針對 STM32 MCU 最佳化的 STM32Cube.AI library

作者：意法半導體

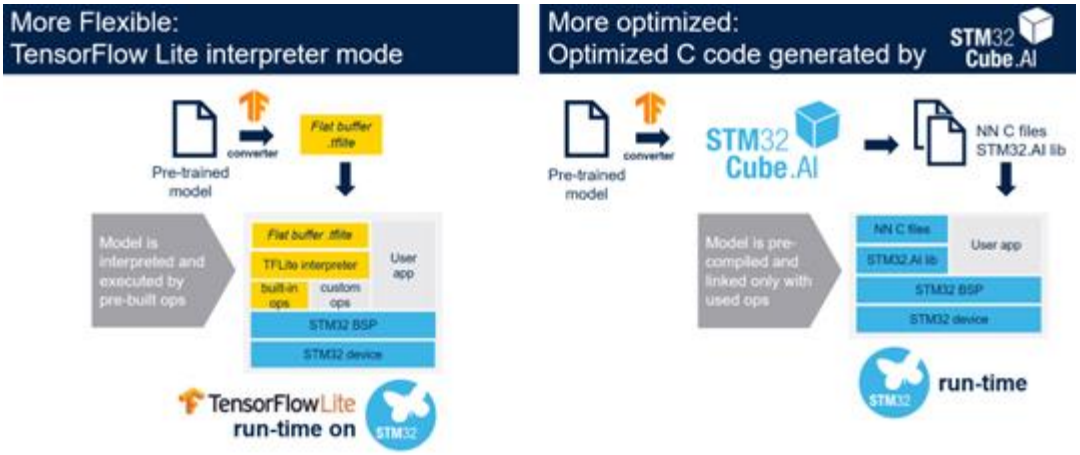
STM32Cube.AI 是意法半導體人工智慧生態系統的 STM32Cube 擴充套件，可以自動轉換預訓練之神經網路及將產生的最佳化函式庫整合到開發者專案中，以進一步擴充 STM32CubeMX 的能力。更提供多種方法在 PC 及 STM32 開發板驗證神經網路模型，並測量 STM32 裝置上的效能，無需使用者手動撰寫 C code。

本文將深入介紹 STM32Cube.AI 進階功能，涵蓋以下主題：

- Runtime 支援：Cube.AI 與 TensorFlow Lite
- 量化支援
- Graph flow 及記憶體設定最佳化
- Relocatable 二進位模型支援

Runtime 支援：Cube.AI 與 TensorFlow Lite

STM32Cube.AI 支援 2 種針對不同應用需求的 runtime——Cube.AI 和 TensorFlow Lite。Cube.AI 為預設執行階段，是針對 STM32 高度最佳化的機器學習函式庫。TensorFlow Lite for Microcontroller 由 Google 設計，用於微控制器及其他裝置上執行機器學習模型，只佔用幾千位元組的記憶體空間。它被廣泛用在基於 MCU 的應用。STM32Cube.AI 整合了一條特別路徑供 STM32 IDE 專案所用，並嵌入 TensorFlow Lite for Microcontrollers 執行階段 （TFLm）及其相關的 TFLite 模型。對於希望擁有一個跨專案通用框架的開發人員，這可視為預設的 Cube.AI runtime 替代方案。



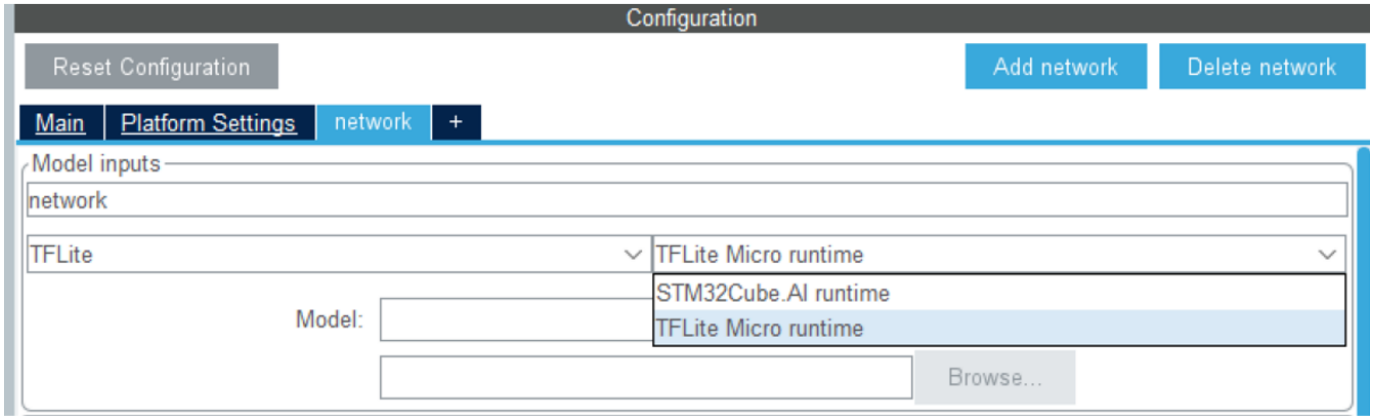
雖然兩種 runtime 均為資源有限的 MCU 專案而設計，但 Cube.AI 更針對 STM32 的獨特架構進一步最佳化。因此，TensorFlow Lite 更注重跨平台移植的需求，而 Cube.AI 則注重運算速度及記憶體佔用要求更高的應用。

下表顯示相同的預訓練神經網路模型中，兩種 runtime 之間的效能比較。

影像分類模型	STM32U585		STM32H723	
	Cube.AI 7.0	TFLu v0.5	Cube.AI 7.0	TFLu v0.5
FLASH	77 kB	96 kB	77 kB	96 kB
RAM	49 kB	53 kB	49 kB	53 kB
Inference Time	164 ms	315 ms	37 ms	61 ms

如表格所示，對於同一個模型，Cube.AI runtime 相較 TFLite runtime 節省了大約 20%的 Flash，和約 8%的 RAM，執行速度幾乎是 TFLite runtime 的 2 倍。

對於 TFLite 模型，使用者可在 STM32Cube.AI 的網路設定中選擇 2 種 runtime。



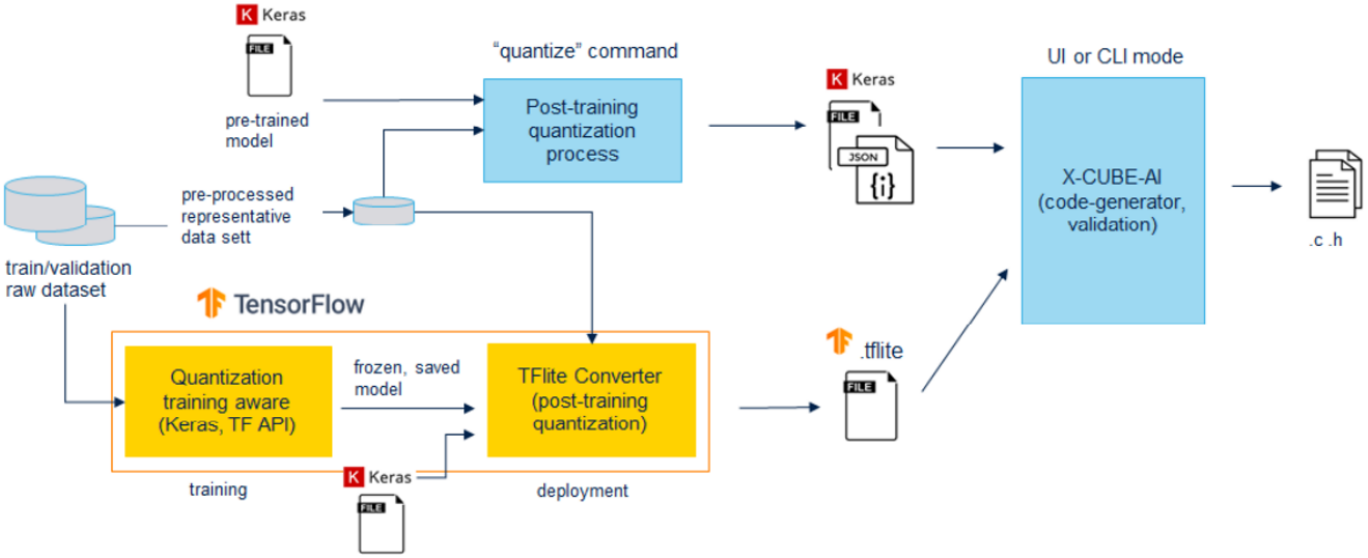
量化支援

量化是一種廣泛使用的最佳化技術，可將 32 位元的浮點數模型壓縮成更少位元的整數模型，進而減少儲存大小及 runtime 的記憶體佔用峰值，並改善 CPU／MCU 的推論時間及功耗，而精度只會略降一些。量化後的模型可利用整數（而非浮點值）來執行部分或是全部的 tensor。它是各種最佳化技術的重要組成部分，如：拓撲導向、feature map reduction、剪枝、weights compression 等，可應用於像是 MCU 這類 runtime 資源有限的環境。

另外兩種經典的量化方法：post-training quantization（PTQ）以及 quantization aware training（QAT）。PTQ 相對容易進行，它容許使用有限的代表性資料集對預訓練模型進行量化。而 QAT 則是在訓練過程中完成，模型精度通常更好。

STM32Cube.AI 以 2 種方式直接或間接支援這 2 種量化方法：

- 首先，可部署經由 PTQ 或是 QAT 量化的 TensorFlow Lite 模型。在此情況下，量化由 TensorFlow Lite 框架執行，主要是通過「TFLite converter」程式匯出 TensorFlow Lite 檔案。
- 其次，命令行介面（CLI）亦整合了一個內部 PTQ 程序，針對預訓練的 Keras 模型提供不同的量化計劃。與採用「TFLite converter」程式的訓練後量化相比，此內部程序提供更多量化計劃，在執行時間及精度方面提供不同的選擇。



下表顯示在 STM32 上部署量化模型相較原有浮點模型的優勢。FD-MobileNet 用作基準測試模型，12 層、145k 參數、24M MACC 運算及輸入維度 224x224x3。

KPI	Flash	RAM	Accuracy	Inference time
Float model	566 kB	844 kB	77.8%	420 ms
Quantized model Cube.AI 5.0.0	148 kB	212 kB	77.1%	153 ms
Save	~4x	~4x		~3x

從表中可見，量化模型可節省約 4 倍的 Flash 和 RAM，執行速度快約 3 倍，而精度僅降低 0.7%。

如果已安裝 X-Cube-AI 軟體套件，使用者可以在此路徑找到使用 Cube.AI CLI 進行量化的教學：C:\Users\username\STM32Cube\Repository\Packs\STMicroelectronics\X-CUBE-AI\7.1.0\Documentation\quantization.html. 文末亦附有「Quantize a MNIST model」範例。

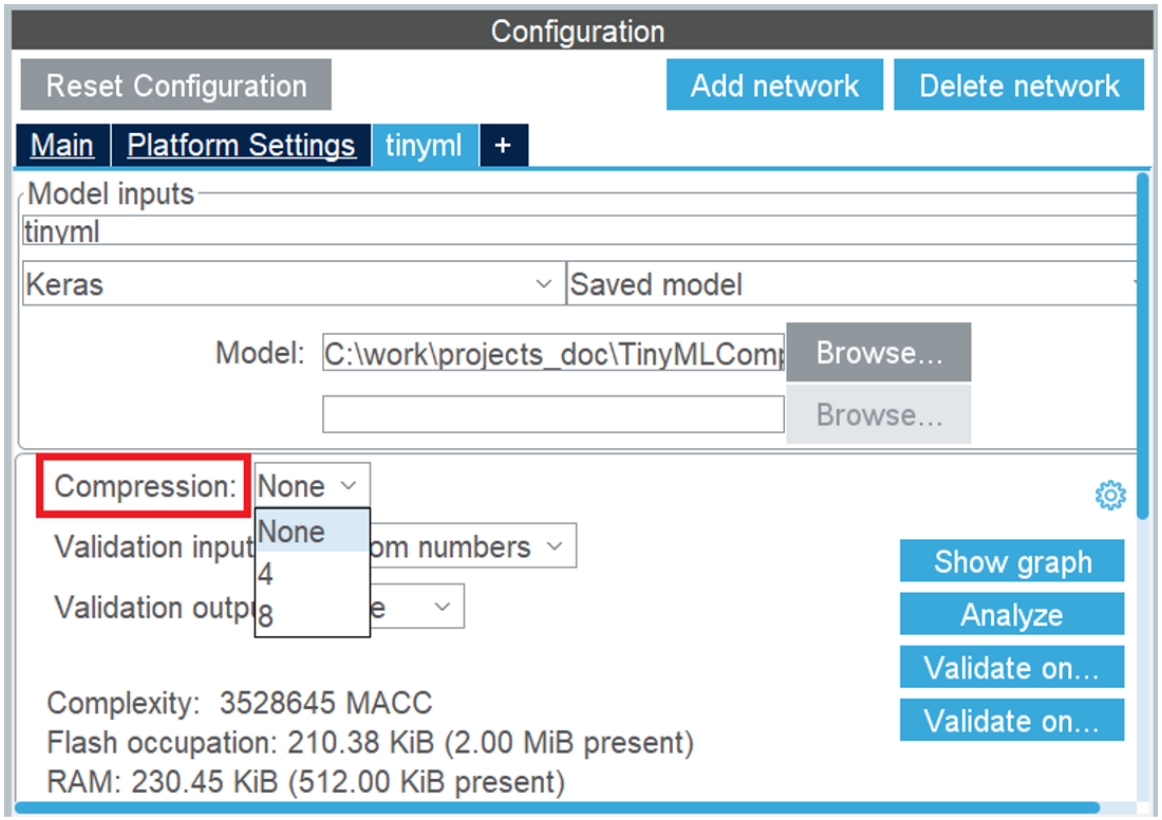
Graph flow及記憶體配置最佳化

除了量化技術外，STM32Cube.AI更追求使用其C Code產生器之最佳化引擎，針對推論時間最佳化記憶體使用（RAM和ROM）。它基於不需要資料集的方式，意味著壓縮及最佳化演算法無需訓練、驗證或測試資料集。

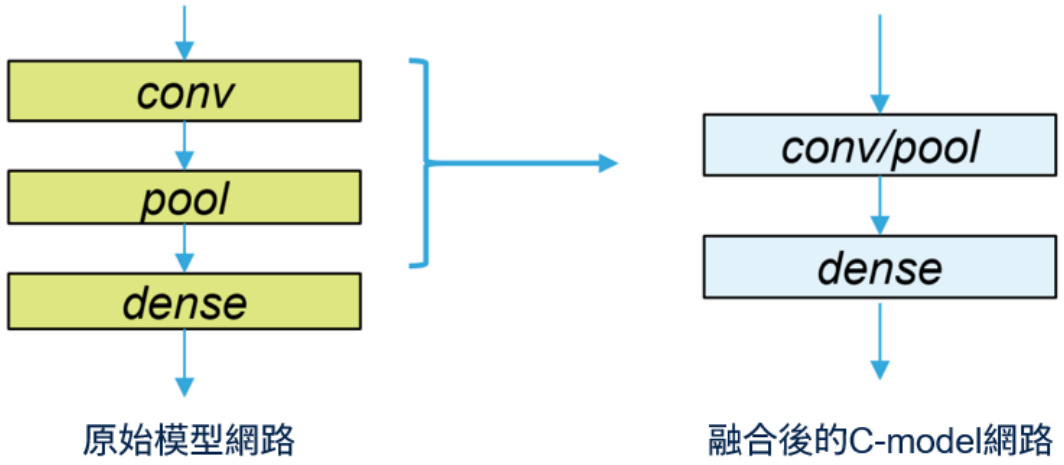
第一種方法是weigh／bias壓縮，當中應用 k-means 聚類演算法。此壓縮算法只適用於dense 層（完全連接層）。此方法的優勢是獲得快速的壓縮過程，但結果並非無損，全域精度可能會受影響。為此，

STM32Cube.AI提供了Validation process作為改善措施，以評估壓縮過程中所造成的誤差。

如下圖所示，壓縮功能可以在STM32Cube.AI網路設定中使用：



第二種方法是運算融合。將各層合併，再以最佳化資料的位置及相關的運算核心。有些層（例如「Dropout」、「Reshape」）在轉換或最佳化的過程中被移除，有些層（例如非線性及卷積層後的「Pooling」）則融入前一層。轉換後網路的層數通常低於原始網路，更減少記憶體中的資料傳輸量。

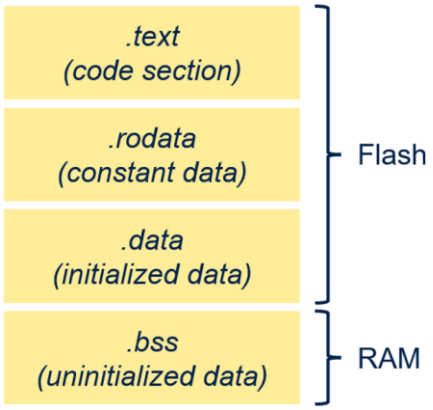


最後一種方法為最佳化activation memory。定義了一個 R／W 區塊，用於儲存暫時隱藏層的值（activation運算子的輸出）。它可被視為推論函數所用的scratch buffer。Activation memory可以在不

同層上被重複使用。因此，activation buffer的大小由兩個連續層的最大記憶體需求所定義。

Relocatable二進位模型支援

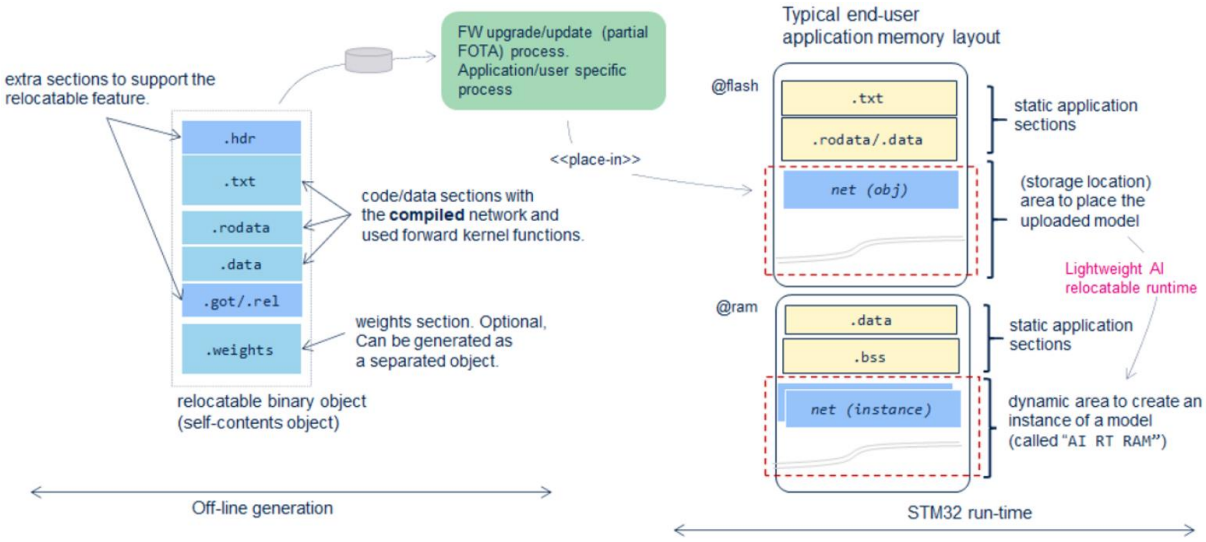
Non-relocatable方法（或所謂的「靜態」方法）指定神經網路的 C檔被編譯後，與終端使用者應用的stack連結在一起。如下圖所示，所有物件（包括神經網路及使用者應用）根據不同資料類型連結到不同的區段。在此情況下，當使用者希望更新函數的特定部分，整個韌體都需要更新。



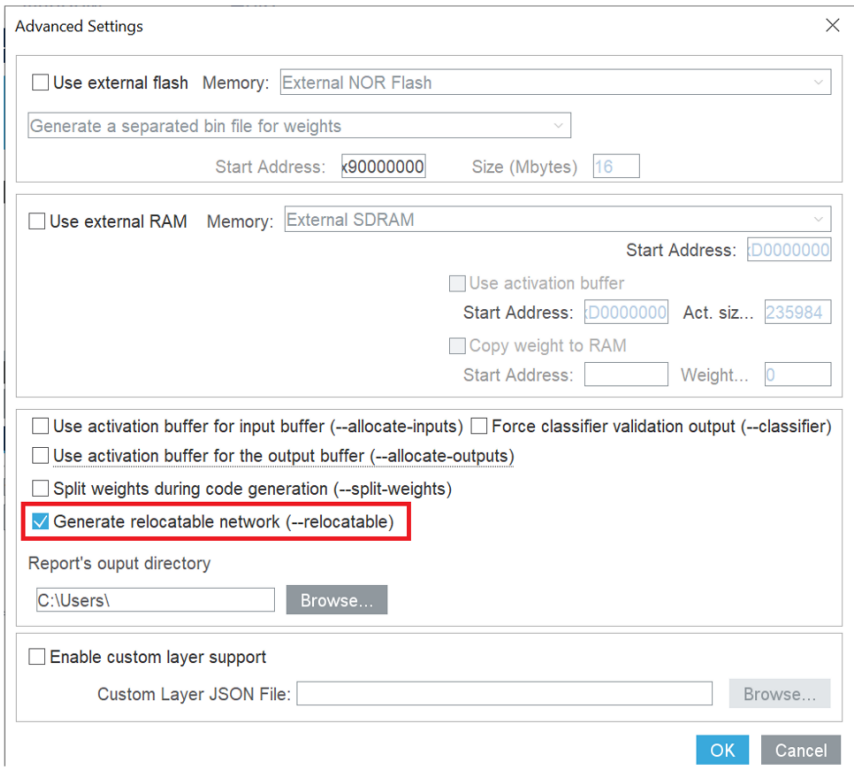
相對地，relocatable二進位模型指定一個二進位物件，該物件可在STM32記憶體子系統內任何地方安裝和執行。它包含被編譯後的神經網路 C檔，亦包括所需的forward kernel函式以及weights。主要目的是提供更靈活的更新 AI應用方式，即無需重新產生和燒錄整個韌體。

產生的二進位物件是一個輕量plug-in，可從任何位址執行（與位置無關的程式），其資料可儲存在記憶體的任何地方（與位置無關的資料）。利用STM32Cube.AI簡單高效的AI relocatable runtime，可將其實體化並加以利用。複雜及消耗資源的Arm® Cortex® -M MCU動態連結器不會嵌入STM32韌體，生成的物件是一個獨立實體，在runtime不需要外部symbols或函式。

下圖左側是神經網路的relocatable二進位物件，該物件是獨立的，並將放置於終端使用者應用的單獨區域（右側部分）。它可由STM32Cube.AI的relocatable runtime實體化及動態連結。因此，使用者在更改AI模型時，只需更新此部分的binary。也可以進一步選擇產生神經網路 weights作為一個獨立的物件，以增加靈活性。



如下圖所示，relocatable network可以在STM32Cube.AI的進階設定中啟用：



最後，作為 意法半導體AI生態系統的核心工具，STM32Cube.AI提供眾多基本或進階功能，幫助使用者輕鬆建立高度最佳化和靈活的AI應用。