-Overview

[7] M. Blott, T. B. Preuber, N. J. Fraser, G. Gambardella, K. O'Brien, Y. Umuroglu, M. Leeser, and K. Vissers, "FinN-R: An end-to-end deep-learning framework for fast exploration of quantized neural networks," *ACM Transactions on Reconfigurable Technology and Systems*, vol. 11, no. 3, 2018, doi: 10.1145/3242897.

[8] M. Machura, M. Danilowicz, and T. Kryjak, "Embedded object detection with custom littlenet, finn and vitis ai dcnn accelerators," *Journal of Low Power Electronics and Applications*, vol. 12, no. 2, 2022. [Online]. Available: https://www.mdpi.com/2079-9268/12/2/30

[9] F. Hamanaka, T. Odan, K. Kise, and T. V. Chu, "An exploration of state-of-the-art automation frameworks for fpga-based dnn acceleration," *IEEE Access*, vol. 11, pp. 5701–5713, 2023.

[10] S. Liu and W. Deng, "Very deep convolutional neural network based image classification using small training sample size," *Proceedings - 3rd IAPR Asian Conference on Pattern Recognition, ACPR 2015*, pp. 730–734, 2016, doi: 10.1109/ACPR.2015.7486599.

[11] Xilinx, "Vitis ai optimizer," 2023. [Online]. Available: https://docs.amd.com/r/en-US/ug1414-vitis-ai/Vitis-AI-Optimizer