AJEA (Actas de Jornadas y Eventos Académicos de UTN)
DOI: https://doi.org/10.33414/ajea.1721.2024
Publicado / Published: 10/2024

Implementación de CNN en FPGA con entornos automatizados para visión por computadora

CNN implementation on FPGA with automated environments for computer vision

Presentación: 08/10/2024

Doctorando:
Nicolás URBANO PINTOS
Grupo de Tecnología Aplicada al Medio Ambiente – Facultad Regional Haedo- Universidad Tecnológica Nacional - Argentina
nurbano@frh.utn.edu.ar

Director:
Mario Blas LAVORATO

Codirector:
Héctor Alberto LACOMI

Resumen

Las Redes Neuronales Convolucionales (CNN) son esenciales en aplicaciones de visión por computadora, pero su implementación en sistemas embebidos es desafiante debido a sus altas demandas de memoria y cómputo. Para abordar esto, se emplean técnicas como la cuantización, permitiendo la ejecución de modelos en hardware embebido, como FPGA, que ofrecen eficiencia energética y flexibilidad. Entornos de desarrollo automatizados como Vitis AI y FINN de Xilinx facilitan la implementación de CNN en FPGA.

Este trabajo compendia modelos de CNN en FPGA usando Vitis AI y FINN para clasificación de imágenes y detección de objetos. Se revisan bibliografía y trabajos previos, describiendo diferencias arquitectónicas, procedimientos de construcción y evaluación de modelos, y analizando rendimiento y eficiencia energética, destacando las virtudes y limitaciones de cada entorno.

Palabras clave: FPGA; Aprendizaje Profundo, Redes neuronales de convolución; Visión por computadora.

Abstract

Convolutional Neural Networks (CNN) are essential in computer vision applications, but their implementation in embedded systems is challenging due to their high memory and computational demands. To address this, techniques such as quantization are employed, allowing the execution of models on embedded hardware, such as FPGAs, that offer energy efficiency and flexibility. Automated development environments such as Vitis AI and FINN from Xilinx facilitate the implementation of CNNs on FPGAs.

This work summarizes CNN models on FPGAs using Vitis AI and FINN for image classification and object detection. Literature and previous work are reviewed, describing architectural differences, model construction and evaluation procedures, and analyzing performance and energy efficiency, highlighting the virtues and limitations of each environment.

Keywords: FPGA; Deep Learning; Convolutional Neuronal Network, Computer vision.

AJEA (Actas de Jornadas y Eventos Académicos de UTN)
DOI: https://doi.org/10.33414/ajea.1721.2024
Publicado / Published: 10/2024

Introducción

Las Redes Neuronales Convolucionales (CNN - Convolutional Neural Networks) son fundamentales en aplicaciones de visión por computadora, tales como la detección de defectos en producción (Jha & Babiceanu, 2023), la detección y clasificación de obstáculos en vehículos autónomos (Gao et al., 2018) y la estimación de profundidad para robótica (Fang et al., 2020).

Las CNN extraen características de las imágenes de entrada mediante el uso de filtros o kernels de convolución, los cuales se aplican a regiones de las imágenes generando mapas de características de salida. Estos mapas contienen características de bajo nivel, como puntos y líneas. Conforme se aplican más filtros sucesivamente, se obtienen características de alto nivel, como formas y texturas. La cantidad de capas de convolución depende de la arquitectura del modelo utilizado, y a medida que el modelo es más profundo (con más capas), se necesita almacenar una mayor cantidad de parámetros de los filtros, lo que conlleva un mayor requerimiento de memoria y un aumento en las operaciones.

Generalmente, los parámetros de los filtros se almacenan en variables de punto flotante de 32 bits (FP32). Por ejemplo, una red de clasificación como VGG16 (Simonyan & Zisserman, 2014), que contiene 13 capas de convolución, requiere 552 MB de memoria y 30.8 giga operaciones de punto flotante. Esto representa una complicación a la hora de implementar dichos modelos en sistemas embebidos.

Existen diferentes enfoques y técnicas para reducir la cantidad de memoria de una red CNN, como la poda, que elimina neuronas que no tienen un peso significativo en la salida del sistema. Sin embargo, el enfoque central es la cuantización, que consiste en representar los parámetros de los filtros y otras etapas de las CNN con variables de menor precisión. Los enfoques de cuantización incluyen el uso de variables de punto flotante de 16 bits (FP16), de enteros de 8 bits (INT8) (Hubara et al., 2018), de enteros de 4 bits (INT4) y de variables binarias (Courbariaux et al., 2016).

Los modelos cuantizados pueden ser ejecutados en diferentes plataformas de sistemas embebidos, como procesadores ARM, procesadores Tegra (Nvidia Jetson) y FPGA (Field Programmable Gate Array). Las FPGA son dispositivos de alta eficiencia energética en comparación con las CPU y las GPU, y son más adaptables que los ASIC porque se pueden reprogramar fácilmente, por ese motivo surgen como una alternativa destacable para la implementación de estos modelos.

Implementar estas redes en FPGA es una tarea compleja, por lo que han surgido diferentes entornos de trabajo que automatizan el proceso de desarrollo. Entre estos destacan Vitis AI (GitHub - Xilinx/Vitis-AI) y FINN (GitHub - Xilinx/Finn).

En este trabajo se realiza un compendio de modelos de redes neuronales convoluciones implementadas en FPGA con Vitis AI y FINN para tareas de clasificación de imágenes y detección de objetos, incluyendo una revisión bibliográfica y trabajos previos de los autores. A su vez, se detallan las diferentes arquitecturas de cada uno de estos entornos, y se describe el procedimiento para construir y evaluar los modelos, destacando las virtudes y falencia de cada entorno. Por otra parte, se lleva a cabo un análisis del rendimiento y la eficiencia energética de cada modelo implementado.

Antecedentes

Entornos de desarrollo automatizados

Vitis AI, implementa módulo 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 (Deep Learning Processor Unit (DPU)).En el DPU se ejecuta un microcódigo a través de un archivo con extensión xmodel generado por el compilador del Vitis AI. El DPU busca instrucciones en la memoria externa (SD), analiza y programa las instrucciones, y opera el motor de cálculo. El motor de convolución ejecuta las convoluciones, encargándose también del agrupamiento y la normalización de lotes. Las operaciones (multiplicador, sumador, acumulador) están implementadas en los denominados elementos de procesamiento.

AJEA (Actas de Jornadas y Eventos Académicos de UTN)
DOI: https://doi.org/10.33414/ajea.1721.2024
Publicado / Published: 10/2024

En Vitis-AI, el procedimiento se ejecuta en el contenedor Docker Vitis-AI(GitHub - Xilinx/Vitis-AI.) en un sistema operativo Linux, preferentemente con distribución Ubuntu. El modelo puede ser entrenado en cualquier entorno de trabajo actual de aprendizaje profundo, como PyTorch, TensorFlow o Caffe. Una vez entrenado, es necesario cuantizar el modelo de punto flotante a INT8. En PyTorch, este proceso se realiza dentro del entorno de PyTorch en el Docker de Vitis AI utilizando *torch_quantization*.

El siguiente paso es compilar el modelo cuantizado. En el caso de PyTorch, se utiliza la función *vai_c_xir*, donde es necesario definir el modelo DPU compatible con las diferentes familias de FPGA. Este proceso genera un archivo con extensión xmodel, que se copia en la memoria SD de la placa de desarrollo de la FPGA.

Finalmente, para evaluar el modelo, se debe crear una aplicación, generalmente en Python, que instancie el DPU y cargue el modelo utilizando el conjunto de instrucciones del archivo xmodel.

Por otra parte, FINN utiliza una arquitectura de tipo de flujo de datos e implementa las CNN a partir de tres componentes principales: una unidad de umbral vectorial de matriz, una unidad de ventana deslizante y una unidad de agrupamiento (Blott et al., 2018). A través de factores de plegado se determina la cantidad de elementos de procesamiento y carriles que realizan las operaciones en paralelo.

A diferencia del DPU de Vitis AI, que utiliza una cantidad fija de recursos de la FPGA determinado por el modelo del DPU, FINN implementa los modelos de acuerdo con la configuración de restricciones específicas, como el rendimiento (cuadros por segundo) y la frecuencia de operación. Por lo tanto, la cantidad de recursos depende no solo del modelo a implementar, sino también de los requerimientos de ejecución de este.

Para llevar a cabo el procedimiento de implementación de redes CNN en FINN es necesario construir el contenedor Docker FINN (GitHub - Xilinx/Finn) en un sistema operativo Linux, preferentemente en distribución Ubuntu. Es necesario también instalar los softwares de Xilinx Vivado y Vitis HLS.

Para implementar CNN con FINN se debe construir y entrenar un modelo con capas cuantizadas utilizando Brevitas (Pappalardo, 2021)durante al menos 500 épocas. En esta etapa, se define el tipo de cuantización, ya sea binaria, ternaria, INT4 o INT8. El modelo entrenado se exporta al formato ONNX.

Para evaluar los recursos utilizados en la FPGA, como LUTs, BRAM, DSP y URAM, así como el rendimiento, se emplea el constructor *build_dataflow*. Es necesario definir los parámetros de diseño, incluyendo el tiempo de reloj, el dispositivo a utilizar y el rendimiento en FPS deseado. Aunque el constructor ofrece un auto plegado para automatizar el proceso, se pueden realizar ajustes manuales para mejorar el rendimiento.

Posteriormente, se genera el código HLS, el IP, el archivo con extensión bit y el controlador utilizando el constructor *build_dataflow*. Los archivos resultantes deben ser copiados a la memoria SD de la placa.

Por último, una vez dentro del sistema operativo Linux de la placa de desarrollo, es necesario instanciar el archivo con extensión bit y crear una aplicación para verificar y evaluar el rendimiento, la latencia, etc.

Trabajos Previos

Umuroglu et al., implementan con FINN tres modelos de clasificación basados CNN a partir del FINN, analizando los recursos utilizados por cada modelo, la latencia, la potencia, el rendimiento y la eficiencia energética.

Ducasse et al. experimentan con el entorno FINN con una red de perceptrón multicapa (MLP) entrenada en los conjuntos de datos MNIST y Fashion-MNIST, analizando la implicancia de la cantidad de bits de cuantización de activaciones y de pesos. Los resultados muestran que las implementaciones con menor precisión en bits pueden alcanzar una precisión comparable a sus contrapartes de mayor precisión con suficiente entrenamiento, a la vez que mantienen una baja utilización de hardware y alta eficiencia.

Utilizando FINN y Vitis AI y un acelerador customizado, Machura et al. implementaron dos modelos de detección basados en CNN, YoloFINN y LittleNet, evaluando el uso de energía, el rendimiento y la precisión de

AJEA (Actas de Jornadas y Eventos Académicos de UTN)
DOI: https://doi.org/10.33414/ajea.1721.2024
Publicado / Published: 10/2024

la detección de objetos en una placa de desarrollo Avnet Ultra96-V2. Obteniendo mejor rendimiento con su modelo customizado, pero con un flujo de desarrollo mucho más complejo. A su vez, lograron el menor uso de recursos con FINN.

Hamanaka et al. utilizaron un modelo de clasificación denominado ResNet-8 implementado en una FPGA SOM K26 de Xilinx, encontrando que FINN supera al acelerador basado en Vitis AI en términos de latencia, eficiencia energética y rendimiento. Los resultados muestran FINN supera al acelerador basado en Vitis AI, en latencia, rendimiento y eficiencia energética.

Urbano Pintos et al. realizaron una comparación de Vitis AI y FINN a partir de 3 modelos de clasificación basados en CNN implementados en la placa KV260 con diferentes cantidades de capas, haciendo un análisis de recursos utilizados, latencia, potencia y eficiencia energética. Donde obtuvieron que a medida que el modelo se hace más complejo, la diferencia de rendimiento entre FINN y Vitis AI se reduce.

Conjunto de datos

A continuación, se detallan los datasets que se utilizaron en este análisis:

- CIFAR-10 (Krizhevsky, 2009) contiene 60,000 imágenes a color de 32x32 píxeles, divididas en 10 clases diferentes, con 6,000 imágenes por clase. Las clases incluyen vehículos y animales. Se encuentra dividido en 50,000 imágenes de entrenamiento y 10,000 imágenes de prueba.

- El conjunto de datos SVHN (Netzer et al., 2011.) contiene imágenes de números de casas extraídas de Google Street View. Está compuesto por más de 600,000 imágenes a color de 32x32 píxeles, que contienen dígitos de números de casas en diferentes condiciones de iluminación y distorsión.

- El MNIST (Alsaafin et al., 2017) 70,000 imágenes en escala de grises de 28x28 píxeles de dígitos escritos a mano (del 0 al 9), divididas en 60,000 imágenes de entrenamiento y 10,000 imágenes de prueba.

- El conjunto de datos VOT (Kristan et al., 2016)contiene secuencias de video anotadas con las trayectorias de los objetos a seguir. Cada secuencia incluye marcos en los que se ha identificado y delimitado el objeto.

Arquitecturas

A continuación, se detallan los modelos estudiados en este trabajo:

- CNN (Urbano Pintos et al., 2024): Tiene una arquitectura formada por 3 conjuntos de capas de convolución 3x3, convolución 3x3 y maxpool 2x2 y una capa totalmente conectada de 128 neuronas. Acepta imágenes a color de 32x32 píxeles y devuelve un vector de 10 elementos como resultado.

- CNV (Umuroglu et al., 2017): Es una red convolucional inspirada en BinaryNet y VGG-16. Consiste en una secuencia de capas (convolución 3x3, convolución 3x3, maxpool 2x2) repetidas tres veces con 64-128-256 canales, seguidas por dos capas completamente conectadas de 512 neuronas cada una. Acepta imágenes a color de 32x32 píxeles y devuelve un vector de 10 elementos como resultado.

- VGG11 (Simonyan & Zisserman, 2014): Es una versión reducida de VGG16, especialmente adaptada para el conjunto de datos CIFAR10. Acepta imágenes RGB de 32x32 píxeles. Consiste en 8 capas de convolución, una capa de 64 filtros, otra de 128, 2 capas de 256, y 4 capas de 512. Para la extracción de característica utiliza 3 capas totalmente conectadas, de 1024 neuronas.

- VGG16 (Simonyan & Zisserman, 2014): Es una red neuronal profunda diseñada para la clasificación de imágenes en el conjunto de datos CIFAR. Acepta imágenes RGB de 32x32 píxeles. Consiste en 13 capas de convolución organizadas en 5 bloques, con filtros de 64, 128, 256 (en 2 capas), y 512 (en 3 capas) respectivamente. Además, cuenta con 3 capas totalmente conectadas al final, cada una con 4096 neuronas. La salida es un vector de 10 elementos.

AJEA (Actas de Jornadas y Eventos Académicos de UTN)
DOI: https://doi.org/10.33414/ajea.1721.2024
Publicado / Published: 10/2024

- LFC (Umuroglu et al., 2017): Un clasificador totalmente conectado de 4 capas de 1024 neuronas. Acepta imágenes en escala de grises de 28x28 píxeles y devuelve un vector de 10 elementos como resultado. Esta arquitectura fue diseñada por los desarrolladores de FINN.

- ResNet8 (He et al., 2015): Es una versión compacta de la arquitectura ResNet, adaptada para el conjunto de datos CIFAR10. Acepta imágenes RGB de 32x32 píxeles. Consiste en 8 capas de convolución organizadas en 3 bloques de, con 16 filtros en el primer bloque, 32 en el segundo, 64 en el tercero, Para la extracción de características, utiliza una capa de activación tipo ReLU y una capa agrupamiento promedio.

- YOLOFINN (Machura et al., 2022): Es una arquitectura adaptada para la detección de objetos en imágenes con un enfoque en la eficiencia computacional. Acepta imágenes RGB de 160x320 píxeles. Comienza con una capa de convolución de 16 filtros, seguida por una capa de Clamping y una capa de MaxPooling2D. Esta secuencia se repite con capas de 32, 64, 64, cada una intercalada con capas de clamping y MaxPooling2D. Luego, el modelo incluye una capa con 30 filtros.

Desarrollo

Este trabajo es un compendio de diferentes implementaciones de CNN con Vitis AI y FINN de la bibliografía y de los mismos autores. Adicionalmente, en este trabajo, se entrenó una red VGG16 con el conjunto de datos CIFAR-10 utilizando un modelo preentrenado con ImageNet a partir del entorno Pytorch. Se utilizó la técnica de transferencia de aprendizaje, congelando los parámetros de las capas de extracción de características y reentrenando solo las capas de clasificación. La última capa de clasificación se adaptó para las 10 clases del conjunto de datos CIFAR-10.

El modelo fue reentrenado en una GPU durante cinco épocas, lo cual tomó aproximadamente 25 minutos. El proceso de cuantización a INT8 se llevó a cabo utilizando la herramienta Vitis AI, y se compiló las instrucciones de DPU para la placa de desarrollo de FPGA Kria KV260.

A su vez, se intentó implementar este modelo con FINN, para ellos se construyó la arquitectura con capas de la biblioteca Brevitas, y se realizó una cuantización QAT de 1 bits de pesos y activaciones. Luego se utilizó el constructor de FINN con la opción de auto-plegado, pero no fue posible culminar el proceso, debido a errores de falta de recursos de memoria Block RAM al compilar, por ese motivo no fue incluido en este trabajo.

Resultados

En la tabla 1 se observa una compilación de los diferentes trabajos citados en este estudio, y de la implementación de la red VGG16. Se detalla el modelo empleado, los autores que lo implementaron, el conjunto de datos usados, el entorno de desarrollo, el dispositivo y la cantidad de bits utilizadas. Cómo resultados se muestran el rendimiento, en cuadros por segundos, la latencia, que es el tiempo que tarda el sistema en inferir una imagen, en milisegundos, y la eficiencia energética que es la relación entre rendimiento y potencia.

En los modelos empleados para clasificación de imágenes, mCNN, CNV y VGG11 se observa que los modelos implementados en FINN superan en rendimiento y eficiencia a los implementados en Vitis AI. Aunque se puede apreciar que a medida que el modelo contiene más capas, como es el caso de VGG11, la diferencia de rendimiento se reduce, pero la eficiencia energética sigue siendo superior en FINN. Respecto a modelo VGG16, solo se pudo implementar en Vitis AI, debido a la limitación de recursos para implementar con auto plegado en FINN.

De manera similar a los modelos descritos anteriormente, ResNet implementado por Hamanaka et al. obtiene mejor rendimiento en FINN, y mayor eficiencia energética. En el caso de la detección de objetos, YOLOFINN implementado por Machura et al. también FINN obtiene mayor rendimiento y eficiencia.

AJEA (Actas de Jornadas y Eventos Académicos de UTN)
DOI: https://doi.org/10.33414/ajea.1721.2024
Publicado / Published: 10/2024

Tarea	Modelo	Implementación	Dataset	Entorno	Dispostivo	Cuantización (bits)	Rendimiento (FPS)	Latencia (ms)	Eficiencia (FPS/W)
Clasificación	mCNN	Urbano Pintos et al.	SVHN	FINN	Xilinx Kria KV260	1	6062	0,165	1595,26
				VITIS AI		8	1315	0,922	355,41
	CNV		CIFAR10	FINN		1	3038	0,329	778,97
				VITIS AI		8	2455	0,648	416,10
	VGG11			FINN		1	505	1,980	117,44
				VITIS AI		8	476	2,401	66,11
	VGG16			VITIS AI		8	180	5,501	18,94
	LFC		MNIST	FINN		1	11597	0,086	3134,32
				VITIS AI		8	1368	0,901	210,46
	Resnet8	He et al.	CIFAR10	FINN	Xilinx Kria K260	4	13475	0,154	2287,78
				VITIS AI		8	4458	1,293	694,39
Detección	YoloFINN	Machura et al.	VOT	FINN	Avnet Ultra96-V2	4	111	9,009	37,58
				Vitis AI		8	53	18,796	17,10

Tabla 1: Compendio de resultados de implementación de CNN en FPGA con FINN y VITIS AI

Conclusiones

Las Redes Neuronales Convolucionales (CNN) son esenciales para aplicaciones de visión por computadora, pero su implementación en sistemas embebidos presenta desafíos debido a los altos requisitos de memoria y cómputo. Este estudio exploró la implementación de CNN en FPGA utilizando Vitis AI y FINN, comparando su rendimiento y eficiencia energética en tareas de clasificación de imágenes y detección de objetos.

Los resultados demostraron que las FPGA, gracias a su eficiencia energética y flexibilidad, son una plataforma viable para CNN cuantizadas. FINN generalmente mostró un mejor rendimiento y eficiencia energética en la mayoría de los modelos estudiados, especialmente en aquellos con menos capas.

A pesar de sus ventajas, FINN enfrentó limitaciones al implementar modelos profundos como VGG16, debido a restricciones de recursos en la FPGA. Vitis AI, aunque menos eficiente en algunos casos, proporcionó una implementación más sencilla y compatible con una amplia gama de modelos preentrenados, permitiendo utilizar técnicas de transferencia de aprendizaje y ajuste fino.

Referencias

Alsaafin, A., Elnagar, A., Alsaafin, A., & Elnagar, A. (2017). A Minimal Subset of Features Using Feature Selection for Handwritten Digit Recognition. Journal of Intelligent Learning Systems and Applications, 9(4), 55–68. https://doi.org/10.4236/JILSA.2017.94006

Blott, M., Preuber, T. B., Fraser, N. J., Gambardella, G., O'Brien, K., Umuroglu, Y., Leeser, M., & Vissers, K. (2018). FinN-R: An end-to-end deep-learning framework for fast exploration of quantized neural networks. ACM Transactions on Reconfigurable Technology and Systems, 11(3). https://doi.org/10.1145/3242897

AJEA (Actas de Jornadas y Eventos Académicos de UTN)
DOI: https://doi.org/10.33414/ajea.1721.2024
Publicado / Published: 10/2024

Courbariaux, M., Hubara, I., Soudry, D., El-Yaniv, R., Bengio, Y., & Com, Y. U. (2016). Binarized Neural Networks: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1. https://arxiv.org/abs/1602.02830v3

Deep Learning Processor Unit (DPU) • Vitis AI User Guide (UG1414) • Reader • AMD Technical Information Portal. (n.d.). Retrieved August 4, 2024, from https://docs.amd.com/r/1.2-English/ug1414-vitis-ai/Deep-Learning-Processor-Unit-DPU

Fang, Z., Chen, X., Chen, Y., & Gool, L. Van. (2020). Towards Good Practice for CNN-Based Monocular Depth Estimation (pp. 1091–1100). https://github.com/

Gao, H., Cheng, B., Wang, J., Li, K., Zhao, J., & Li, D. (2018). Object Classification Using CNN-Based Fusion of Vision and LIDAR in Autonomous Vehicle Environment. IEEE Transactions on Industrial Informatics, 14(9), 4224–4230. https://doi.org/10.1109/TII.2018.2822828

GitHub - Xilinx/finn: Dataflow compiler for QNN inference on FPGAs. (n.d.). Retrieved August 4, 2024, from https://github.com/Xilinx/finn

GitHub - Xilinx/Vitis-AI: Vitis AI is Xilinx's development stack for AI inference on Xilinx hardware platforms, including both edge devices and Alveo cards. (n.d.). Retrieved August 4, 2024, from https://github.com/Xilinx/Vitis-AI

Hamanaka, F., Odan, T., Kise, K., & Chu, T. Van. (2023). An Exploration of State-of-the-Art Automation Frameworks for FPGA-Based DNN Acceleration. IEEE Access, 11, 5701–5713. https://doi.org/10.1109/ACCESS.2023.3236974

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2016-December, 770–778. https://doi.org/10.1109/CVPR.2016.90

Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., & Bengio, Y. (2018). Quantized Neural Networks: Training Neural Networks with Low Precision Weights and Activations. Journal of Machine Learning Research, 18(187), 1–30. http://jmlr.org/papers/v18/16-456.html

Jha, S. B., & Babiceanu, R. F. (2023). Deep CNN-based visual defect detection: Survey of current literature. Computers in Industry, 148, 103911. https://doi.org/10.1016/J.COMPIND.2023.103911

Kristan, M., Matas, J., Leonardis, A., Vojir, T., Pflugfelder, R., Fernandez, G., Nebehay, G., Porikli, F., & Cehovin, L. (2016). A Novel Performance Evaluation Methodology for Single-Target Trackers. IEEE Transactions on Pattern Analysis and Machine Intelligence, 38(11), 2137–2155. https://doi.org/10.1109/TPAMI.2016.2516982

Krizhevsky, A. (2009). Learning Multiple Layers of Features from Tiny Images.

Machura, M., Danilowicz, M., & Kryjak, T. (2022). Embedded Object Detection with Custom LittleNet, FINN and Vitis AI DCNN Accelerators. Journal of Low Power Electronics and Applications 2022, Vol. 12, Page 30, 12(2), 30. https://doi.org/10.3390/JLPEA12020030

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., & Ng, A. Y. (n.d.). Reading Digits in Natural Images with Unsupervised Feature Learning. Retrieved August 4, 2024, from http://ufldl.stanford.edu/housenumbers/

Pappalardo, A. (2021). Xilinx/brevitas. Zenodo. https://doi.org/10.5281/zenodo.3333552

Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings. https://arxiv.org/abs/1409.1556v6

AJEA (Actas de Jornadas y Eventos Académicos de UTN)
DOI: https://doi.org/10.33414/ajea.1721.2024
Publicado / Published: 10/2024

Umuroglu, Y., Fraser, N. J., Gambardella, G., Blott, M., Leong, P., Jahre, M., & Vissers, K. (2017). 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, February, 65–74. https://doi.org/10.1145/3020078.3021744

Urbano Pintos, N., Lacomi, H. A., & Lavorato, M. B. (2024). Comparación de Vitis AI y FINN para la implementación de redes neuronales de convolución en FPGA. Congreso Argentino de Sistemas Embebidos.