FAKULTET ELEKTROTEHNIKE, STROJARSTVA I BRODOGRADNJE SPLIT

Usporedba GPU u Google Colab

Napredne arhitekture računala - seminarski rad

Ivan Dagelić
Bruno Grbavac
Toni Jakovčević
Duje Srhoj
Andrej Vidović

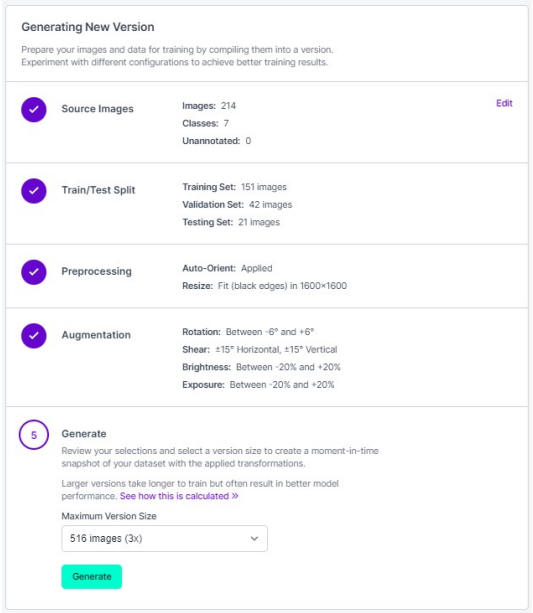
Računarstvo – diplomski studij (250)
Akademska godina 2021./22.

Sadržaj

1 Uvod

Google Colaboratory ili skraćeno Colab je produkt Google Research-a. Colab omogućava da bilo tko može pisati i izvršavati Python kod putem preglednika, te je posebno prikladan za strojno učenje i obradu podataka. Colab je baziran na *Jupyter notebook* servisu koji ne zahtjeva postavljanje prije korištenja, a istovremeno pruža besplatan pristup resursima kao primjerice GPU. U poglavlju 2 biti će objašnjeno kako su paralelizam te memorijski *bandwith* grafičkih kartica doveli do toga da je GPU prevladao nad CPU u korštenju za strojno i duboko učenje. Također u poglavlju 3 prikazana je optimizacija rada algoritama strojnog učenja na GPU. Zahvaljujući raznim metrikama koje danas postoje može se bolje iskoristiti vrijeme i resurse kako bi nam model postigao što bolje rezultate.

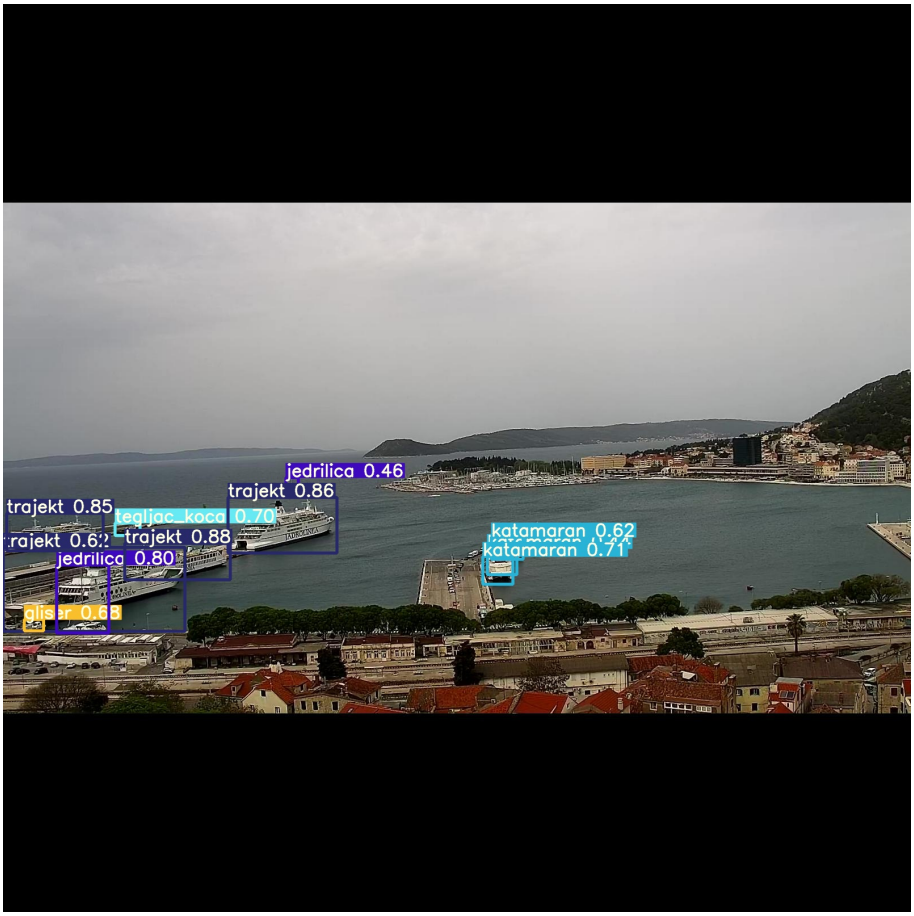
Vrsta GPU koja se može dobiti za rad i izvršavanje programa varira tijekom vremena jer Colab cijelo vrijeme pokušava osigurati besplatan pristup resursima za sve korisnike kako bi uspješno i što učinkovitije izvršili svoje programe. Najčešće vrste koje se mogu dobiti za rad i koje ćemo opisati i usporediti u seminaru su **Tesla T4**, **Tesla K80** te **Tesla P100**.
Projekt na kojem ćemo pokrenuti zadane grafičke kartice je model za nadzor luke temeljen na YOLOv5. Sustav za nadzor luke kao zadaću ima prepoznavanje i lociranje plovila unutar prostora luke. Koristeći ustaljenu terminologiju, sam model vrši zadaće **lokalizacije** (određivanja položaja) i **višeklasne klasifikacije** plovila.



Slika 1: Roboflow sučelje za predobradu.

Plovila se klasificiraju u sljedeće klase: **kruzer, trajekt, katamaran, tegljač/koća, jahta, jedrilica, gliser, jetski, glider** i **ostalo**.

Izvorni skup podataka (eng. *dataset*) sadrži **214 slika** Trajektne luke Split u formatu **1920x1080** dubine **24 bita**.



Slika 2: Primjer rada YOLOv5 modela

2 Uporaba GPU za strojno učenje

Primarni razlog zašto su grafičke kartice postale centralni hardver za primjenu u strojnom učenju je **paralelizam**. Naime, iako su moderni procesori vrlo dobri u paralelnom izvršavanju **svestranih** zadataka, poput rukovođenja operacijskim sustavom, kompresije i dekompresije datoteka itd., njihova se sposobnost u **specifičnim i jednostavnim** operacijama ne može mjeriti sa rezultatima GPU-ova.

Dok se broj jezgri, iako mnogo većih i sposobnijih, kod CPU-a jedva doseže dvoznamenakste brojeve kod **user spec** procesora, te nekoliko desetaka kod **procesora za profesionalnu upotrebu** (na serverima). Grafički procesori, s druge strane, broj doduše manjih jezgara, namijenjenih specifičnim operacijama, broje u tisućama. I jedan i drugi tip procesora oslanjaju se na **cache memoriju** vezanu uz jezgru, po principu **locality reference**. Pogledamo li **umjetne neuralne mreže**, mnogi izvori ih nazivaju ***sramotno paralelnima***. Naime, većina operacija, poput **unatražne propagacije** i proračuna **aktivacijskih funkcija** potpuno su nezavisne pa se mogu paralelizirati. **Konvolucijske mreže** također su veoma pogodne za paralelizam. Naime, algoritmi poput ***sliding window-a*** kod izrade ***feature mapa*** mogu se u potpunosti paralelizirati na raziini izlaznog piksela mape.

Osim samog paralelizma, za uspjeh grafičkih procesora u operacijama poput matričnog množenja zaslužan je i veći **memorijski bandwith**. Centralne procesorske jedinice optimizirane su s ciljem najmanjeg mogućeg kašnjenja, dok su grafički procesori usredotočeni na bandwidth. Stoga će učitavanje podataka iz ravne memorije zahtjevati kraće vrijeme na CPU, no količina podataka koja će se dohvaćati od jednom biti će podosta manja od one kod GPU. Ponajbolje procesorske jedinice rade s bandwithom od 50GB/s dok GPU-ovi dosežu veličine poput 750GB/s. Iz ovoga možemo zaključiti da su GPU-ovi jako prikladni za **podatkovno intenzivne** operacije. Iako je načelno dohvat na GPU sporiji i uzrokuje veće kašnjenje, takvo ponašanje se gotovo anulira kod podatkovno zahtjevnih operacija (poput dubokog učenja), parlelizmom na razini jezgre. Naime, dohvat prvog "paketa" uzrokovat će određeno čekanje, dok sljedeći paketi neće pridonjeti kašnjenju zbog čekanja izvršenja rada nad prethodnim paketom podataka. Osim brzine i bandwidtha dohvata iz memorije, grafički procesori nude poboljšanje i kada su u pitanju L1 cache registri. Optimalni oblik ovakvih registara je da su blizu samoj ALU jedinici, te da su relativno mali, kako bi se smanjilo vrijeme pretrage/pristupa. S obzirom da CPU jezgre zahtjevaju rad s operacijama široke namjene, takvi registri podosta su veći od onih kod GPU. Grafički procesori imaju veliki broj malih L1 registara, vezanih uz odgovarajuće ***stream procesore***. Ovakva arhitektura dovodi do ukupno 30 puta većeg L1 memorijskog kapaciteta uz dvostruko veću brzinu pristupa. GPU registri dohvaćaju brzine od oko 80TB/s, s ukupnom veličinom registara od 14MB dok CPU raspolaže s oko 1MB L1 cachea pri 5 TB/s.

2.1 NVIDIA i strojno učenje

CUDA je predstavljena 2007. godine i alat je za konfiguraciju paralelizma na grafičkim karticama na optimiziran način, iskoristivši pritom maksimalne performanse iste. ***Compute Unified Device Architecture*** toolkit uključuje okruženje sa C/C++ kompajlerom, debuggerom i bibliotekama te pripadajućim driverima za komunikaciju sa GPU.

Veliki iskorak za strojno učenje dogodio se 2018. Nvidijinim predstavljanjem grafičkih kartica serije 2000, prve serije RTX kartica. Ove kartice predstavile su **tensor jezgre** posvećene dubokom učenju, temeljene na **Volta arhitekturi**. Duboko učenje koristi intenzivan matrični račun, stoga su ove kartice ponudile znatan napredak u tom području. Naime, **tensor jezgre** izvršavaju matrično množenje 4x4 FP16 matrica i zbrajanje 4x4 FP16 ili FP32 matrica s polovičnom preciznošću pri čemu je izlaz 4x4 FP16 ili FP32 matrica s potpunom preciznošću. Ove jezgre donijele su ogromno ubrzanje u odnosu na jezgre sa **Pascal** arhitekturom.

2020. godine Nvidia je predstavila grafičke kartice serije 3000 sa **Ampere** arhitekturom koje su praktički udvostručile rezultate dotadašnjih tensor jezgara. Također su uvedene nove vrijednosti za preciznost poput TF32 i FP64. TF32 funkcionira na jednak način kao dotadašnji FP32, no pritom nudivši 20x ubrzanje, pri čemu su iz Nvidije najavili smanjenje vremena treniranja i zaključivanja sa tjedana na nekoliko sati.
Posvećenost NVIDIA-e strojnom učenju vidi se i u implementaciji **cuDNN**-a, biblioteke za umjetne neuralne mreže optimizirane za rad na Nvidijinim grafičkim karticama. Implementirane su svi temeljni algoritmi neuralnih mreža poput, aktivacijskh funkcija, unaprijedne i unatražne propagacije te poolinga.

3 Optimizacija rada algoritama strojnog učenja na GPU

Kako bi kvalitetno iskoristili utrošeno vrijeme i resurse, prate se najčešće neke od sljedećih metrika.

- **Iskorištenost GPU** nam govori koliki je udio grafičkih jezgri pokrenut. Mali iznos ove metrike može ukazati na nepotrebnost korištenja jakih kartica, no najčešće je uzrok loša distribucija workloada. Visoki iznos ove metrike s druge strane, pokazuje dobar raspored workloada, a u krajnjim situacijama i nedostatnost resursa na korištenoj kartici.

- **Pristupanje i iskorištenost memorije GPU** mjeri aktivnost memorijskog kontrolera kartice. Bilježe se svi pristupi memoriji, a metrika se najčešće pregledava tokom postavljanja veličine jednog ***batcha*** u procesu treniranja modela. Kao i ostale metrike, lako je dostupna kroz NVIDIA-smi.

- **Time to solution** metrika donosi vrijeme potrebno da model pri treniranju postigne određenu preciznost (ili drugu metriku) na validacijskom setu. Ova metrika naravno, osim o postavkama samog treniranja i arhitekturi modela ovisi o grafičkoj kartici.

Iako je generalna zamisao većine ljudi koji su se susreli sa strojnim učenjem da su grafičke kartice jednostavan ispravan izbor za sve operacije vezane uz model, to nije istina. GPU je pretežno dominantan uz operacije odnosno instrukcije temeljene na kompleksnom matričnom računu poput raznih **konvolucija**. S druge strane CPU će, usprkos mišljenju većine, donijeti znatno poboljšanje u operacijama poput **konkatenacije**, **Non Maxima Supression** i općenito operacijama temeljenim na kontroli toka (if, while, where). Promjena uređaja na kojem se operacije izvode stoga može donijeti značajna poboljšanja u vremenima treniranja ili zaključivanja. U navedenom članku, SSD MobileNet V2 model za detekciju pasa, postigao je poboljšanje u vremenu zaključivanja sa 50ms na 30ms.

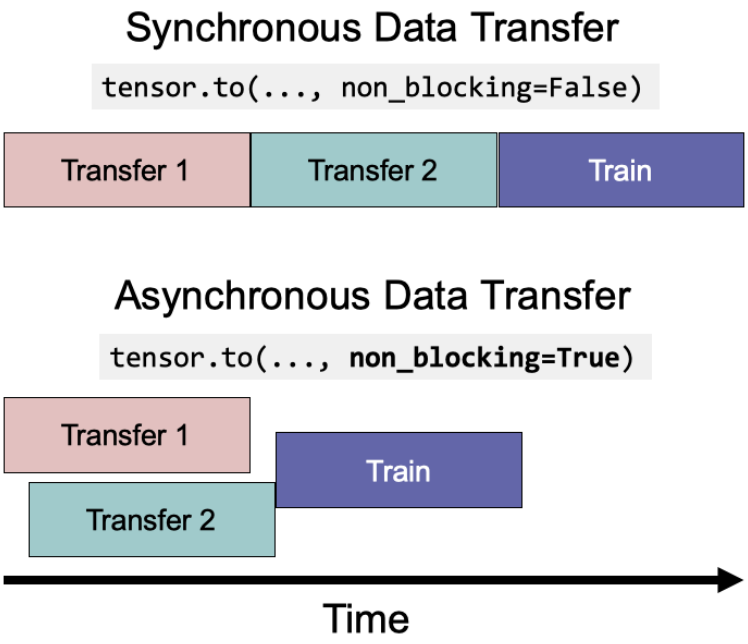
Kod direktnog rada s pojedinim tensorima potrebno ih je za optimalan rad stvoriti upravo na uređaju čije ćemo operacije nad njima izvršavati. Ovo sprječava bespotrebne prijenose podataka (tensora) između CPU-a i GPU-a. Isto je potrebno napraviti izbjegavajući sljedeće funkcije koje će zahtijevati transfer podataka između centralne procesorske jedinice i grafičke kartice.

```
# BAD! AVOID THEM IF UNNECESSARY!
print(cuda_tensor)
cuda_tensor.cpu()
cuda_tensor.to_device('cpu')
cpu_tensor.cuda()
cpu_tensor.to_device('cuda')
cuda_tensor.item()
cuda_tensor.numpy()
cuda_tensor.nonzero()
cuda_tensor.tolist()

# Python control flow which depends on operation
results of CUDA tensors
if (cuda_tensor != 0).all():
    run_func()
```

Slika 3: Pozivi koji uzrokuju vremenski zahtjevan prijenos podataka između CPU i GPU.

Preporučljiva praksa je i obraćanje pozornosti na "paralelizam na nivou kernela", tj. cilj je izbjeći čekanja dohvaćanje podataka dok se druga instrukcija izvršava. Ovo se postiže odobravanjem asinkronog prijenosa podataka, kao na donje navedenoj slici.




```
for features, target in loader:
    # these two calls are nonblocking and
overlapping
    features = features.to('cuda:0',
non_blocking=True)
    target = target.to('cuda:0', non_blocking=True)

    # This is a synchronization point
    # It will wait for previous two lines
    output = model(features)
```

Slika 4: Primjer asinkronog prijenosa podataka.

Jedan od optimizacijskih trikova primjenjivih na naš model, bio bi akumuliranje gradijenata tokom treniranja više batchova, tj. aržuriranje težina svako određeni broj batcheva kako bi se preciznije približavali minimumu funkcije koštanja. Ovo je izričito primjenjivo u slučajevima kada je zbog memorijskog ograničenja kartice ili velikih ulaznih podataka broj ulaza u batchu mali (nama je 8, zbog velike rezolucije slika).

```
for i, (features, target) in enumerate(dataloader):
    # Forward pass
    output = model(features)
    loss = criterion(output, target)

    # Backward pass
    loss.backward()

    # Only update weights every other 2 iterations
    # Effective batch size is doubled
    if (i+1) % 2 == 0 or (i+1) == len(dataloader):
        # Update weights
        optimizer.step()

        # Reset the gradients to None
        optimizer.zero_grad(set_to_none=True)
```

Slika 5: Primjer aržuriranja s preskakanjem batcheva.

Kako Nvidijine Tensor jezgre za matrično množenje najbolje rezultate postižu pri dimenzijama matrica zadanim potencijama broja 2, preporuča se da se veličine poput **veličine ulaznih podataka**, **batch sizea** i **veličine izlaznih podataka** budu zadane kao potencije broja 2. Kako je upravo matrično množenje **najčešće usko grlo** u strojnom učenju, ovakvo zadavanje donosi značajna poboljšanja. Istraživanje iz [10], pokazuje da su zapažena ubrzaanja od 1.3x pri usporedbi batcha veličine 33708 i 33712 ulaza te 4x usporedbom 4084 i 4095 ulaza. Naravno, poboljšanja ovise o arhitekturi grafičke kartice i cuDNN verziji.

Iako je to intuitivno i u YOLOv5 postavljeno za default opciju. Naime, izračun gradijenata nije potreban tokom faze inferencije i validacije, jer je u tim fazama fokus na proračunu izraza. PyTorch biblioteka doduše koristi zaseban memorijski međuspremnik za operacije s gradijentima. S toga je potrebno eksplicitno isključiti proračun gradijenata kako bi izbjegli trošenje resursa i bespotrebne proračune.

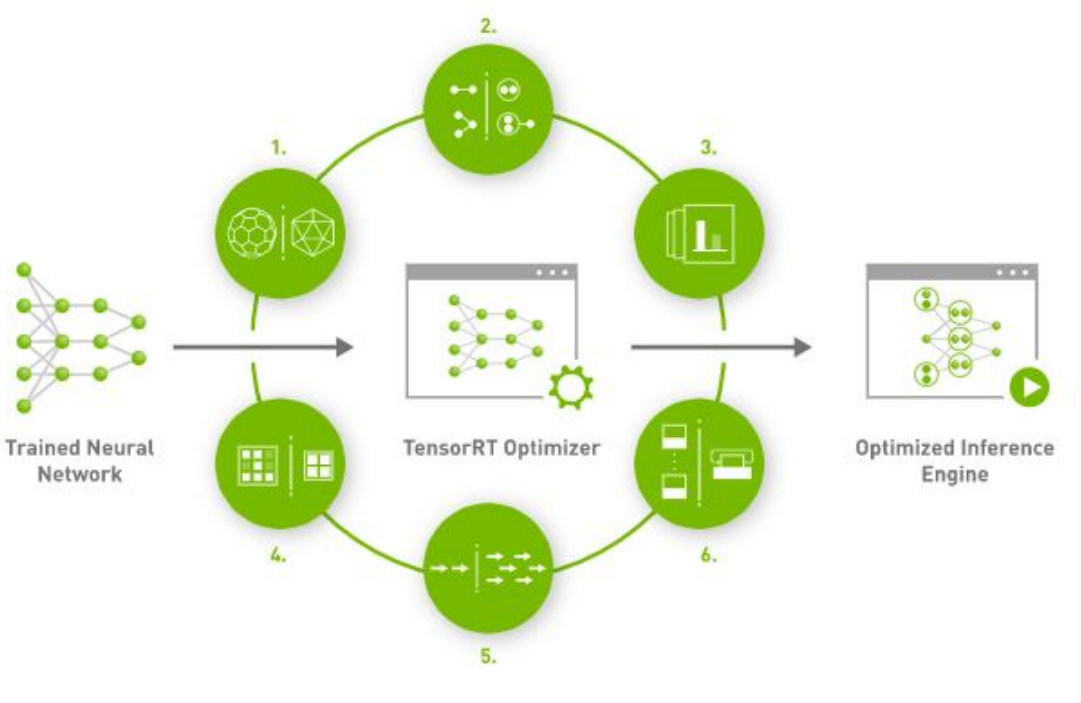
```
# torch.no_grad() as a context manager:
    with torch.no_grad():
    output = model(input)

# torch.no_grad() as a function decorator:
@torch.no_grad()
def validation(model, input):
    output = model(input)
return output
```

Slika 6: Primjer isključivanja proračuna gradijenata.

3.1 TensorRT

TensorRT je SDK (software development toolkit) za zaključivanje visokih performansi, te uključuje optimizaciju zaključivanja dubokog učenja i vrijeme izvođenja koje pruža nisko kašnjenje i visoku propusnost aplikacija zaključivanja. Aplikacije koje koriste Tensor su i do 36 puta brže od platformi koje se temelje samo na CPU. TensorRT omogućava razvojnim programerima da optimiziraju modele neuronskih mreža na svin glavnim platformama, te također pruža preciznost s vrlo viskom točnošću.

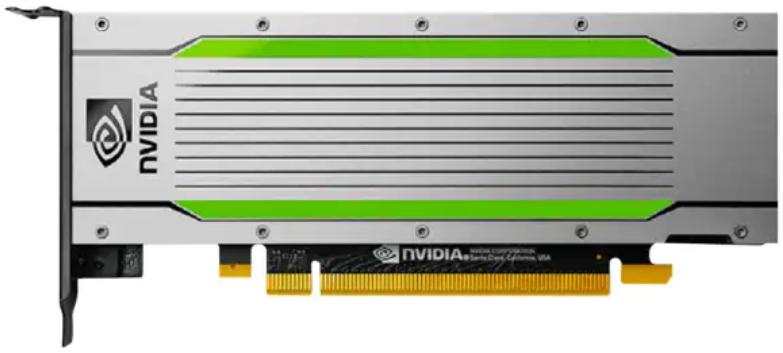


Slika 7: TensorRT.

TensorRT pruža INT8 koristeći kvantizaciju, te FP16 optimizaciju produkcije aplikacija za duboko učenje (video streaming, prepoznavanje govora, obrada prirodnog jezika,...). Smanjena preciznost zaključivanja značajno smanjuje kašnjenje aplikacija, što je vrlo korisna stvar kod usluga koje se pružaju u stvarnom vremenu, ali kao i kod ugradbenih aplikacija. Pomoću TensorRT programeri se mogu usredotočiti na stvaranje novih AI aplikacija umjesto na samu optimizaciju. Također optimizirani TensorRT modeli mogu biti implementirani uz NVIDIA Triton, softver otvorenog koda za posluživanje zaključivanja koji uključuje TensorRT kao jedan od svojih pozadinskih dijelova.

4 Tesla T4

Tesla T4 je profesionalna grafička kartica razvijena od NVIDIA, puštena u prodaju 13.09.2018. Tesla T4 napravljena je na 12 nm procesoru i bazirana na grafičkom procesoru TU104 te podržava DirectX 12 Ultimate. Što znači da se slike podižu na neku skroz novu razinu realističnosti sa podrškom za praćenje zraka, sjenčanje mreže, sjenčanje mreža s promjenjivom brzinom. Sadrži 13.6 milijardi tranzistora, GPU sa jednim slotom, ima 2560 CUDA jezgri i 320 Tensor jezgri. Također, 16 GB GDDR6 memorije je uparen sa T4, koji su povezani sa 256 bitnim memorijskim sučeljem. Radi na 1250 MHz ili 10Gbps što rezultira propusnošću memorije od 320 GB po sekundi [1].

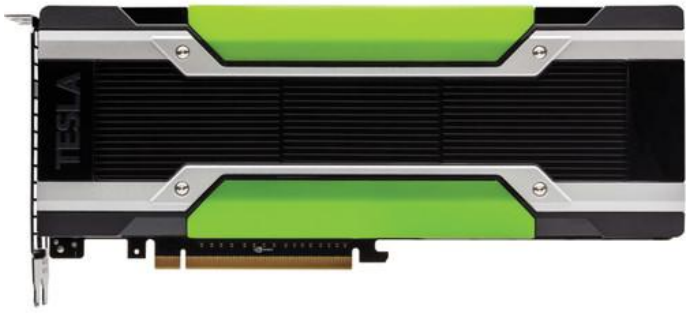


Slika 8: Tesla T4

T4 je optimizirana za maksimalnu iskoristivost u poslovnim podatkovnim centrima te za spremanje podataka u oblacima. Zahvaljujući optimiziranosti i revolucionarnoj više preciznoj izvedbi idealna je za obuku i treniranje dubokog učenja, strojnog učenja, analizu podataka i grafiku. Budući da nije dizajnirana da na njega budu povezani monitori, Tesla T4 nema utore za povezivanje s monitorom, ali ima mnoge druge značajke. Jedna od značajki jest da T4 koristi manje energije, samo 70 W, što može uštedjeti energiju a samim time smanjiti troškove tijekom vremena. GPU ne zahtjeva nikakav dodatni priključak za napajanje, stoga je spojen na sustav pomoću PCIe3.0 x16 sučelja, te ima hlađenje s jednim utorom.

5 Tesla K80

Tesla K80 je kao i Tesla T4 profesionalna grafička kartica razvijena od strane NVIDIA, te puštena u prodaju 17.11.2014. Izgrađena je na 28nm procesoru i bazirana na grafičkom procesoru GK210 te podržava DirectX12. Za razliku od T4 kod ove verzije NVIDIA se odlučila da kombinira dva grafička procesora za povećanje performansi. NVIDIA je uparila 24 GB GDRR5 memorije s K80, koji su povezani sa 384 bitnim memorijskim sučeljem, te radi na frekvenciji od 1253 MHz ili oko 5Gbps.

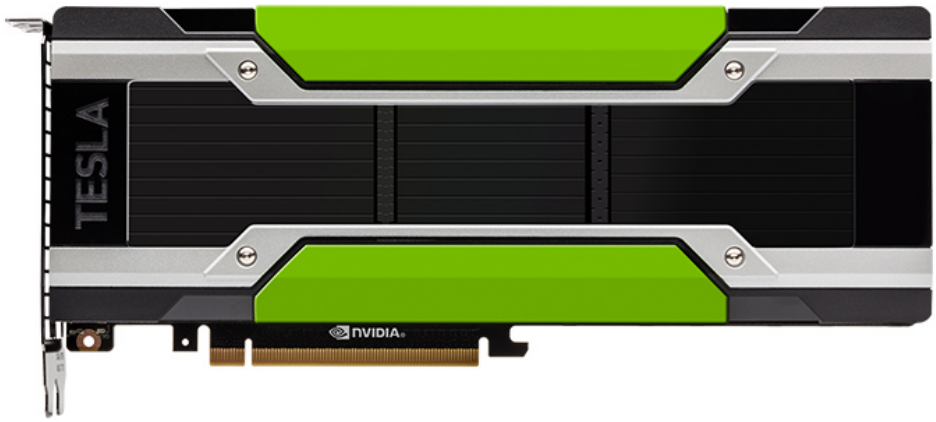


Slika 9: Tesla K80

S obzirom da se radi o kartici s dva utora Tesla K80 crpi energiju s 8-pinskog konektora za napajanje, s maksimalnom snagom od 300 W. Ovaj uređaj nema mogućnost povezivanja sa zaslonom jer nije dizajniran da se na njega povezuju monitori. GPU je povezan s ostatkom sustava pomoću PCIe3.0 x16 sučelja, te ima rješenje za hlađenje s dva utora.

6 Tesla P100

Tesla P100 profesionalna je grafička kartica razvijane od strane Nvidia, te je lansirana u prodaju 20.06.2016. Izrađena je na 16 nm procesoru i bazirana na GP100 grafičkom procesoru. Kao i K80 podržava DirectX12. Koristi se u mnogim svjetskim super računalima koja služe za ubrzanje kompleksnih poslova. Podržava *Unified Memory page* što omogućuje aplikacijama da idu iznad same fizičke veličine memorije ove grafičke čime se praktički omogućuju neograničene razine za učinkovitiju obradu. Kod ove grafičke kartice Nvidia se odlučila na drugu generaciju *High Bandwidth Memory* tj. HBM2. HMB2 podržava *error-correcting code* zaštitu memorije za veću pouzdanost. Upareno je 16 GB HBM2 memorije s P100 preko 4096 bitnim memorijskim sučeljem, te radi na frekvenciji od 715MHz ili oko 1430 Mbps.



Slika 10: Tesla P100

Tesla P100 sadrži 3584 jezgri te s ovolikom snagom omogućuje da jedan čvor grafičke kartice zamijeni pola CPU čvorova u širokom rasponu aplikacija (aplikacije za prepoznavanje objekata na slikama, prepoznavanje govora) pružajući munjevitu brzinu rada. S obzirom da se radi o grafičkoj s dva utora P100 crpi napajanje s 8-pinskog konektora, s maksimalnom snagom od 250W. Također, kao i ostale objašnjene grafičke kartice nije namijenjena za povezivanje s monitorima pa nema dodatne utore. Nvidia P100 povezana je s ostatkom sustava pomoću PCIe3.0 x16 sučelja, te koristi pasivnu rashladnu jedinicu (ovisi o unutarnjem protoku zraka sustava) kako bi povećala pouzdanost i smanjila potrošnju energije.

7 Usporedba

Za početak ćemo prikazati i usporediti glavne karakteristike svih grafičkih kartica na kojima ćemo pokrenuti program u Colab-u. Kao što se može vidjeti iz tablice 1 T4 je nešto novija od ostalih pa je danas najčešća u upotrebi. P100 ima najbolji GPU *clock speed* što znači da će i samim time procesor brže izvršavati zadaće. T4 u usporedbi sa K80 koristi GDDR6 tip memorije koja je dovela do napretka u odnosu na GDDR5, što se može i vidjeti iz tablice 1. Memorija je veća pa može pohraniti više podataka, također brzina kojom se podatci upisuju/dohvaćaju je također veća pa memorija brže radi sa samim podacima. P100 u odnosu na prethodne dvije vrste GPU koristi HBM2 memoriju. Smatra se da je ovaj tip memorije donio puno koristi istraživačima i znanstvenicima u područjima umjetne inteligencije i neuralnih mreža jer je dovela do rješavanja projekata koji su dosad bili ne rješivi. HMB2 ima prednost na preostalim dvima zbog same veličine memorije i brzine rada, ali danas se sve manje upotrebljava zbog prevelike cijene i potrošnje energije jer je manje ekonomična od preostalih GPU. Nvidia P100 ima najveći broj jezgri pa samim time i više zadataka može paralelno obrađivati u istom trenutku. Uz veliki broj jezgri i veliku brzinu GPU Nvidia P100 se čini kao najbolja grafička od ovih triju koje nudi Google Colab.

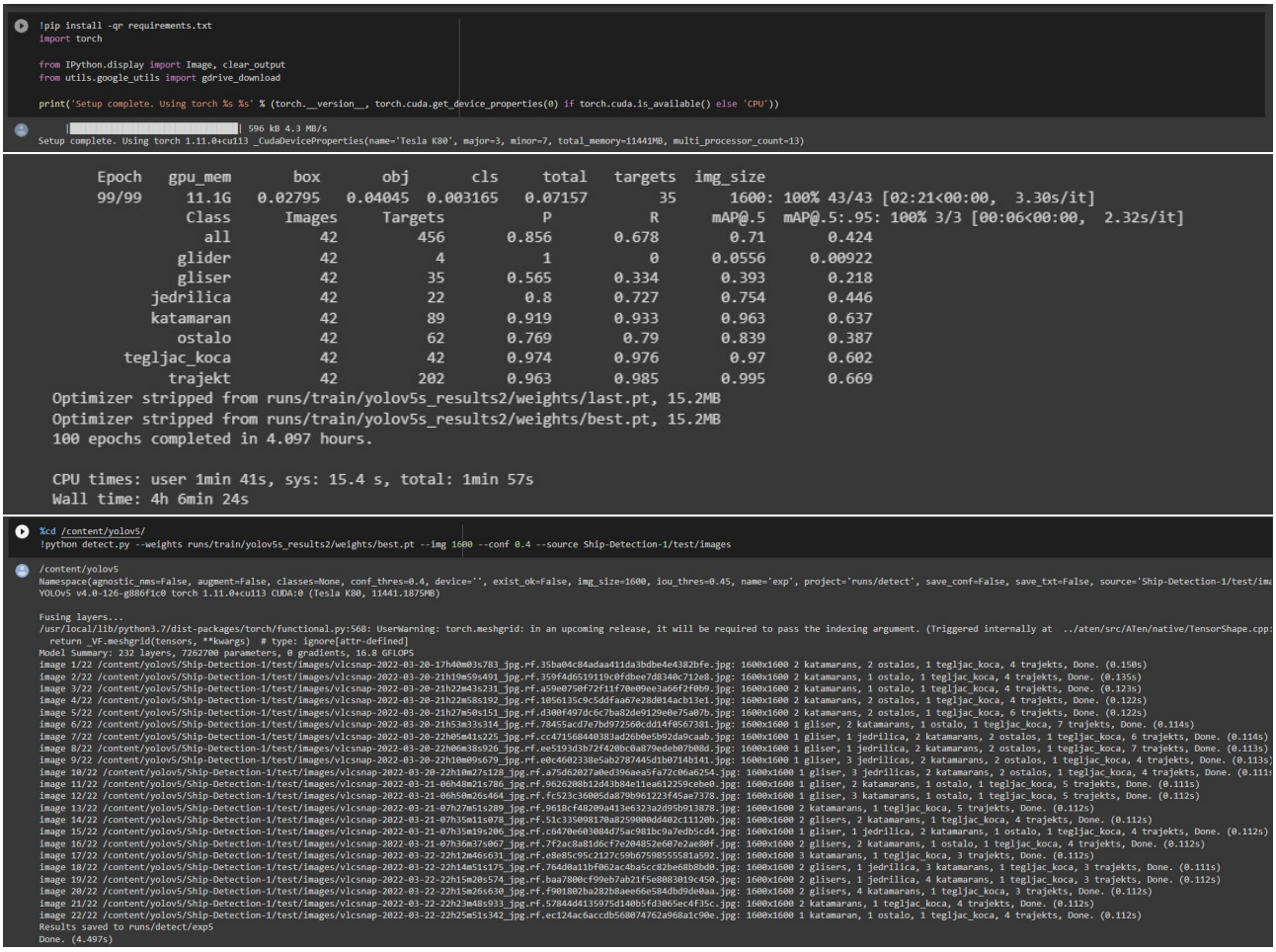
	K80	T4	P100
Datum izlaska	17.11.2014.	13.09.2018.	20.06.2016.
Generacija	Tesla	Tesla	Tesla
GPU procesor	GK210	TU104	GP100
GPU *Clock speed*	745 MHz	1005 MHz	1190 MHz
Arhitektura	Kepler 2.0	Turing	Pascal
Broj jezgri	2496 x2	2560	3584
Broj tranzistora	7,100 milijuna	13,600 milijuna	15,300 milijuna
Tip memorije	GDDR5	GDDR6	HBM2
Veličina memorije	12GB x2	16 GB	16GB
Sabirnica memorije	384 bita x2	256 bita	4096 bita
bandwidth	240.6 GB/s x2	320.0 GB/s	732GB/s
Broj slotova	2	1	2

Tablica 1: Usporedba GPU

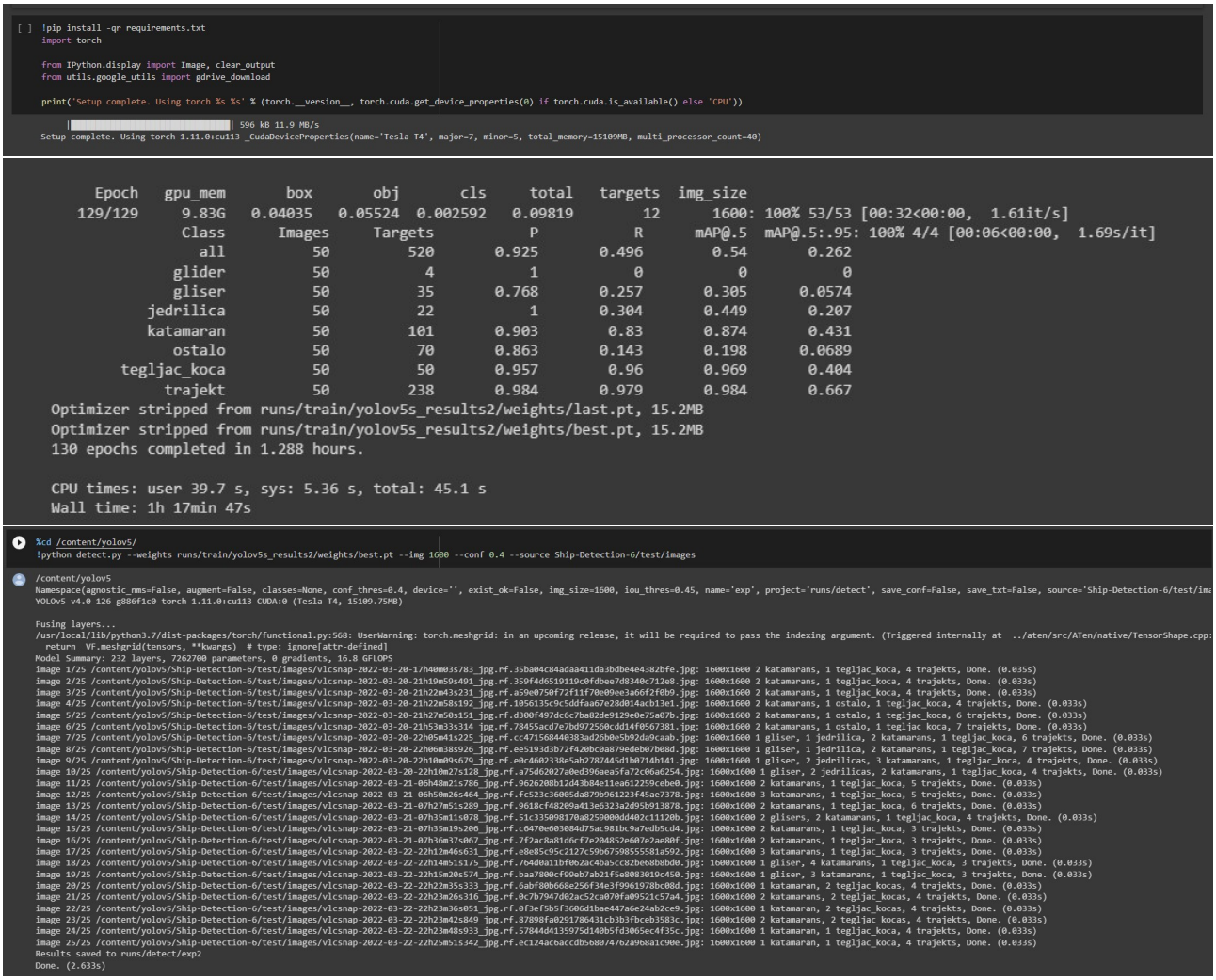
Program koji ćemo pokrenuti biti će model za detekciju plovila u trajektnoj luci temeljen na YOLOv5. Nakon što se program izvrši na svim grafičkim karticama na temelju dobivenih podataka usporediti ćemo koja je grafička kartica brže i kvalitetnije izvršila program.

	K80	T4	P100
Broj epoha	100	130	130
Batch size	10	8	8
Ukupno vrijeme treniranja	4h 6min 24s	1h 17min 47s	1h 32min 44s
Prosječno vrijeme po epohi	2min 27,84s	35,9s	42,8s
Ukupno vrijeme inferencije	4.497s	2.633s	2.638s
Broj slika za inferenciju	22	25	25
Prosječno vrijeme po slici	0,204409s	0,10532s	0,10552s

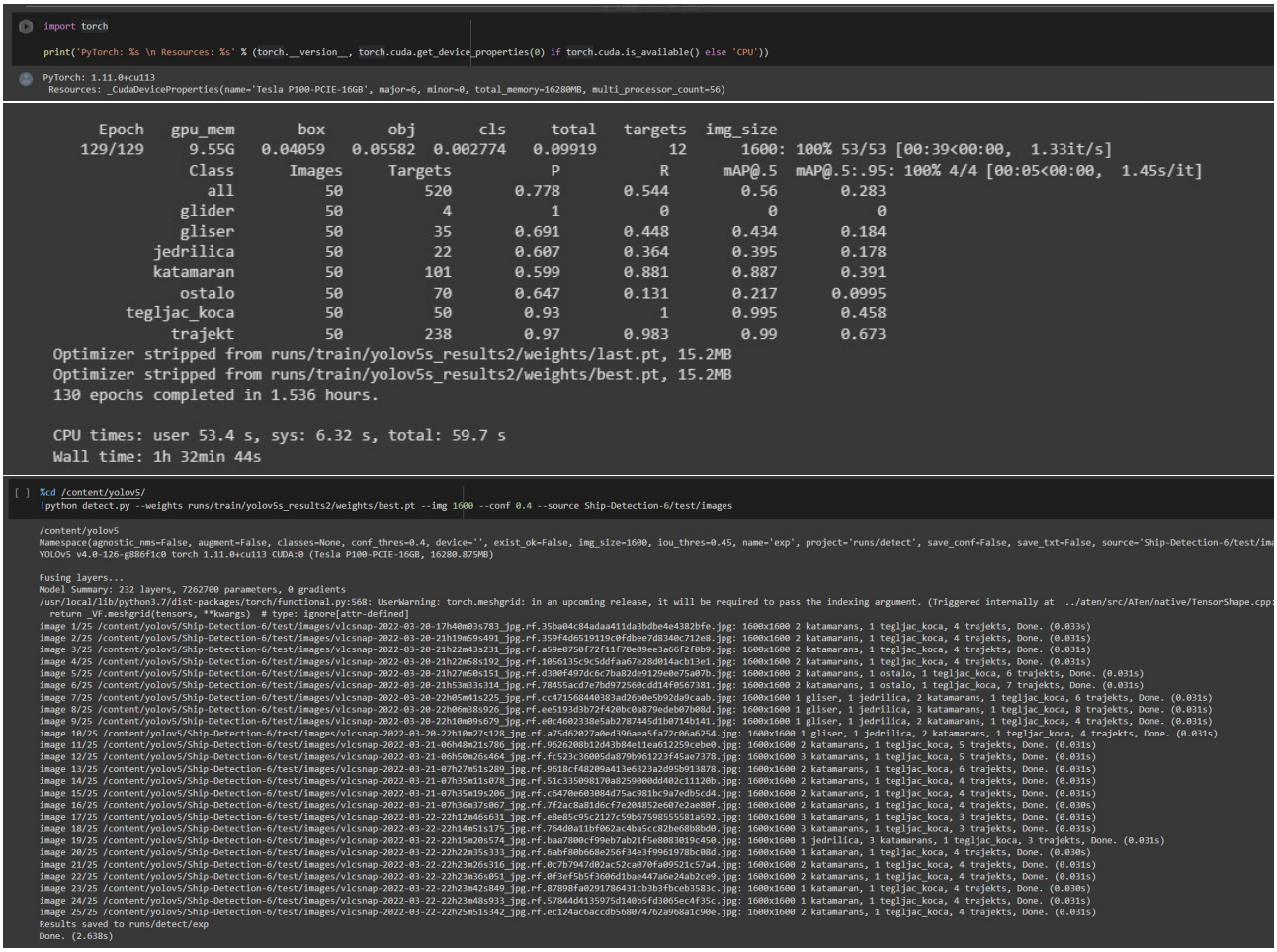
Tablica 2: Usporedba vremena kartica



Slika 11: Izvođenje treniranja i inferencije na Tesla K80 kartici.



Slika 12: Izvođenje treniranja i inferencije na Tesla T4 kartici.



Slika 13: Izvođenje treniranja i inferencije na Tesla P100 kartici.

Literatura

[1] NVIDIA Tesla T4 GPU - System Overview, s interneta: https://www.itcreations.com/nvidia-gpu/nvidia-tesla-t4-gpu, zadnji pristup: 15.06.2022.

[2] NVIDIA Tesla K80, s interneta: https://www.techpowerup.com/gpu-specs/tesla-k80.c2616, zadnji pristup: 15.06.2022.

[3] NVIDIA Tesla P100, s interneta: https://www.techpowerup.com/gpu-specs/tesla-p100-pcie-16-gb.c2888, zadnji pristup: 15.06.2022.

[4] Why Deep Learning Uses GPUs?, s interneta: https://towardsdatascience.com/why-deep-learning-uses-gpus-c61b399e93a0, zadnji pristup: 20.06.2022.

[5] Why GPUs are more suited for Deep Learning?, s interneta: https://www.analyticsvidhya.com/blog/2020/09/why-gpus-are-more-suited-for-deep-learning/, zadnji pristup: 20.06.2022.

[6] Why does machine learning use GPUs?, s interneta: https://www.quora.com/Why-does-machine-learning-use-GPUs, zadnji pristup: 20.06.2022.

[7] Making the Most of GPUs for Your Deep Learning Project, s interneta: https://www.run.ai/guides/gpu-deep-learning, zadnji pristup: 20.06.2022.

[8] Optimize NVIDIA GPU performance for efficient model inference, s interneta: https://towardsdatascience.com/optimize-nvidia-gpu-performance-for-efficient-model-inference-f3e9874e9fdc, zadnji pristup: 20.06.2022.

[9] Performance tuning guide, s interneta: https://pytorch.org/tutorials/recipes/recipes/tuning_guide.html, zadnji pristup: 20.06.2022.

[10] Optimize PyTorch Performance for Speed and Memory Efficiency (2022), s interneta: https://towardsdatascience.com/optimize-pytorch-performance-for-speed-and-memory-efficiency-2022-84f453916ea6, zadnji pristup: 20.06.2022.

[11] Accelerating Inference Up to 6x Faster in PyTorch with Torch-TensorRT, s interneta: https://developer.nvidia.com/blog/accelerating-inference-up-to-6x-faster-in-pytorch-with-torch-tensorrt/, zadnji pristup: 20.06.2022.

[12] Improving PyTorch inference performance on GPUs with a few simple tricks, s interneta: https://tullo.ch/articles/pytorch-gpu-inference-performance/, zadnji pristup: 20.06.2022.

[13] Leveraging PyTorch to Speed-Up Deep Learning with GPUs, s interneta: https://www.analyticsvidhya.com/blog/2021/10/leveraging-pytorch-to-speed-up-deep-learning-with-gpus/, zadnji pristup: 20.06.2022.

[14] NVIDIA Deep Learning Performance, s interneta: https://docs.nvidia.com/deeplearning/performance/index.html, zadnji pristup: 20.06.2022.

[15] optimizing yolo5 for detecting objects from recorded video, s interneta: https://github.com/ultralytics/yolov5/discussions/2645, zadnji pristup: 20.06.2022.

[16] Hyperparameter Evolution, s interneta: https://docs.ultralytics.com/tutorials/hyperparameter-evolution/, zadnji pristup: 20.06.2022.