

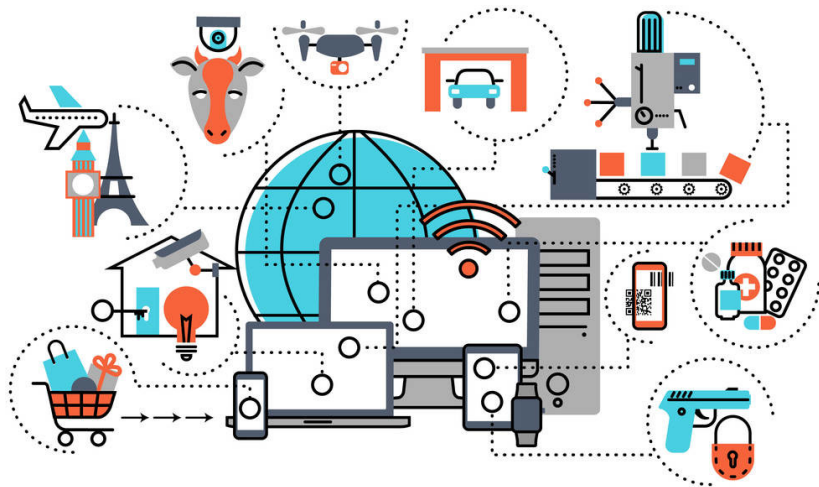
Apprentissage profond embarqué

Zhi YAN

ENSTA

12 février 2025

L'ère de l'intelligence artificielle



Qu'est-ce qu'un système embarqué ?

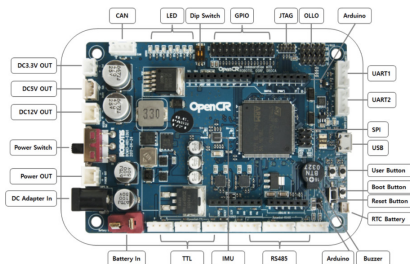
- ▶ Un système informatique à usage spécifique qui est “intégré” dans un autre appareil ou système.
- ▶ Optimisé pour des tâches spécifiques.
- ▶ Exemples courants : smartphones, montres intelligentes, appareils domestiques intelligents, véhicule autonome, robots, etc.



Qu'est-ce qu'un système embarqué ?

Particularités :

- ▶ **Contraintes de ressources** : mémoire (RAM), espace de stockage (ROM/Flash), puissance de traitement (CPU/MCU/GPU), consommation électrique, etc.
- ▶ **Exigences en temps réel** : contrôle industriel, électronique automobile, robotique mobile, etc.



Qu'est-ce qu'un système embarqué ?

Biais : Système embarqué = architecture ARM

Caractéristique	ARM	x86
Architecture du jeu d'instructions	RISC	CISC
Complexité des instructions	simple	complexe
Efficacité d'exécution	élevée	faible
Consommation d'énergie	faible	élevée
Domaines d'application	appareils mobiles, contrôle industriel, électronique automobile, etc.	contrôle industriel, équipements réseau, serveurs embarqués, etc.
...	...	...

Apprentissage profond embarqué

Quoi ?

- ▶ Apprentissage profond + Système embarqué

Pourquoi ?

- ▶ **Exigences en temps réel** : le traitement des données “hors-bord” peut entraîner une réponse retardée.
- ▶ **Connectivité réseau non garantie** : les appareils doivent fonctionner de manière indépendante.
- ▶ **Exigences en matière de protection de la confidentialité** : les données sensibles doivent être traitées localement pour protéger la vie privée des utilisateurs.

Apprentissage profond embarqué

Les enjeux (principalement causée par des **contraintes de ressources**) :

- ▶ **Déploiement de modèle** : Comment exécuter un modèle entraîné sur un serveur ou un cluster GPU sur un appareil embarqué ?
- ▶ **“Edge computing”** : Comment traiter et analyser les données à la source de leur génération ?
- ▶ **Mise à jour de modèle** : Un modèle peut-il être mis à jour en ligne pour s'adapter aux changements environnementaux ?
- ▶ **Sécurité et confidentialité des données** : Est-il nécessaire de stocker des données sensibles en périphérie ?

Apprentissage profond embarqué

Contexte :



Source de l'image : Microsoft (communiqué le 21 mai 2024)

- ▶ **Phi-3-mini-4k** : 4k tokens (environ 3000 mots)
- ▶ **Phi-3-mini-128k** : 128k tokens (environ 96 000 mots)
- ▶ **Phi-3-mini** : 3,8 milliards de paramètres * 4 octets = 15,2 milliards d'octets, soit environ 15,2 Go.

Apprentissage profond embarqué

Technologies clés :

- ▶ **Compression du modèle** (réduire la taille et la complexité de calcul du modèle) :
 - ▶ **“Pruning”** : Supprimer les connexions ou les neurones sans importance dans le réseau.
 - ▶ **“Quantization”** : Réduire la précision des paramètres du modèle (p. ex. de *float32* à *int8*).
 - ▶ **“Knowledge distillation”** : Utiliser un modèle plus grand (modèle enseignant) pour entraîner un modèle plus petit (modèle étudiant).

L'objectif de cette séance !

Apprentissage profond embarqué

Technologies clés :

► **Optimisation du modèle :**

- **Optimisation de la structure du réseau :** Concevoir une structure de réseau plus légère (par exemple MobileNet, ShuffleNet).
- **Optimisation des algorithmes :** Optimiser les algorithmes pour des plates-formes matérielles spécifiques.
- **Optimisation du code :** Utiliser des techniques de programmation et des bibliothèques efficaces.

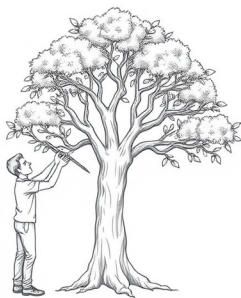
► **Accélération matérielle :**

- **Optimisation CPU :** Utiliser les jeux d'instructions SIMD, etc.
- **Accélération GPU :** Utiliser des GPU mobiles (par exemple ARM Mali), des GPU intégrés (par exemple Nvidia Jetson), etc.
- **Accélérateurs dédiés :** NPU (Neural PU), TPU (Tensor PU), etc.

Lectures étendues !

Pruning

- ▶ Un grand nombre de paramètres $\implies$ un modèle volumineux (espace de stockage, ressources de calcul).
- ▶ Paramètres redondants $\implies$ pas d'impact significatif sur les résultats de prédiction du modèle.
- ▶ Surparamétrage $\implies$ gaspillage de ressources de calcul, peut conduire à un surapprentissage.



Pruning

Types d'élagage :

- ▶ **Élagage des poids (Weight Pruning)** : Supprime les connexions dont les poids sont proches de zéro ou ont une faible valeur absolue.
- ▶ **Élagage des neurones (Neuron Pruning)** : Supprime les neurones jugés non importants.
- ▶ **Élagage des canaux (Channel Pruning)** : Spécifiquement pour les réseaux convolutionnels (CNN), qui supprime les canaux (ou filtres) non importants des couches de convolution.
- ▶ **Élagage des filtres (Filter Pruning)** : Similaire à l'élagage des canaux, mais supprime des filtres entiers (ensembles de noyaux de convolution).

Pruning

Étapes de l'élagage :

1. Entraînement du modèle original;
2. Évaluation de l'importance :
 - ▶ La magnitude des poids
 - ▶ Les statistiques des activations
 - ▶ Les métriques basées sur le gradient
 - ▶ etc.
3. Suppression des éléments non importants;
4. Réglage fin;
5. Répétition (les étapes 2 à 4).

Pruning

Stratégies d'élagage :

- ▶ **Élagage unique (One-shot Pruning)** : L'élagage est effectué une seule fois après l'entraînement initial.
- ▶ **Élagage itératif (Iterative Pruning)** : Les étapes d'élagage et de réglage fin alternent plusieurs fois.

La différence, en fait, est de savoir s'il faut ou non effectuer les étapes 4 et 5.

Pruning

Types d'implémentation :

- ▶ **Élagage non structuré (Unstructured Pruning)** : Supprime les connexions ou les neurones de manière aléatoire.
- ▶ **Élagage structuré (Structured Pruning)** : Supprime des canaux ou des filtres entiers.

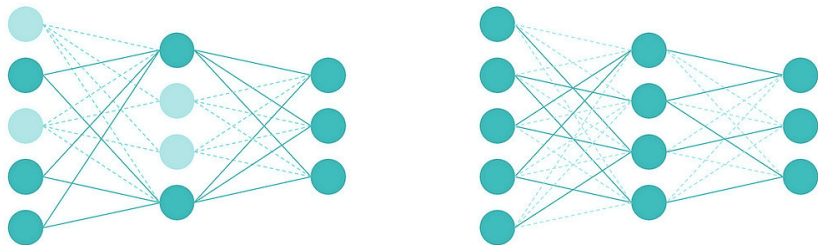


Figure: Structured pruning (left) vs. Unstructured pruning (right)

Pruning

Définition formelle de l'élagage :

Considérons un réseau neuronal profond représenté par une fonction $f_{\theta}(x)$, l'objectif de l'élagage est de trouver un sous-ensemble de paramètres θ' , où $\theta' \subseteq \theta$, de sorte que le modèle élagué $f_{\theta'}(x)$ fonctionne de manière proche du modèle d'origine $f_{\theta}(x)$ mais avec beaucoup moins de paramètres.

L'objectif de l'élagage comme un problème d'optimisation :

Minimiser : $|\theta'|$

Sous réserve de : $\mathcal{L}(\theta') \leq \mathcal{L}(\theta) + \varepsilon$

Question : ε ?

Élagage des poids :

Définir une matrice de masque binaire M de même dimensions que la matrice de poids W , où

$$\begin{cases} M_{ij} = 1, & \text{si le poids } W_{ij} \text{ est gardé} \\ M_{ij} = 0, & \text{s'il est élagué} \end{cases}$$

La matrice de poids élaguée W' peut être représentée par :

$$W' = W \odot M$$

Élagage des neurones/canaux/filtres :

De manière similaire, pour l'élagage des neurones/canaux/filtres de la couche l , un vecteur de masque m^l est défini, où

$$\begin{cases} m_i^l = 1, & \text{si le } i\text{-ème neurone/canal/filtre est gardé} \\ m_i^l = 0, & \text{s'il est élagué} \end{cases}$$

Les activations ou cartes de caractéristiques élaguées peuvent être représentées par :

$$a'' = a^l \odot m^l$$

Où a^l sont les activations ou cartes de caractéristiques de la couche l .

Une question jusqu'à présent : comment déterminer quels poids, neurones, canaux ou filtres doivent être élagués ?

Mesures pour l'estimation de l'importance :

- ▶ **Magnitude du poids** : $|W_{ij}|$
- ▶ **Régularisation L1** : $\lambda \sum_{ij} |W_{ij}|$
- ▶ **Régularisation L2** : $\lambda \sum_{ij} W_{ij}^2$
- ▶ **Mesures basées sur le gradient** $\partial \mathcal{L} / \partial W_{ij}$, matrice hessienne, etc.
- ▶ **Statistiques d'activation** : la moyenne, la variance, etc.

Pruning

Réécrivons les étapes d'élagage :

1. $\theta^\theta = \theta$ // Initialiser les paramètres
2. **Pour** $k = 1$ à K :
3. $M^k = \text{Prune}(\theta^{k-1})$ // Générer une matrice de masque basée sur une stratégie d'élagage
4. $\theta'^k = \theta^{k-1} \odot M^k$ // Appliquer l'élagage
5. $\theta^k = \text{FineTune}(\theta'^k)$ // Réglage fin
6. **Fin pour**

Quantization

Qu'est-ce que c'est ?

Utiliser des types de données de plus faible précision (par exemple, *int8*) au lieu de types de données de haute précision (par exemple, *float32*) pour représenter les poids et les valeurs d'activation dans les modèles.

Objectifs :

- ▶ Réduire la taille du modèle,
- ▶ Améliorer la vitesse d'inférence,
- ▶ Réduire la consommation d'énergie.

Quantization

Analogie :

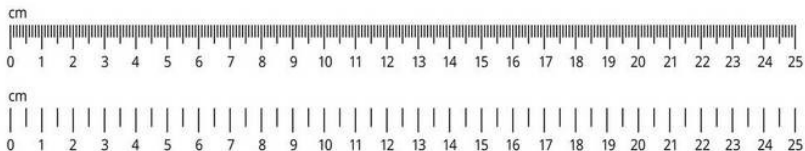


Figure: Mesurer une longueur avec une règle graduée en millimètres vs. en centimètres

Coût :

Sacrifier dans une certaine mesure la précision du modèle.

Quantization

Nombres à virgule flottante vs. Nombres à virgule fixe :

- ▶ 12.5 en *float32* : 01000001010010000000000000000000
- ▶ 5 en *int8* : 00000101

Caractéristique	float32	int8
Plage de valeurs	Très large	Limitée (-128 à 127)
Précision	Relative constante	Absolue constante
Complexité des calculs	Plus complexe et plus lent	Plus simple et plus rapide
Erreurs d'arrondi	Possibles	Généralement non (sauf overflow)
Utilisation	Calculs scientifiques, applications générales	Applications embarquées, optimisation

Quantization



Quantization

L'essence de la quantification :

“Mapper” les nombres à virgule flottante sur une plage de nombres entiers.

How-to :

- ▶ **Le facteur d'échelle (Scale)** : Mapper la plage de *float32* à la plage de *int8*.
p. ex. : $Scale = (max_float - min_float) / (max_int - min_int)$
- ▶ **Le point zéro (Zero Point)** : Représenter la position de la valeur zéro dans *float32* dans *int8*.
p. ex. : $ZeroPoint = round(min_float / Scale - min_int)$

Quantization

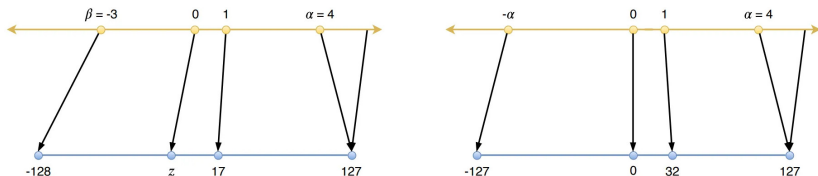


Figure: Affine (or asymmetric) quantization (left) vs. Symmetric quantization (right)

Quantification :

$$\text{integer_value} = \text{round}(\text{float_value} / \text{Scale} + \text{ZeroPoint})$$

Déquantification :

$$\text{float_value} = (\text{integer_value} - \text{ZeroPoint}) \times \text{Scale}$$

Quantization

Un exemple simple :

Supposons que nous ayons déterminé par des statistiques (ou d'autres méthodes) que la plage de valeurs de tous les poids dans un modèle est $[-5, 5]$, et qu'il existe une valeur de poids *float32* de 3,14, et que nous souhaitons la quantifier en *int8*.

Quantization

Un exemple simple :

Supposons que nous ayons déterminé par des statistiques (ou d'autres méthodes) que la plage de valeurs de tous les poids dans un modèle est $[-5, 5]$, et qu'il existe une valeur de poids *float32* de 3,14, et que nous souhaitons la quantifier en *int8*.

Solution :

- ▶ $Scale = (5 - (-5)) / (127 - (-128)) = 10 / 255 \approx 0.039$
- ▶ $ZeroPoint = \text{round}(-5 / 0.039 - (-128)) \approx \text{round}(-128.2 + 128) \approx 0$
- ▶ $integer_value = \text{round}(3.14 / 0.039 + 0) \approx \text{round}(80.5) = 81$
- ▶ $float_value = (81 - 0) \times 0.039 \approx 3.16$

Quantization

Un exemple simple :

Supposons que nous ayons déterminé par des statistiques (ou d'autres méthodes) que la plage de valeurs de tous les poids dans un modèle est $[-5, 5]$, et qu'il existe une valeur de poids *float32* de 3,14, et que nous souhaitons la quantifier en *int8*.

Solution :

- ▶ $Scale = (5 - (-5)) / (127 - (-128)) = 10 / 255 \approx 0.039$
- ▶ $ZeroPoint = \text{round}(-5 / 0.039 - (-128)) \approx \text{round}(-128.2 + 128) \approx 0$
- ▶ $integer_value = \text{round}(3.14 / 0.039 + 0) \approx \text{round}(80.5) = 81$
- ▶ $float_value = (81 - 0) \times 0.039 \approx 3.16$

Remarque :

Perte de précision : $3.14 \Rightarrow 3.16$

Quantization

Types de quantification :

- ▶ *Post-Training Quantization (PTQ)* : Quantification effectuée après l'entraînement du modèle, sans réentraînement.
 - ▶ **Avantages** : Simple et facile à utiliser.
 - ▶ **Inconvénients** : Perte de précision potentiellement importante.
- ▶ *Quantization-Aware Training (QAT)* : Simulation des opérations de quantification pendant l'entraînement du modèle.
 - ▶ **Avantages** : Perte de précision réduite.
 - ▶ **Inconvénients** : Nécessite un réentraînement du modèle, processus d'entraînement plus complexe.

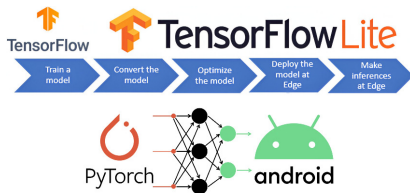
Quantization

Quantification en précision mixte :

- ▶ Utiliser différentes précisions de quantification pour différentes couches ou différents poids (p.ex. une précision plus élevée pour les couches importantes et une précision plus faible pour les couches moins importantes).

Outils de la quantification :

- ▶ TensorFlow Lite
- ▶ PyTorch Mobile
- ▶ NVIDIA TensorRT
- ▶ etc.



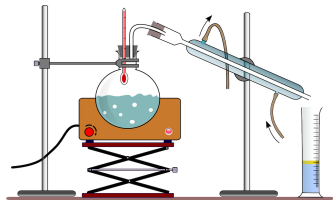
Knowledge distillation

L'idée :

Transférer la “connaissance” d'un modèle large et complexe (appelé “modèle enseignant”) à un modèle petit et simple (appelé “modèle étudiant”), afin que le modèle étudiant puisse être aussi proche que possible des performances du modèle enseignant tout en conservant une petite taille de modèle et une complexité de calcul réduite.

Le papier :

G. Hinton, O. Vinyals, J. Dean. “Distilling the knowledge in a neural network”. *arXiv preprint*, 2015.



Knowledge distillation



Figure: L'essence de la distillation des connaissances : transfert de connaissances

Knowledge distillation



Deux modèles :

- ▶ Modèle enseignant : entraîné en premier.
- ▶ Modèle étudiant : entraîné en second, sous la “direction” du modèle enseignant.

Les connaissances :

La distribution de probabilité de la sortie du modèle enseignant, les soi-disant “*soft targets*”.

Knowledge distillation



Figure: Hard label (or one-hot encoding) vs. Soft label

Les “*soft targets*” contiennent des informations plus riches, telles que la relation de similarité entre différentes catégories, ce qui aide le modèle étudiant à mieux apprendre !

Knowledge distillation

Une idée intuitive : “*soft targets*” par *softmax* :

$$q_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

Knowledge distillation

Une idée intuitive : “*soft targets*” par *softmax* :

$$q_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

Deux problèmes :

- ▶ Distributions de probabilité pointues.
- ▶ Risque de perte d'information des étiquettes négatives.

Knowledge distillation

Une idée intuitive : “*soft targets*” par *softmax* :

$$q_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

Deux problèmes :

- ▶ Distributions de probabilité pointues.
- ▶ Risque de perte d'information des étiquettes négatives.

Solution : **Coefficient de “Temperature”** :

$$q_i = \frac{\exp(z_i / T)}{\sum_j \exp(z_j / T)}$$

- ▶ T élevé : distribution plus aplatie et informative.
- ▶ T faible : rapprochement d'un encodage one-hot.

Knowledge distillation

Entraîner le modèle étudiant :

- ▶ Utiliser les mêmes données d'entraînement.
- ▶ Entraîner à la fois avec des “*hard labels*” et des “*soft targets*”.
- ▶ La fonction de perte :

$$Loss = \alpha \times HardTargetLoss + (1 - \alpha) \times SoftTargetLoss$$

- ▶ α : hyperparamètre, équilibrer les poids.
- ▶ *HardTargetLoss* : perte d'entropie croisée entre les prédictions du modèle étudiant et les “*hard labels*”.
- ▶ *SoftTargetLoss* : perte d'entropie croisée (ou divergence KL) entre les prédictions du modèle étudiant et les “*soft targets*”.



Knowledge distillation

- ▶ *Response-based Distillation* : Utiliser directement la sortie (*soft targets*) du modèle enseignant.
- ▶ *Feature-based Distillation* : Transférer la carte des caractéristiques de la couche intermédiaire du modèle enseignant.
- ▶ *Relation-based Distillation* : Se concentrer sur la relation entre les échantillons.

Pour en savoir plus : J. Gou, B. Yu, S. J. Maybank, D. Tao.
“Knowledge Distillation: A Survey”. *International Journal of Computer Vision*, 2021.

Fin