Propuesta de Investigación:
Machine Learning en dispositivos embebidos.

Francisco Ollero Pacheco
follep00@estudiantes.unileon.es

April 24, 2022

Resumen

El aprendizaje automático (ML) y el aprendizaje profundo (DL) se han convertido en una parte intrínseca de múltiples campos. La capacidad de resolver problemas complejos hace que el aprendizaje automático sea un panacea. En los últimos años, ha habido una explosión en la generación de datos, que ha mejorado mucho los modelos de aprendizaje automático.

Pero esto tiene un costo de alta computación, lo que invariablemente aumenta el uso de energía y el costo del hardware lo que habre nuevas posibilidades en dispositivos de borde y baja frecuencia.

Estos modelos, llamados tiny machine learning, o TinyML, son adecuados para dispositivos que tienen memoria y potencia de procesamiento limitadas, y en los que la conectividad a Internet no está presente o es limitada.

1 Introducción

¿Cúal es el problema?

El aprendizaje automático (ML) y el aprendizaje profundo (DL) se han convertido en una parte intrínseca de múltiples campos. La capacidad de resolver problemas complejos hace que el aprendizaje automático sea un panacea. En los últimos años, ha habido una explosión en la generación de datos, que ha mejorado mucho los modelos de aprendizaje automático.

Pero esto tiene un costo de alta computación, lo que invariablemente aumenta el uso de energía y el costo del hardware lo que habre nuevas posibilidades en dispositivos de borde y baja frecuencia.

Estos modelos, llamados tiny machine learning, o TinyML, son adecuados para dispositivos que tienen memoria y potencia de procesamiento limitadas, y en los que la conectividad a Internet no está presente o es limitada.

Una breve visión del ‘estado del arte’.

El aprendizaje automático es una tarea que requiere muchos recursos. Por lo tanto uno de los nuevos enfoques de optimización se basa en mejorar el rendimiento y el consumo de recursos computacionales y de energía. Aunque esta característica se acentúa más en sistemas TinyML es un reto para toda ML.

Por lo tanto la eficiencia energética es un reto muy popular en nuestros días.

¿Qué beneficios o logros genera estas investigaciones?

El echo de aumento de la actividad móvil y el avance en hardware más eficiente permite abordar una nueva experiencia de ML donde no sea necesario la conectividad con el cloud, lo que además proporciona un nuevo control sobre la privacidad. Y la posibilidad de incorporar dispositivos 'inteligentes' es entornos hasta ahora impensables. (entornos naturales, entornos rurales, medicina...)

¿Qué se va a presentar en este documento?

Se analizan los entornos TinyML y se realizan comparativas y evoluciones de los modelos. También se revisan las necesidades tecnológicas en los que pueden estar presentes y se explica el proceso completo hasta la puesta en marcha de estos sistemas.

2 Soluciones propuestas

Ente las soluciones propuestas y tenidas en cuenta sobre todo a partir del año 2017 nos encontramos modelos como MobileNet [2], CMSIS-NN [3], MNASNET [4], PROXYLESSNAS [5], ESPNETV2 [6] , MCUNET [1].

De estos tenemos que indicar que Mobilenet ha ido evolucionando desde las versiones v1 a v3 (2019), que MCUNET también en v2 (2021).

Network	Top 1	Params	MAdds	CPU
MobileNetV1	70.6	4.2M	575M	113ms
ShuffleNet (1.5)	71.5	**3.4M**	292M	-
ShuffleNet (x2)	73.7	5.4M	524M	-
NasNet-A	74.0	5.3M	564M	183ms
MobileNetV2	**72.0**	**3.4M**	**300M**	**75ms**
MobileNetV2 (1.4)	**74.7**	6.9M	585M	**143ms**

Nota: MobileNet en comparativa con otras redes.

3 Método seleccionado

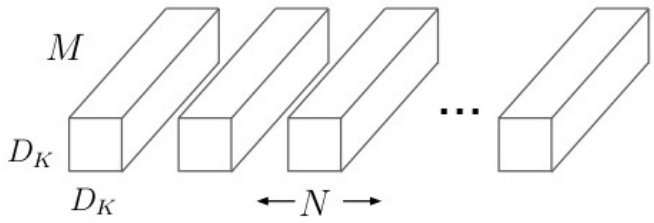
El modelo seleccionado es MobileNet que ha ido evolucionando desde su primera estructura hasta llegar a la versión3. La idea inicial está basada en la creación de una red neuronal convolucional eficiente y no muy intensiva en computación pare aplicaciones móviles.

Incluye la detección de objetos y clasificación.

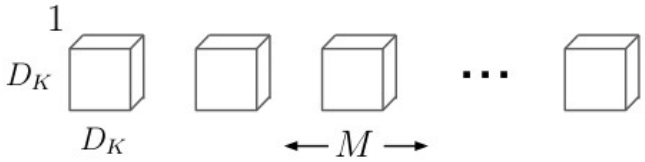
Las dos partes de la que consta son ‘Convolución separable en profundidad’ que básicamente son dos capas, la primera para filtrar los canales de entrada y la segunda para combinarlos.

La primera o ‘Convolución en profundidad’ se utiliza para aplicar un solo filtro en cada canal de entrada (novedad en comparación con la aplicación de filtro a todos los canales).

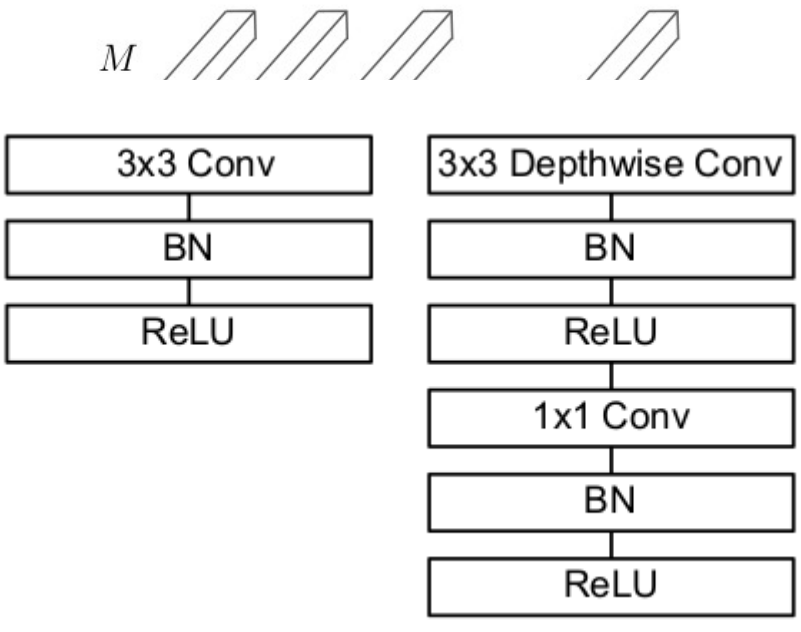
La segunda o ‘Convolución punto a punto’ que es una combinación lineal de la salida de la convolución profunda, usando filtros de 1 x 1.



(a) Standard Convolution Filters



(b) Depthwise Convolutional Filters



4 Datasets

Para la evaluación de clasificación utilizan ImageNet y muestran los siguientes resultados:

Table 8. MobileNet Comparison to Popular Models

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
GoogleNet	69.8%	1550	6.8
VGG 16	71.5%	15300	138

Table 9. Smaller MobileNet Comparison to Popular Models

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
0.50 MobileNet-160	60.2%	76	1.32
Squeezenet	57.5%	1700	1.25
AlexNet	57.2%	720	60

Y en detección lo hacen sobre COCO.

Framework Resolution	Model	mAP	Billion Mult-Adds	Million Parameters
SSD 300	deeplab-VGG	21.1%	34.9	33.1
	Inception V2	22.0%	3.8	13.7
	MobileNet	19.3%	1.2	6.8
Faster-RCNN 300	VGG	22.9%	64.3	138.5
	Inception V2	15.4%	118.2	13.3
	MobileNet	16.4%	25.2	6.1
Faster-RCNN 600	VGG	25.7%	149.6	138.5
	Inception V2	21.9%	129.6	13.3
	Mobilenet	19.8%	30.5	6.1

5 Metricas

Las métricas utilizadas es fundamentalmente ‘Accuracy’, pero debido a la peculiaridad de uso de estos modelos en dispositivos con menor capacidad de computo, involucran valores de procesamiento y tamaño del modelo.

Table 14. MobileNet Distilled from FaceNet

Model	1e-4 Accuracy	Million Mult-Adds	Million Parameters
FaceNet [25]	83%	1600	7.5
1.0 MobileNet-160	79.4%	286	4.9
1.0 MobileNet-128	78.3%	185	5.5
0.75 MobileNet-128	75.2%	166	3.4
0.75 MobileNet-128	72.5%	108	3.8

6 Implementación Python

Podemos encontrar varias fuentes de código relacionadas con este modelo, pero se la seleccionada contiene una generación del modelo (no uso directo del mismo) lo que ayuda a comprender sus componentes.

Además nos permite llevar la implementación hasta una ejecución práctica de lo obtenido (aunque sea a nivel de simulador).

https://github.com/PacktPublishing/TinyML-Cookbook/blob/main/Chapter07/ColabNotebooks/prepare_model.ipynb

Referencias

[1] Lin, J., Chen, W. M., Lin, Y., Gan, C., & Han, S. (2020). **Mcunet:** Tiny deep learning on iot devices. *Advances in Neural Information Processing Systems*, *33*, 11711-11722. ttps://arxiv.org/abs/2007.10319

[2] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... & Adam, H. (2017). **Mobilenets:** Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

[3] Lai, L., Suda, N., & Chandra, V. (2018). Cmsis-nn: Efficient neural network kernels for arm cortex-m cpus. *arXiv preprint arXiv:1801.06601*.(19/01/2018). https://arxiv.org/abs/1801.06601

[4] Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., & Le, Q. V. (2019). Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 2820-2828).https://arxiv.org/abs/1807.11626

[5] Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., & Le, Q. V. (2019). Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 2820-2828). Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., & Le, Q. V. (2019). Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 2820-2828).

[6] Mehta, S., Rastegari, M., Shapiro, L., & Hajishirzi, H. (2019). **Espnetv2**: A light-weight, power efficient, and general purpose convolutional neural network. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 9190-9200). https://arxiv.org/abs/1811.11431