AJEA - Actas de Jornadas y Eventos Académicos de UTN
VI Jornadas de Intercambio y Difusión de los Resultados de Investigaciones de los Doctorandos en Ingeniería
DOI: https://doi.org/10.33414/ajea.1132.2022

Cuantización Binaria en redes neuronales de convolución

Binary Quantization in Convolutional Neural Network

Presentación: 4 y 5 de Octubre de 2022

Doctorando:
Nicolás Urbano Pintos
Grupo Tecnología Aplicada al Medio Ambiente, Facultad Regional Haedo, Universidad Tecnológica Nacional- Argentina
División Radar Laser, Instituto de Investigaciones Científicas y Técnicas para la Defensa - Argentina
nurbano@frh.utn.edu.ar

Director:
Mario Lavorato

Codirector:
Héctor Lacomi

Resumen

En este trabajo se propone implementar una red neuronal binarizada (BNN – Binarized Neural Network) de convolución para clasificar objetos a partir de imágenes RGB. Las BNN reducen la cantidad de recursos computacionales y de memoria, y permiten inferirlas en sistemas embebidos como las matrices de compuertas lógica programables en campo (FPGA – Field Programmable Gate Array) logrando respuestas en tiempo real. El modelo se basa en la red VGG16 (Visual Geometry Group) y se entrena con el conjunto de datos CIFAR10. La red se cuantiza de forma binaria con la técnica de cuantización consciente del entrenamiento (QAT – Quantization Aware Training). Se logró una precisión cercana al 88% con el conjunto de evacuación de CIFAR10.

Palabras clave: Aprendizaje profundo, Redes Neuronales de convolución, Cuantización binaria.

Abstract

In this work, it is proposed to implement a convolution Binarized Neural Network (BNN) to classify objects from RGB images. BNNs reduce the amount of computational and memory resources, and allow them to be inferred in embedded systems such as FPGAs, achieving real-time responses. The model is based on the VGG16 network and is trained on the CIFAR10 dataset. The network is binary quantized with the training-aware quantization technique (QAT). An accuracy close to 88% was achieved with the CIFAR10 evacuation set.

Keywords: Deep Learning, Convolutional Neural Network, Binary Quantization.

Introducción

Las redes neuronales de convolución (CNN – Convolutional Neural Network) son algoritmos de aprendizaje profundo (Aloysius & Geetha, 2018) los cuales tienen como entrada una imagen, y se les asignan pesos y bias a los aspectos u objetos de la imagen, de modo de poder diferenciarlos entre sí.

Este tipo de redes utilizan pesos y activaciones de punto flotante, generalmente 16 o 32 bits, por lo tanto, dependen de dispositivos capaces de operar con estas variables, ya sea para realizar entrenamiento como la inferencia. Se denomina inferencia al proceso por el cual los datos de entrada pasan por el modelo ya entrenado y se obtiene como salida las probabilidades de cada clase. Inferir estas redes basadas en variables de punto flotante, requieren de una gran cantidad de memoria y millones de operaciones. Es por este motivo, que para aquellas aplicaciones que se necesitan respuestas en tiempo real, es necesario utilizar placas gráficas (GPU- Graphic Processing Unit) que poseen una baja eficiencia energético.

En aplicaciones como la navegación autónoma, la robótica y la realidad aumentada, es necesario implementar inferencias en dispositivos capaces de tener un alto rendimiento a niveles de energía menores, y a un costo menor al de las GPU. Es por ello por lo que diversos investigadores han enfocado en los últimos años la implementación de inferencias redes neuronales en sistemas embebidos. Cómo es el caso de las FPGA (Field Programmable Gate Array), que son dispositivos que tienen un gran rendimiento en operaciones de bits y, por lo tanto, permiten realizar inferencias en tiempo real a un bajo costo energético (Shawahna et al., 2019). Para utilizarlas, es necesario cuantizar los pesos y las activaciones.

Este trabajo realiza las siguientes contribuciones:

- Implementa una red VGG16 (Simonyan & Zisserman, 2015) cuantizada de forma binaria.
- Obtiene con el modelo cuantizado una precisión comparable a la del modelo completo entrenando y evaluando con el dataset CIFAR10 (Liu & Deng, 2016).
- Utiliza activaciones bipolares, de modo que modelo pueda ser inferido en FPGA con compuertas OR exclusiva negativa (XNOR – Exclusive Negative OR).

Desarrollo

Dentro del estudio de redes neuronales cuantizadas se encuentra en evolución las redes binarizadas. En donde todas las operaciones de la red se resuelven con el cómputo de salidas que están restringidas únicamente a bits, como el caso de BinaryNet (Courbariaux et al., 2016). BinaryNet implementa una BNN, donde los pesos y las activaciones se restringen a -1 y 1, el objetivo primordial, es que sean simples de implementar en hardware con compuertas tipo XNOR. Utilizan una binarización determinística basada en la función sign que se describe según la Ecuación 1.

$$sign(x) = \begin{cases} +1 \; si \;\; x \geq 0 \\ -1 \; si \;\; x < 0 \end{cases} \tag{1}$$

Si bien, las activaciones y los pesos son binarios, los gradientes de valor real son acumulados en variables de valor real de alta resolución. Estos valores son utilizados por el gradiente descendiente estocástico, que explora el espacio de parámetros en pasos pequeños y ruidosos, este ruido se promedia por las contribuciones de los gradientes estocásticos acumulados en cada peso. El hecho de sumar ruido al cálculo de los gradientes no es más ni menos que una forma de regularización, por lo tanto, contribuye a una mejora de la generalización. Esto es similar a lo que

ocurre con Dropoput (Srivastava et al., 2014), aunque en lugar de colocar en cero la mitad de las activaciones de forma aleatoria, se binarizan los pesos y las activaciones.

Al derivar la función sign, se observa que en la mayoría de los casos es cero, por lo tanto, no es compatible con la retro propagación, en otras palabras, no tiene la capacidad de aprender. Para solucionar este inconveniente, los autores utilizan la estimación directa (STE - Straight-through gradient estimation), pero tomando en cuenta el efecto de la saturación. La estimación STE establece que los gradientes entrantes a la función sean igual a las salientes, sin tener en cuenta la derivada de la función. Courbariaux obtiene resultados competitivos con CNN y Redes completamente conectadas binarizadas y con capas de normalización de lote en datasets como MNIST, SVHN y CIFAR10.

En la actualidad el equipo de Xilinx ha desarrollado el repositorio BNN-PYNQ (Umuroglu et al., 2017), que es parte del proyecto FINN, está basado en BinaryNet, en la cual utilizan un modelo inspirado en VGG11, la cual es una variación del modelo VGG16 pero con 11 capas de convolución, al que denominan CNV, y obtienen una precisión en el dataset CIFAR10 del 84.22% con una cuantización de 1 bit de pesos y 1 bit de activación.

En este trabajo se utiliza como modelo a la variación de la red VGG16 propuesta por Lui (Liu & Deng, 2016), La cual consiste de pilas de capas formadas por convolución, activación, y pooling. Se utilizó el dataset CIFAR-10 (Krizhevsky, 2009) que consiste de 60 mil imágenes color (RGB) de 32x32 píxeles, con 6 mil imágenes por cada clase. Hay 50 mil imágenes para entrenamiento y 10 mil para evaluación, separadas en 10 clases.

Como se observa en la Figura 1 la primera pila de capas consta de 2 convoluciones de 64 filtros cada una, y un Maxpooling, que reduce el tamaño de la matriz a la mitad, utilizando un kernel de 2x2, y seleccionando el mayor elemento de esa porción de la matriz. La segunda, consta de 2 convoluciones de 128 filtros cada una con un Maxpooling a la salida. La tercera de 3 convoluciones de 256 filtros con Maxpooling. Y la cuarta y la quinta constan de 3 convoluciones de 512 filtros cada una, en este caso sin Maxpooling, ya que las dimensiones de las matrices no son divisibles por 2. Por último, se utilizan dos redes neuronales totalmente conectadas para relacionar los 512 elementos con 1024, y la segunda los 1024 elementos con las 10 clases de salida de la red. Luego se utiliza la función Softmax.

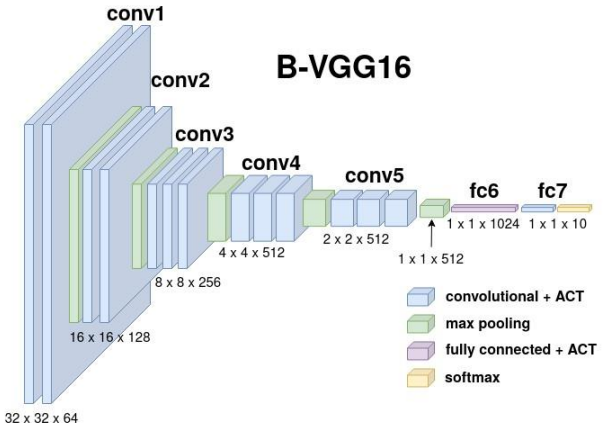

Figura 1: Arquitectura del modelo VGG16

Se utiliza una cuantización tipo QAT, se implementa la misma con la librería Brevitas (Pappalardo, 2021). Se utilizan pesos de 1 bits y activaciones de 1 bit. Se configuró la red con los parámetros e hiperparámetros que se observan en la Tabla 1.

Arquitectura	VGG16
Dataset	CIFAR10
Épocas	1000
Learning Rate (LR)	0.02
Tamaño del Lote	100
Función de perdidas	Cross Entropy
Optimizador	ADAM
Bits de Pesos	1 Bit
Bits de Activación	1 Bit

Tabla 1: Hiperparámetros de entrenamiento

Se realizó el entrenamiento en una computadora de escritorio con procesador I7 11700k, 16 GB DDR4-2666 MHz de Memoria, y una placa GPU NVIDIA GeForce RTX-3060. Se entrenaron 1000 épocas en casi 5 horas. Vale la pena aclarar, que, si bien se obtienen pesos y activaciones binarios, para realizar el entrenamiento se utilizan valores reales que luego son cuantizados en cada capa. En la Figura 2 se observa la evolución de la precisión de validación por épocas. A la derecha de la Figura 2 se visualiza la variación de las pérdidas de validación para cada época.

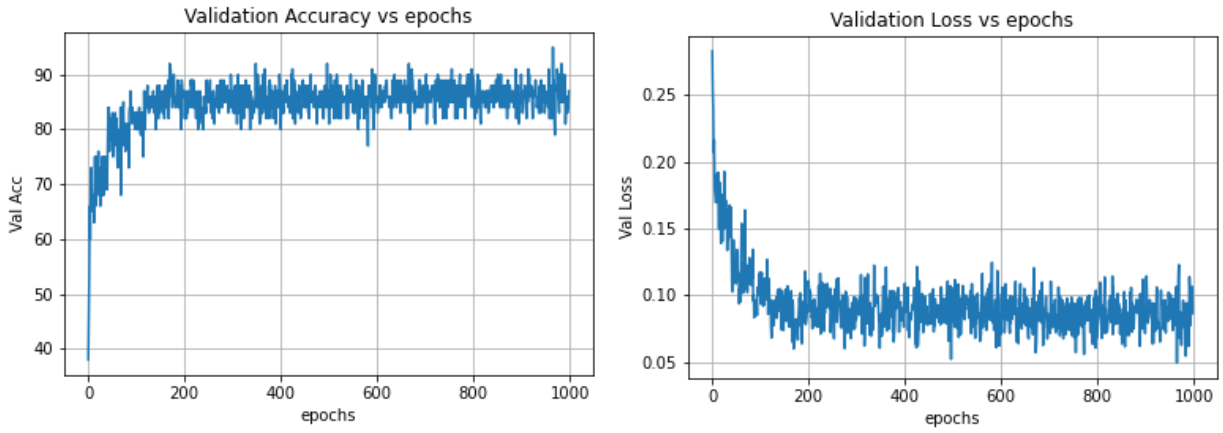

Figura 2: Precisión de Validación y Pérdidas de Validación vs épocas

Resultados

Con el modelo VGG16 propuesto se obtiene una precisión con el conjunto de evaluación de un 87.97%. La misma se calcula como el cociente de las predicciones correctas con el número total de predicciones. Se denomina a esta métrica TOP1. Además, se evaluó con la métrica TOP5, la cual, toma la predicción como correcta si la clase verdadera se encuentra dentro de las primeras 5 clases predichas. Se observan los resultados en la Tabla 2.

Test de Evaluación	*Precisión*
TOP1	87.97%
TOP5	98.92%

Tabla 2: Precisión TOP1 y TOP5

En la Figura 3 se observa una imagen de 32x32 píxeles RGB obtenida del conjunto de evaluación de CIFAR10. Luego de ser inferida por el modelo en análisis se observa a la derecha de la imagen las probabilidades predichas

para cada clase, luego de ser aplicada la función Softmax, quién se encarga de que la suma de todas las probabilidades sea igual a 1.

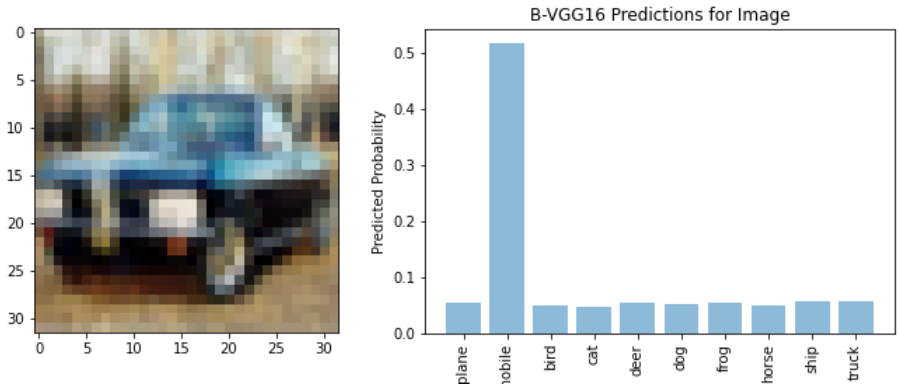

Figura 3: Imagen del dataset de evaluación de CIFAR10 y probabilidades de la clasificación

Además, de la evaluación con imágenes pertenecientes al dataset se evaluó con fotografías, como es el caso de la Figura 4, donde en primer lugar se recortó la imagen para centrar al vehículo, y luego se la redujo a 32x32 píxeles. Se pueden observar las predicciones en el gráfico que se encuentra a la derecha de la imagen.

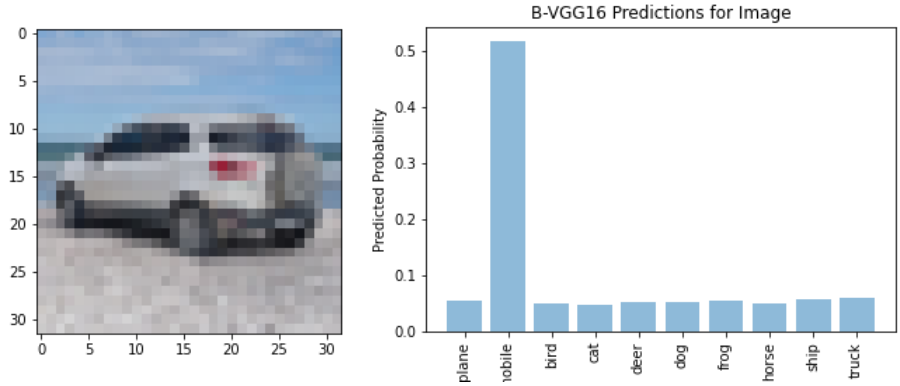

Figura 4: Imagen de vehículos y probabilidad de clasificación

Conclusiones

El modelo propuesto obtiene una mayor precisión de evaluación respecto al modelo CNV (Umuroglu et al., 2017), esto se debe a que se aumentan la cantidad de pilas de capas de convolución, activación y pooling. En la Tabla 3 se observa una mejora de casi 4 puntos porcentuales. Al utilizar un modelo más profundo, con mayor cantidad de activaciones y parámetros, también aumenta la cantidad de memoria en juego. Esto podría implicar un problema a la hora de implementar la inferencia de la red en sistemas tipo FPGA, pero el modelo CNV está optimizado para un chip de Xilinx ZYNQ 7020, que se encuentra dentro de la generación anterior de FPGA. Actualmente, la nueva generación de Xilinx Zynq Ultrascale+ es capaz de albergar dichos modelos.

Al cuantizar los pesos y las activaciones se pierde precisión, pero gracias a la cuantización tipo QAT, a medida que el modelo se entrena, estas pérdidas se compensan. Se puede ver que el modelo cayó poco más de 2 puntos en porcentaje en comparación con el modelo VGG16-BN-DROPOUT (Liu & Deng, 2016) entrenada con parámetros de FP32. Vale destacar, que el entrenamiento de estas redes tipo QAT, demora más tiempo que un entrenamiento con valores reales, aún entrenando en GPU, esto es debido al proceso de cuantización.

Por lo tanto, se puede concluir que con el modelo VGG16 binarizado propuesto se obtienen precisiones comparables con las del modelo VGG16 tradicional. Esto implica que es posible lograr resultados similares utilizando menos memoria, ya que, en comparación con las redes de punto flotante de 32 bits, requiere un tamaño en memoria

32 veces menor como se observa en la Tabla 3. También se reducen la cantidad de accesos en memoria, 32 tiempos menos de acceso a memoria. Respecto al modelo CNV, la mejora en la precisión de evaluación trae aparejado un mayor costo, debido a las pilas de capas que se agregaron.

Modelo	*Variables*	*TOP1*	Memoria
VGG16 (Liu & Deng, 2016)	FP32	~ 90%	466 MB
CNV (Umuroglu et al., 2017)	1 bit (Quant 1W1A)	84%	1,87 MB
VGG16 Binarizado (propio)	1 bit (Quant 1W1A)	87,97%	14,57 MB

Tabla 3: Comparación con otros modelos

El modelo planteado se puede implementar con una cantidad de hardware menor, ya que al estar los pesos y las activaciones binarizadas de forma bipolar, restriginados a +1 y -1, puede ser resuelto con compuertas XNOR en dispositivos tipo FPGA. A partir del framework experimental FINN (Blott et al., 2018), se implementará esta red en una FPGA de Xilinx Ultrascale+.

Referencias

Aloysius, N., & Geetha, M. (2018). A review on deep convolutional neural networks. *Proceedings of the 2017 IEEE International Conference on Communication and Signal Processing, ICCSP 2017*, *2018-Janua*, 588–592. https://doi.org/10.1109/ICCSP.2017.8286426

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

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

Krizhevsky, A. (2009). *Learning multiple layers of features from tiny images*.

Liu, S., & Deng, W. (2016). Very deep convolutional neural network based image classification using small training sample size. *Proceedings - 3rd IAPR Asian Conference on Pattern Recognition, ACPR 2015*, 730–734. https://doi.org/10.1109/ACPR.2015.7486599

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

Shawahna, A., Sait, S. M., & El-Maleh, A. (2019). FPGA-Based accelerators of deep learning networks for learning and classification: A review. *IEEE Access*, *7*, 7823–7859. https://doi.org/10.1109/ACCESS.2018.2890150

Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 1–14.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*, 1929–1958.

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