

UMETNA INTELIGENCA NA ROBU (EDGE AI)

Z NVIDIA JETSON ORIN NANO



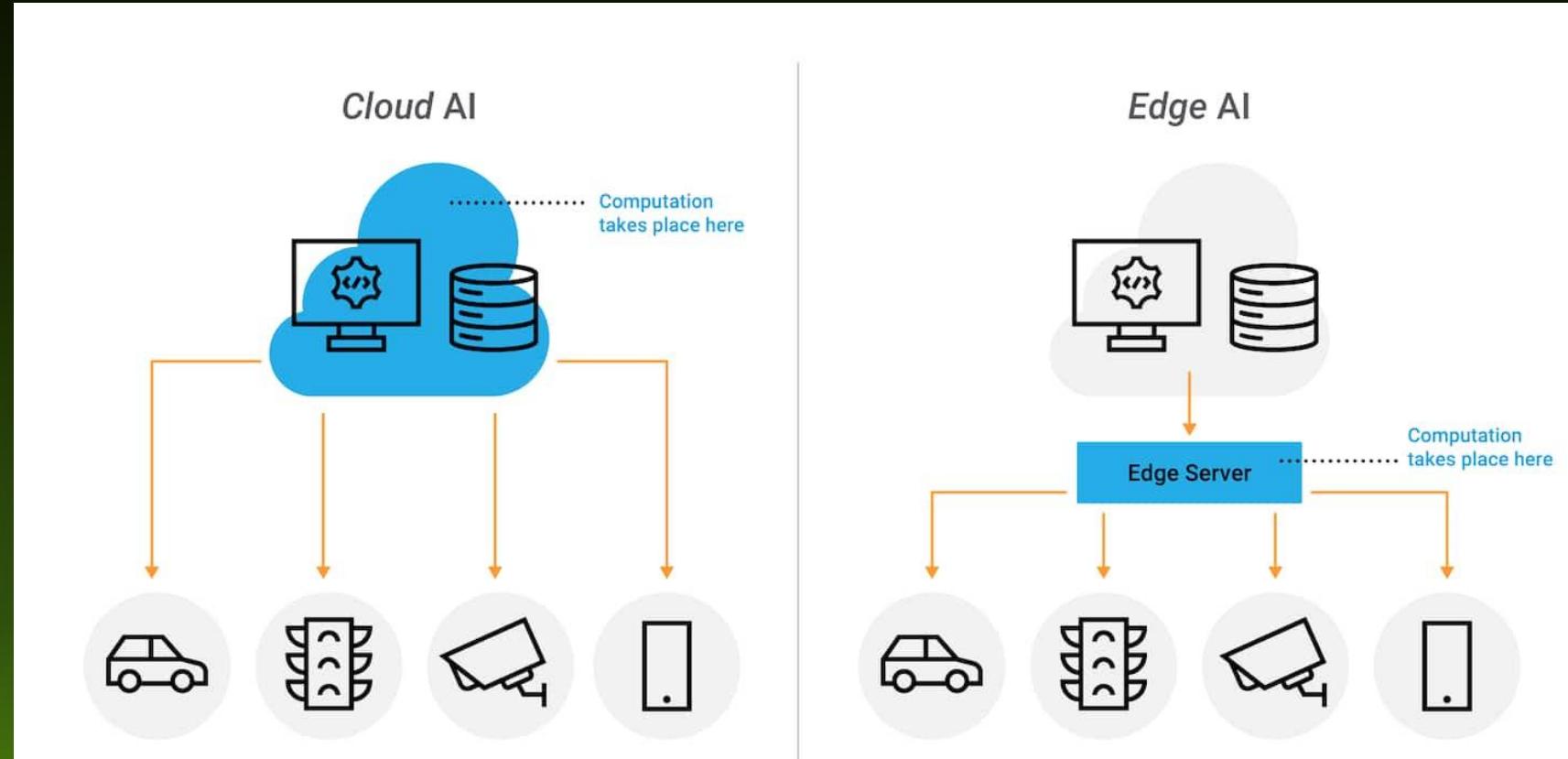
Kaj je Edge AI?

- ▶ **Umetna inteligenca na robu (Edge AI)** se nanaša na uporabo umetne inteligence v obliki algoritmov strojnega učenja (ML), ki se izvajajo in delujejo na avtonomnih napravah (na robu omrežja).
- ▶ **Strojno učenje (ML)** je široko področje, ki je v zadnjih letih doživelo izjemen napredok. Temelji na načelu, da lahko računalnik avtonomno izboljša svojo zmogljivost pri določeni nalogi z učenjem iz podatkov – včasih celo preko človeških zmožnosti.
- ▶ **Edge AI** lahko opravlja nešteto nalog, kot so zaznavanje predmetov, prepoznavanje govora, zaznavanje prstnih odtisov, odkrivanje goljufij, avtonomna vožnja itd.

Kaj je Edge AI?

Edge AI

pomeni izvajanje algoritmov in modelov umetne inteligence (AI) neposredno na lokalnih napravah ali strežnikih, ki so blizu viru podatkov – na robu omrežja namesto v centraliziranem oblaku.



Cilj je zagotoviti računalniško moč in inteligenco tam, kjer je potrebno sprejemati takojšnje odločitve.

Izzivi umetne inteligence na robu?

- ▶ Omejeni viri naprave. Potrebna je močna optimizacija AI modelov (kompresija).
- ▶ Razvojna orodja se pospešeno razvijajo, kar pomeni nestabilno razvojno okolje. Veliko sodelujočih, ogromno različic, knjižnic...
- ▶ Upravljanje in posodabljanje. Oddaljeno uvajanje in posodabljanje modelov na tisoče naprav (OTA - Over The Air).
- ▶ Zbiranje podatkov, učenje in ponovno učenje. Izbiranje in pošiljanje le najpomembnejših podatkov v oblak za učenje ali izboljšanje modela.
- ▶ Delovanje brez povezave.

Edge AI - Zakaj ravno zdaj?

Zrelost nevronskega mrež

Nevronske mreže in z njimi povezana infrastruktura umetne inteligence so se končno razvile do te mere, da omogočajo posplošeno strojno učenje. Organizacije se učijo, kako uspešno usposobiti modele umetne inteligence in jih uporabiti v produkciji na robu omrežja.

Napredek v računalniški infrastrukturi

Za delovanje umetne inteligence na robu omrežja je potrebna zmogljiva porazdeljena računalniška moč. Nedavni napredek na področju visoko vzporednih grafičnih procesorjev je bil prilagojen za izvajanje nevronskega mrež.

Uporaba IoT (interneta stvari)

Široka uporaba interneta stvari je spodbudila eksplozijo velikih količin podatkov. Z nenadno možnostjo zbiranja podatkov v vseh vidikih poslovanja – od **industrijskih senzorjev**, **pametnih kamer**, **robotov** in še več – imamo zdaj podatke in naprave, potrebne za uvedbo modelov umetne inteligence na robu omrežja. Poleg tega 5G zagotavlja zagon za internet stvari s hitrejšo, stabilnejšo in varnejšo povezljivostjo.

Področja uporabe

Avtonomna vozila

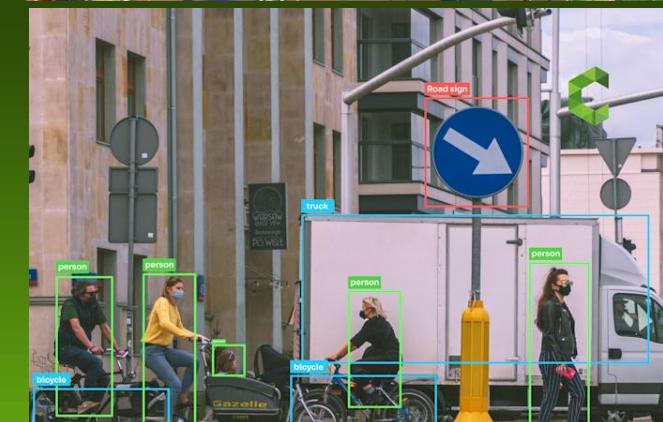
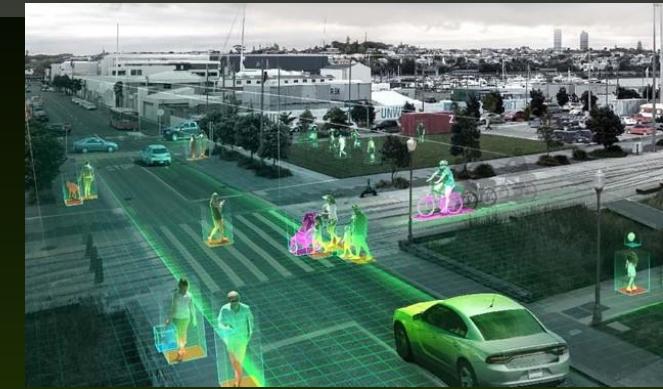
Vozila morajo v delčku sekunde obdelati podatke iz kamer, radarjev in LiDAR-jev, da lahko sprejmejo kritične odločitve (zaviranje, izogibanje oviram). Zanašanje na oddaljeni oblak bi bilo prepočasno.

Droni

Samostojno letenje, prepoznavanje objektov, izogibanje oviram, in izvajanje misij brez stalne povezave.

Računalniški Vid (Computer Vision)

- Prepoznavanje in sledenje objektom
- Pametne kamere za nadzor prometa in varnost
- Nadzor kakovosti v realnem času
- Avtomatski pregled izdelkov na proizvodni liniji
- Prepoznavanje obrazov/gest - Uporaba v varnostnih sistemih



Področja uporabe

Mali Jezikovni Modeli (SLM) na robu

- Lokalno prepoznavanje govora - pretvorba glasu v besedilo na sami napravi (za glasovne asistente) uporaba npr. v avtomobilih.
- Povzemanje in klepet (Chatbots).
- Filtriranje podatkov - obdelava in prepoznavanje ključnih informacij.

<https://blogs.nvidia.com/blog/cerence-generative-ai-in-car-experience/>



Robotika

- Avtonomna vozila: takojšnja obdelava podatkov iz vseh senzorjev za varno odločanje.
- Logistični in dostavni roboti: Lokalizacija, mapiranje in izogibanje oviram (SLAM).
- Droni: Samostojno letenje in izvajanje misij brez stalne povezave.

<https://blogs.nvidia.com/blog/roscn-2025-open-framework-robotics/>



Računalniški vid in analiza videa (Computer Vision)

Prepoznavanje objektov (Object Detection)

Identificiranje in lociranje več predmetov na sliki ali v video toku (npr. prepoznavanje avtomobilov, pešcev, posebnih delov na tekočem traku).

Segmentacija slik (Image Segmentation)

Razvrščanje vsakega piksla v sliki, kar omogoča zelo natančno določitev meja in oblik predmetov (npr. v avtonomnih vozilih, robotiki).

Sledenje objektom (Object Tracking)

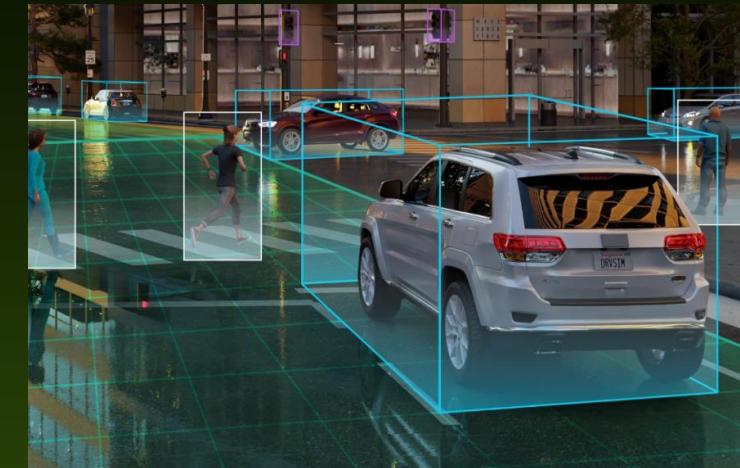
Spremljanje gibanja specifičnih objektov skozi čas v video posnetkih.

Analiza vedenja

Prepoznavanje nenavadnih aktivnosti ali dogodkov (npr. nadzor prometa, varnostni sistemi, prepoznavanje padcev).

Internet stvari (IoT) in pametne naprave

Pametni domovi (Siri ali Alexa), mesta (pametna parkirišča, semaforji, obdelava odpadkov), fitnes naprave, trackerji...



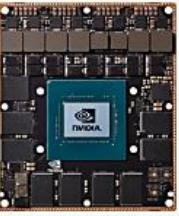
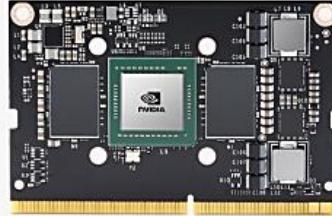
Prednosti umetne inteligence na robu

- ▶ Nizka latenca in odzivnost v realnem času ključno za kritične in časovno občutljive operacije.
- ▶ Izboljšana zasebnost in varnost občutljivi podatki ostanejo na lokalni napravi.
- ▶ Zanesljivost in delovanje brez povezave, tako je zagotovljeno avtonomno delovanje.
- ▶ Manjše potrebe po pasovne širini.
Ni potrebe po prenosu ogromnih količin podatkov.

NVIDIA Jetson Family

			
Jetson Thor series	Jetson AGX Orin series	Jetson Orin NX series	Jetson Orin Nano™ series
Up to 2070 TOPS	Up to 275 TOPS	Up to 100 TOPS	Up to 40 TOPS
40-130 W	15-60W	10-25W	7-15W
100 mm x 87 mm	100mm x 87mm	70mm x 45mm	70mm x 45mm
Coming Soon	Coming Soon	Coming Soon	Coming Soon

<https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-orin/nano-super-developer-kit/>

			
Jetson AGX Xavier™ series	Jetson Xavier NX series	Jetson TX2 series	Jetson Nano™
Up to 32 TOPS	21 TOPS	1.3 TFLOPS	0.5 TFLOPS
10-30W 20-40W	10-20W	7.5-15W 10-20W	5-10W
100mm x 87mm	70mm x 45mm	Starting at 70mm x 45mm	70mm x 45mm
		Starting at \$149 USD	\$99 USD

Koraki razvoja aplikacij AI

1. Nastavitev okolja (JetPack SDK)

Prvi korak je priprava same strojne opreme (Jetson Orin Nano) in namestitev potrebnega razvojnega okolja.

Namestitev JetPack SDK: To je temeljni programski paket podjetja NVIDIA, ki vključuje:

- ▶ **L4T (Linux for Tegra):** Optimiziran operacijski sistem (Ubuntu) za Jetson.
- ▶ **CUDA Toolkit:** Omogoča uporabo GPU-ja za splošne izračune.
- ▶ **cuDNN:** Knjižnica za pospeševanje nevronskih mrež (globoko učenje).
- ▶ **TensorRT in VPI:** Orodja za optimizacijo in inferenco modelov (glej korak 4).
- ▶ Konfiguracija osnovnega operacijskega sistema (Ubuntu Linux 22.04, swap particija...), nastavitev periferije (kamere, senzorjev: (GPIO, MIPI CSI-2), diska in drugih perifernih naprav, s katerimi bo aplikacija delovala.

Koraki razvoja aplikacij AI

2. Korak: pridobivanje podatkov in treniranje AI modela (oblak ali lokalno)

AI model se skoraj nikoli ne trenira neposredno na Jetson napravi, saj ta nima dovolj virov za hitro in učinkovito treniranje velikih modelov.

► Izbira modela

Izbira ustrezne arhitekture (npr. YOLO za prepoznavanje objektov, ResNet za klasifikacijo, manjši LLM za obdelavo jezika).

► Trening modela

Trening poteka na zmogljivih Cloud/PC GPU strežnikih z uporabo ogrodij, kot so PyTorch ali TensorFlow.

► Prenos in prilagoditev

Po treniranju se model shrani in prenese na Jetson napravo v enem od standardnih formatov (npr. ONNX, TensorRT).

Koraki razvoja aplikacij AI

3. Korak: optimizacija modela za Jetson (TensorRT)

Ta kora je zelo pomemben in ključen za uspeh projekta in tukaj se razlikuje razvoj aplikacij za Edge AI od Cloud AI, saj omogoča, da se model izvaja izjemno hitro in energetsko učinkovito na Jetson napravi.

► Pretvorba v TensorRT format

Uporaba orodja NVIDIA TensorRT, ki analizira model in ga optimizira za specifično strojno opremo.

► Kvantizacija (Quantization)

Zmanjšanje natančnosti številk znotraj modela (npr. iz 32-bitne plavajoče vejice na 8-bitno celo število - INT8 ali celo INT4). To bistveno poveča hitrost inference in zmanjša porabo moči, z minimalnim vplivom na natančnost.

► Obrezovanje modela (Model Pruning)

Odstranjevanje nepotrebnih povezav in uteži, kar dodatno zmanjša velikost modela in pohitri inferenco.

Koraki razvoja aplikacij AI

4. Korak: Razvoj Aplikacijskega Cevi (Inference Pipeline)

Po optimizaciji je treba model vključiti v delajočo aplikacijo, ki obdeluje podatke v realnem času. Večinoma si sledijo naslednji koraki:

► Pridobivanje (zajem) podatkov

Uporaba senzorjev ali video tokov (npr. OpenCV), ključne tu so knjižnice za zajem podatkov.

► Predprocesiranje

Priprava surovih podatkov za vnos v model (npr. spremenjanje velikosti slike, normalizacija).

► Inference (sklepanje)

Pomeni proces uporabe in izvajanja že natreniranega modela strojnega učenja na novih, še nevidenih podatkih za ustvarjanje napovedi, odločitev ali prepoznavanje vzorcev.

► Postprocesiranje in odločanje

Interpretacija rezultatov modela in sprožitev dejanske akcije (npr. "neznan objekt, ustavi robota," ali "zapiši lastnosti prepoznanega objekta v DB" ali "Delavec ne uporablja zaščitnih sredstev, alarm").

Koraki razvoja aplikacij AI

5. Korak: testiranje in uvajanje

Aplikacijo je treba temeljito testirati na sami napravi, kjer bo delovala. To testiranje vključuje:

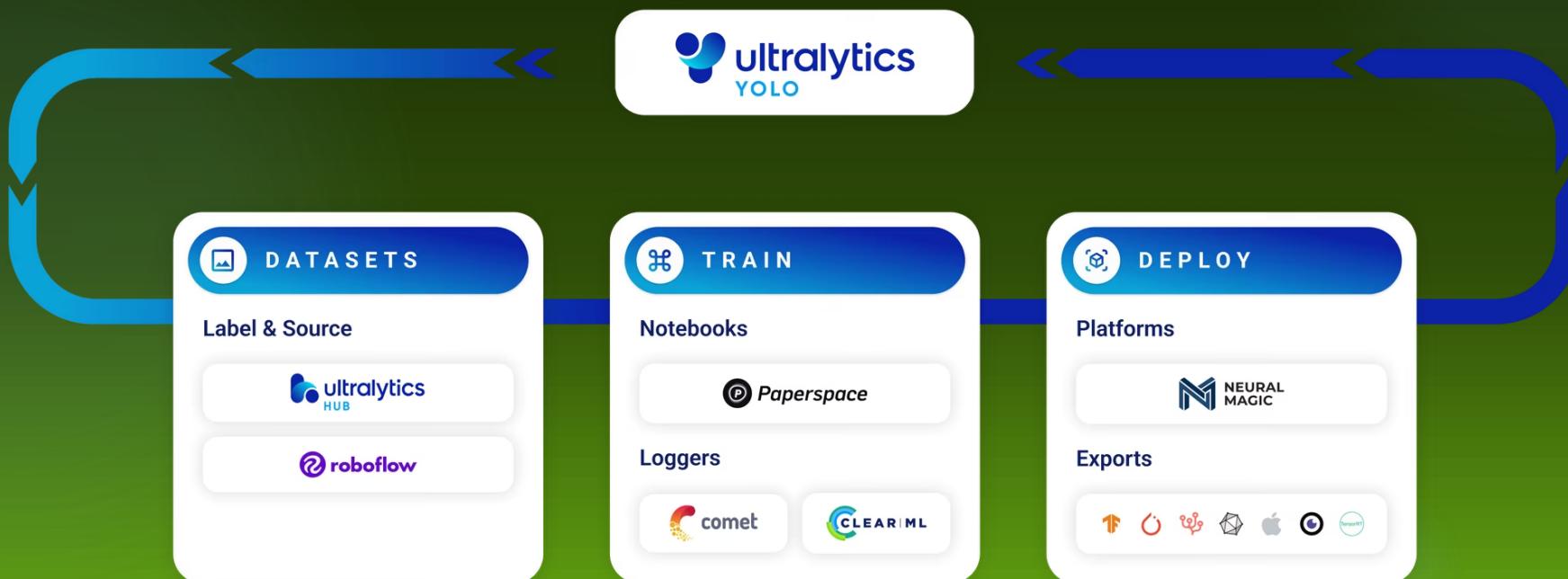
- ▶ **Testiranje v realnem času**
Preverjanje, ali aplikacija dosega zahtevano hitrost (FPS - sličice na sekundo) in nizko latenco.
- ▶ **Merjenje porabe virov**
Uporaba orodij za preverjanje obremenitve GPU/CPU in porabe pomnilnika (Memory Usage).
- ▶ **Energetska učinkovitost**
Pri napravah, ki delujejo na baterijo (kot so droni), je ključno preverjanje in optimizacija porabe energije.

Uvajanje in Posodabljanje (Deployment & OTA)

- ▶ Zadnji korak je uvedba aplikacije v produkcijo (Deployment). Namestitev aplikacije na končno napravo, pogosto v obliki Docker kontejnerjev za poenostavitev upravljanja odvisnosti.
- ▶ Upravljanje flote naprav. Pri razvoju za več naprav je potrebno centralno orodje (npr. NVIDIA Fleet Command) za daljinsko spremljanje in odpravljanje napak.
- ▶ Posodobitve (OTA - Over The Air). Priprava mehanizmov za oddaljeno posodabljanje programske opreme in AI modelov. To omogoča stalno izboljševanje modela, ko se v oblaku natrenira nova, natančnejša različica.

Treniranje modelov

- ▶ Učenje modela (Deep Learning) pomeni priprava, vnašanje podatkov in prilagajanje parametrov ter treniranje modelov, da lahko dajejo natančne odgovore.
- ▶ Na razpolago je veliko orodij za uporabo vnaprej treniranih modelov in pa seveda gradnjo svojih modelov. Pri tem je potrebno upoštevati sposobnosti strojne opreme, na kateri bo model deloval (omejitve HW pri AI na robu).



DEMO

Pripravil sem dve aplikaciji, ki prikažeta sposobnosti Jetson Orin Nano na področju Edge AI

- ▶ majhen jezikovni model (SLM)
Lama3
- ▶ računalniški vid
prepoznavanje in štetje objektov na sliki zajeti s kamere
- ▶ ...

Komentar k DEMO - Ollama LLM

Kateri model uporabiti? je ključno vprašanje pri Jetson napravah, kjer imamo omejen RAM in VRAM. Poglejmo razliko med temi tremi modeli: **Llama 3.2 1B**, **Llama 3.2 3B** in **Llama 3.1 8B**

Model	Parametri	Namen	Potrebni viri
Llama 3.2 1B	~1 milijarda	izjemno lahek model za robne naprave	nizek VRAM, nižja kvaliteta
Llama 3.2 3B	~3 milijarde	srednji model, ravnovesje med velikostjo in kvaliteto	srednja poraba VRAM
Llama 3.1 8B	~8 milijard	polni splošni model (najzmožljivejši)	visoka poraba VRAM

Oznaka "B" pomeni milijardo parametrov. Več parametrov = večji model = boljša natančnost / razumevanje, ampak tudi večja poraba pomnilnika in počasnejše delovanje.

Vpliv kvantizacije na kvaliteto LLM

Kaj pomeni "kvantizacija" v umetni inteligenci?

Kvantizacija pomeni, da številke, s katerimi model računa (teže in aktivacije), niso več shranjene kot 32-bitne ali 16-bitne vrednosti (float), ampak jih pretvorimo v števila z manjšim številom bitov, na primer 8, 6, 5 ali 4 bite.

Zakaj to naredimo?

Vsak parameter v modelu (npr. "utež") je število. Veliki modeli, kot je Llama, imajo na milijarde takih števil.

Oznaka	Bitov na parameter	Velikost modela	Hrast	Kvaliteta
FP16	16 bitov	največja	počasnejša	najbolj natančna
Q8_0	8 bitov	~50 % manjša	hitrejša	skoraj brez izgube
Q6_K	6 bitov	~60 % manjša	hitrejša	majhna izguba
Q5_K	5 bitov	~70 % manjša	hitra	dobra uravnoteženost
Q4_K/Q4_0	4 bitov	~75 % manjša	zelo hitra	nekaj izgube kakovosti
Q2_K	2 bitov	~90 % manjša	izredno hitra	velika izguba kakovosti

Če je model velik 7B (7 milijard parametrov), in je vsako število 16-bitno:

$$7 \text{ G} \times 2 \text{ B} = 14 \text{ GB} \text{ samo za uteži}$$

Če pa uporabimo kvantizacijo na 4 bite:

$$7 \text{ G} \times 0,5 \text{ B} = 3,5 \text{ GB}.$$

Torej model postane 4× manjši in porabi 4× manj pomnilnika.



Hvala