Implementación sobre FPGA de redes neuronales convolucionales para clasificación de imágenes

UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

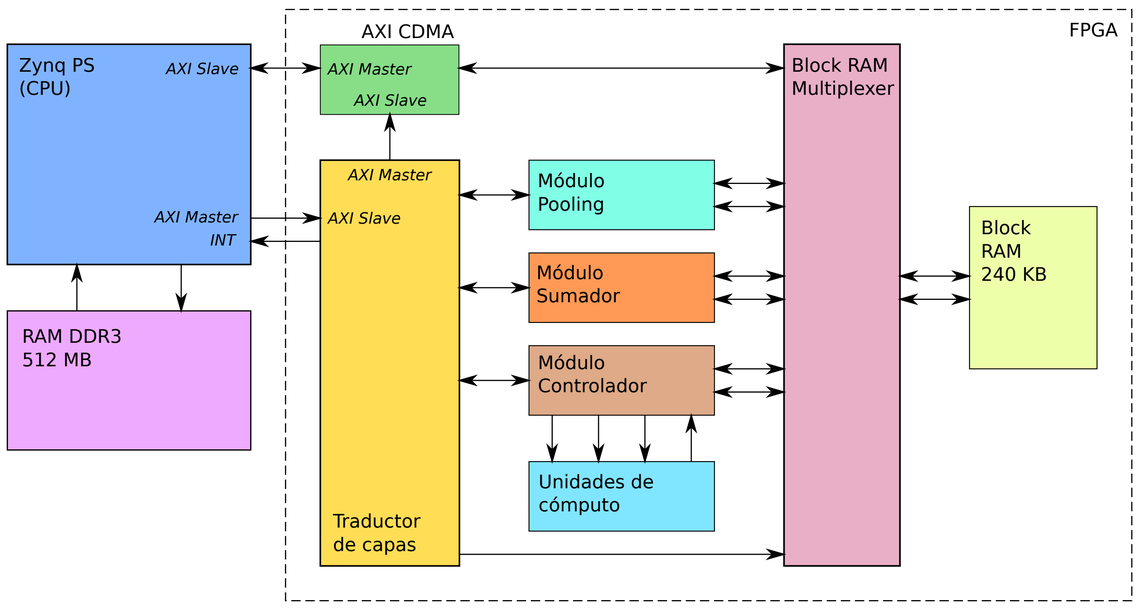
Federico Nicolás Peccia

Departamento de Electrónica
Universidad Tecnológica Nacional FRBA
peccfederico@frba.utn.edu.ar

Introducción

En los últimos años, el procesamiento de imágenes experimentó un avance notable debido a la popularidad de los algoritmos denominados comúnmente como *Redes Neuronales Convolucionales*. Pese a que su uso comercial aumenta cada día, existe un amplio campo de investigación orientado hacia el área de IOT que busca desarrollar plataformas donde estos algoritmos logren ser ejecutados con el menor consumo de energía posible y con tiempos de inferencia por imagen aceptables. Dentro de este marco, el presente trabajo se enfocó en desarrollar una plataforma genérica que permita ejecutar una red neuronal convolucional sobre un FPGA, y además comparar la influencia de distintas técnicas de entrenamiento sobre el comportamiento del modelo ejecutado en el diseño de hardware desarrollado.

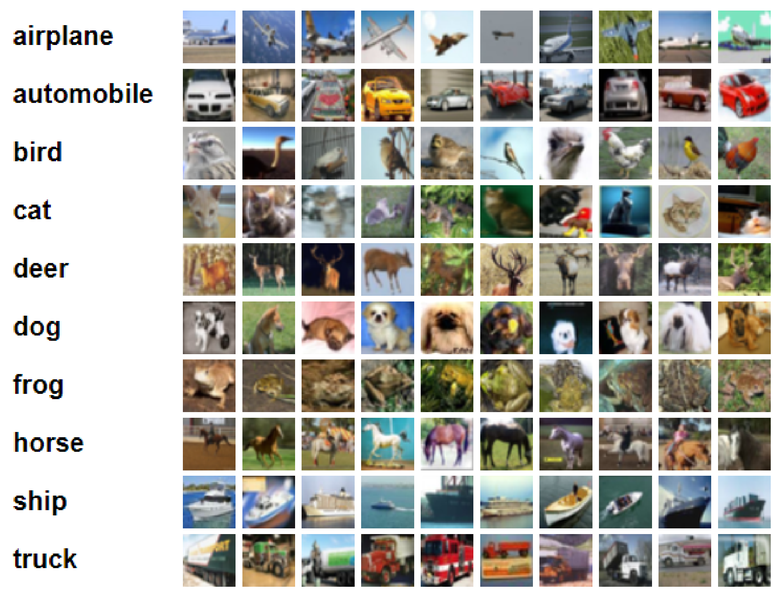
Diagrama en bloques



- **CPU:** ejecuta un sistema operativo Linux y tiene como tarea obtener la imagen a analizar, avisarle al hardware a través de un driver para que de inicio al procesamiento, y esperar el resultado.
- **CDMA:** este bloque realiza transferencias de datos entre la memoria externa y la memoria interna de la FPGA.
- **Unidades de cómputo:** este bloque contiene los módulos DSP que son los encargados de realizar las operaciones matemáticas que necesita una red neuronal convolucional. Internamente, dentro de este bloque se encuentra un diseño digital donde el flujo de datos se encuentra altamente paralelizado, maximizando la reutilización espacial y temporal de datos.
- **Módulos principales:** los módulos controlador, sumador y pooling resuelven cada uno de ellos una capa de una red neuronal clásica. Obtienen los datos necesarios de la memoria interna, realizan la operación y guardan nuevamente el resultado en la misma.
- **Traductor de capas:** este módulo es el que contiene el conocimiento sobre la arquitectura de la red neuronal que se quiere ejecutar (es decir, la sucesión de capas, los tipos de cada capa, los parámetros específicos de cada una, etc), y a su vez se encarga de coordinar el funcionamiento de todos los módulos del desarrollo.

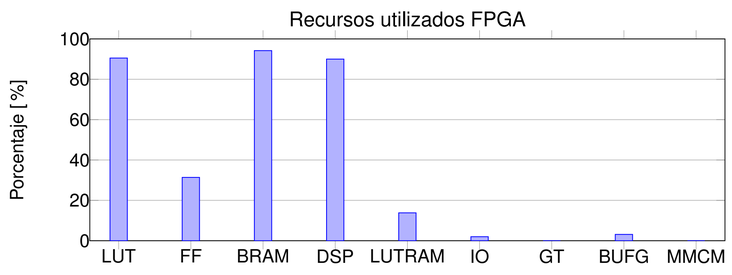
Validación

Se entrenaron 4 versiones de la red neuronal ResNet20 [2, 3] sobre el set de datos CIFAR10 [4]. Este set de datos contiene 60000 imágenes de 32x32x3 píxeles, dispuestas en 10 categorías mutuamente excluyentes, las cuales se encuentran separadas en 50000 imágenes para entrenar la red y 10000 imágenes para validar el entrenamiento realizado.



Cada una de las redes neuronales entrenadas tiene 153178 parámetros (o coeficientes) y debe realizar 0,046 GOP para procesar una imagen.

Resultados



Todas las versiones implementadas utilizan una aritmética de punto fijo de $S_{(3,4)}$ para las entradas y salidas de cada capa convolucional, y de $S_{(2,5)}$ para los coeficientes de la red neuronal. La columna *t* indica el tiempo de procesamiento por imagen, y la columna GOPs el *throughput* (cantidad de giga operaciones por segundo realizadas por el hardware).

- QKeras: modelo entrenado sin aplicar técnicas especiales de entrenamiento
- Pruning: modelo entrenado aplicando la técnica de *pruning* (elimina coeficientes redundantes)
- Selective pruning: eliminación de conjuntos enteros de coeficientes que serían procesados en paralelo por el hardware

Modelo	Exactitud [%]	*t* [ms]	GOPs
QKeras	81,88	21,17	2,173
Pruning	81,81	21,14	2,176
Selective Pruning	79,72	19,34	2,379

Conclusiones

Luego de la investigación y el desarrollo realizados, se logró implementar un acelerador de hardware para redes neuronales convolucionales **completamente funcional**, capaz de ejecutar redes neuronales de una variedad de arquitecturas distintas. Además se verificó que la técnica de *pruning* uniforme **no** acelera la ejecución de la predicción en el hardware. Para lograr esto, hubo que encontrar **otra** técnica de *pruning* que aportó cierta pérdida de exactitud en la predicción, pero que mejoró el tiempo de procesamiento por imagen casi en un 10 %.
Finalmente, se compara el desarrollo realizado con otros aceleradores de hardware. Se presentan distintos desarrollos de implementaciones de redes neuronales convolucionales sobre FPGAs. Cabe destacar que aunque parece haber grandes diferencias entre este proyecto y otros desarrollos, los otros proyectos no solamente trabajan a frecuencias mayores, sino que también utilizan muchos mas DSPs, y también FPGAs con mayor cantidad de recursos.

Desarrollo	Chip FPGA	Fre. [MHz]	DSPs	GOPs
Este desarrollo	XC7Z010	70	72	2,397
[1]	XC7Z020	214	190	84,3
[1]	XC7Z030	150	400	105,2
[5]	5SGSD8	120	1963	117,8
[6]	VX690T	100	1436	222,1

Referencias

[1] K. Guo, L. Sui, J. Qiu, J. Yu, J. Wang, S. Yao, S. Han, Y. Wang, and H. Yang. Angel-eye: A complete design flow for mapping cnn onto embedded fpga. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(1):35–47, 2018.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. *CoRR*, abs/1603.05027, 2016.

[4] Alex Krizhevsky. Learning multiple layers of features from tiny images. *University of Toronto*, 05 2012.

[5] Naveen Suda, Vikas Chandra, Ganesh Dasika, Abinash Mohanty, Yufei Ma, Sarma Vrudhula, Jae-sun Seo, and Yu Cao. Throughput-optimized opencl-based fpga accelerator for large-scale convolutional neural networks. In *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, FPGA '16, page 16–25, New York, NY, USA, 2016. Association for Computing Machinery.

[6] Zhiqiang Liu, Yong Dou, Jingfei Jiang, and Jinwei Xu. Automatic code generation of convolutional neural networks in fpga implementation. In *2016 International Conference on Field-Programmable Technology (FPT)*, pages 61–68, 2016.

Agradecimientos

Al equipo del DPLAB, a los docentes de la materia Introducción a la Inteligencia Artificial y a la cátedra de Proyecto Final