Blogpost

Hoe maak je AI-modellen klein en efficiënt?

AI-modellen worden steeds krachtiger, maar vaak ook steeds groter. Grote modellen vereisen veel rekenkracht, geheugen en energie - wat ze minder geschikt maakt voor gebruik op edge devices, maar ook realtime computervisie applicaties zoals de **Howest Virtual Mirror**. Gelukkig zijn er diverse technieken om AI-modellen te verkleinen zonder de accuraatheid ervan te verlagen. In deze blogpost bespreken we de belangrijkste optimalisatiestrategieën: **quantisatie**, **distillatie**, **pruning** en meer. We bekijken ook welke tools je kunt gebruiken, op welke platformen ze draaien, en hoe groot de impact kan zijn.

1. Quantisatie

Inleiding

Binnenin een AI model zitten er **miljoenen tot miljarden kommagetallen** - ook wel "*gewichten*" en "*activaties*" genoemd. Elk getal neemt typisch 32 bits (nullen en enen) geheugen in beslag.

Voor een computer zijn kommagetallen eigenlijk **zeer lastig** om mee te rekenen. Één zo'n berekening duurt uiteraard slechts enkele nanoseconden (of minder), maar door de enorme hoeveelheid ervan loopt deze vertraging zo hoog op dat het uiteindelijke model er **seconden tot minuten** over doet om data te verwerken.

Voorbeeld: elk sub-woord dat door GPT-4 genereert wordt moet door **1 tot 2 biljoen** parameters verwerkt worden. Beeld je maar in hoeveel berekeningen nodig zijn om een volledige tekst te genereren. Dit is de reden dat er voor AI zo'n grote nood is aan **krachtige GPU's**.

Bestaand model quantiseren

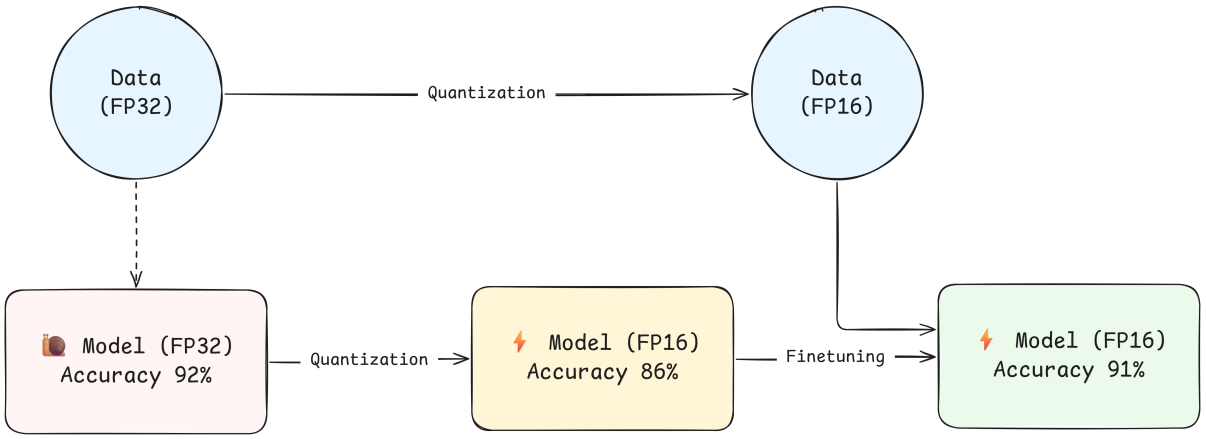
Post-training quantization (PTQ) is het proces waarbij de getallen in een reeds getraind model worden omgezet van 32-bit floating-point (FP32) naar een **lagere precisie**, zoals 16-bit (FP16) of zelfs 8-bit integers (INT8). Concreet heb

je dan **minder cijfers na de komma**, of zelfs helemaal geen kommagetallen meer.

Impact:

- Tot 4x verkleining van modelgrootte (van FP32 naar INT8).
- 2x tot 4x snelheidsverbetering.
- Vaak minimaal verlies in nauwkeurigheid (±5%).

Wat men vaak doet is het gequantiseerde model nog eens **kort hertrainen of finetunen** met gequantiseerde data. Dikwijls **verhoogt dit de accuraatheid** opnieuw naar het oorspronkelijke niveau van het niet-gequantiseerde model.



Gequantiseerd trainen

Een tweede mogelijkheid is **Quantized Aware Training (QAT)**. Hierbij traint men het model met reeds afgeronde getallen om het quantisatie-effect te simuleren. Typisch levert dit **nog betere resultaten** op dan PTQ, maar je moet het op voorhand in rekening brengen of je model volledig hertrainen.

Voorbeelden: op Ollama kan je van sommige **taalmodellen** - zoals de gemma3 familie - ook QAT-varianten terugvinden die tot 3 keer sneller zijn (https://ollama.com/library/gemma3:4b-it-qat). Ook van **computervisie** modellen zoals YOLO bestaan PTQ-varianten zoals YOLOv8-Detection-Quantized (https://huggingface.co/qualcomm/YOLOv8-Detection-Quantized).

Frameworks

Er bestaan verschillende Python-libraries die functies bevatten om jouw model te quantiseren:

- Gebruik de “tf.lite.Optimize” library voor **TensorFlow** modellen.

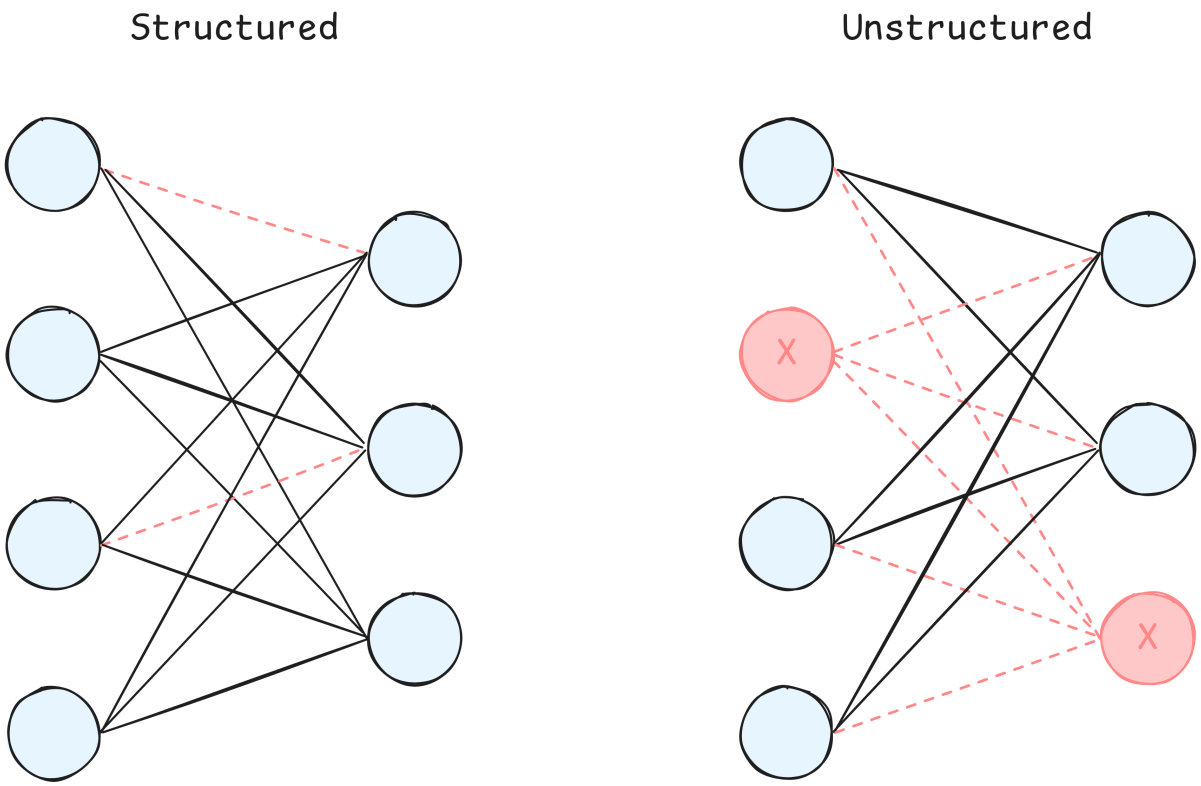
- Gebruik de “torch.ao.quantization” library voor **PyTorch** modellen. Dit is een onderdeel van de PyTorch Architecture Optimization library.
- Gebruik “onnxruntime.quantization” library voor modellen die geëxporteerd zijn als een gestandaardiseerd **ONNX formaat.**
- Ook Hugging Face’s **Optimum** library beschikt over quantisatie mogelijkheden voor ONNX modellen en Intel-systemen.

2. Pruning

Pruning **verwijdert elementen** uit het model die **weinig bijdragen** aan het uiteindelijke resultaat ervan. Het verminderen van overbodige parameters verkleint het aantal berekeningen die moeten gemaakt worden waardoor het model **sneller** zal zijn.

Er bestaan twee soorten pruning:

- **Niet-gestructureerde pruning:** verwijderen van individuele gewichten of verbindingen.
- **Gestructureerde pruning:** verwijderen van hele filters, neuronen of lagen.



Ook hier kan je het model **hertrainen of finetunen** na het prunen om het kleine verlies in accuraatheid te compenseren.

Frameworks

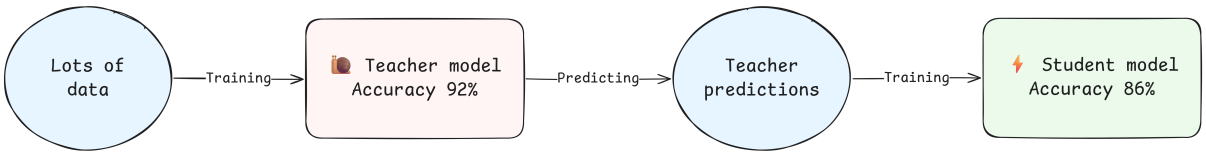
Er bestaan Python-libraries die functies bevatten om jouw model te prunen:

- Gebruik de “torch.nn.utils.prune” library voor **PyTorch** modellen.
- Gebruik de “tensorflow_model_optimization.sparsity.keras” library voor **TensorFlow** modellen.
- Ook Hugging Face’s Optimum library beschikt over pruning mogelijkheden voor Intel-systemen.

3. Kennisdistillatie

Kennisdistillatie is een techniek waarbij een klein model wordt getraind om de output van een groot, krachtig model te **imiteren**. Men beschrijft dit vaak als een **student-leerkracht** relatie.

Het leerkracht-model werd getraind door te leren uit grote hoeveelheden **data.** Het student-model daarentegen, wordt enkel getraind wordt op de **output** van een (reeds getraind) leerkracht-model en komt eigenlijk nooit in aanraking met de oorspronkelijke data.



In de praktijk blijkt dat student-modellen vaak **bijna dezelfde nauwkeurigheid** kunnen bereiken als het leerkracht-model terwijl ze **veel kleiner en sneller** zijn.

Voorbeeld: Het Chinese DeepSeek ontwikkelt gedistilleerde taalmodellen aan zoals DeepSeek-R1-Distill-Qwen-7B (https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Qwen-7B). Een ouder succesverhaal van distillatie is TinyBERT. Die is ruim 7 keer kleiner en 9 keer sneller dan het oorspronkelijke BERT terwijl het slecht een daling van ±3% heeft in nauwkeurigheid (https://arxiv.org/pdf/1909.10351).

Samenvatting

Optimalisatietechnieken kunnen een sterk voordeel bieden op vlak van snelheid, energieconsumptie, kostprijs, hardware beperkingen en speelt een grote rol in computervisie en taalmodellen. Het vergt wel enige tijd om mee te experimenteren en het verlies in accuraatheid te beoordelen en/of te compenseren. Deze extra ontwikkelingstijd kan afhankelijk van de schaal van het project wel of niet de moeite waard zijn. Of de implementatie van één of meerdere van deze technieken interessant is, moet dus voor elk scenario individueel beoordeeld worden.

Samenvatting

Optimalisatietechnieken zoals quantisatie, pruning en distillatie kunnen heel voordelig zijn op vlak van snelheid, energieverbruik, kostprijs en hardware-eisen. Zeker in toepassingen zoals computervisie en taalmodellen maken ze het verschil.

Het vergt wel enige tijd om mee te experimenteren en het verlies in accuraatheid te beoordelen en/of te compenseren. Deze extra ontwikkelingstijd kan afhankelijk van de schaal van het project al dan niet de moeite waard zijn.