Univerzitetna založba
Univerze v Mariboru

Demonstracijska celica za prikaz globokega učenja v praktičnih aplikacijah

Domen Tabernik,[1] Peter Mlakar,[1] Jakob Božič,[1] Luka Čehovin Zajc,[1] Vid Rijavec,[1] Danijel Skočaj[1]

[1] Univerza v Ljubljani, Fakulteta za računalništvo in informatiko, Ljubljana, Slovenija
domen.tabernik@fri.uni-lj.si, danijel.skočaj@fri.uni-lj.si

Izvleček. V zadnjih letih so metode globokega učenja postale ključno orodje za reševanje raznolikih praktičnih izzivov. Kljub temu pa potencial takih metod pogosto ostaja slabo razumljiv širši javnosti zaradi pogostega ločevanja razvoja in demonstracije algoritmov od dejanskih praktičnih problemov, ki jih algoritmi naslavljajo. V tem članku predstavljamo demonstracijsko celico, ki združuje strojno in programsko opremo ter algoritme globokega učenja, omogočajoč enostavno prikazovanje delovanja teh metod v različnih aplikativnih domenah. Celica vključuje kamere, grafični vmesnik in pet demonstracijskih programov, ki demonstrirajo klasifikacijo lesenih desk, detekcijo površinskih anomalij, štetje polipov, detekcijo prometnih znakov in detekcijo vogalov tekstilnih izdelkov. Implementiran modularni pristop omogoča enostavno integracijo različnih algoritmov globokega učenja. Sistem omogoča boljše razumevanje in uporabo teh metod v praktičnih scenarijih ter prispeva k razvoju inovativnih rešitev na področju globokega učenja.

Ključne besede: Demonstracijska celica, Globoko učenje, Integracija algoritmov, Praktične aplikacije, Klasifikacija, Detekcija anomalij, Štetje polipov, Prometni znaki, Detekcija vogalov tekstilnih izdelkov

Demonstration Cell for Showcasing Deep Learning in Practical Applications

Domen Tabernik,[1] Peter Mlakar,[1] Jakob Božič,[1] Luka Čehovin Zajc,[1] Vid Rijavec,[1] Danijel Skočaj[1]

[1] University of Ljubljana, Faculty of Computer and Information Science, Ljubljana, Slovenia
domen.tabernik@fri.uni-lj.si, danijel.skočaj@fri.uni-lj.si

Abstract. In recent years, deep learning methods have become a crucial tool for solving diverse practical challenges. However, their potential often remains poorly understood by a broader audience due to a separation between development and demonstration of algorithms and actual problems being solved by those algorithms. In this article, we introduce an innovative demonstration cell that combines hardware, software, and deep learning algorithms, enabling easy showcasing of these methods in various application domains. The cell includes cameras, a graphical interface, and five demonstration programs that demonstrate the classification of wooden boards, detection of surface anomalies, counting of polyps, recognition of traffic signs, and detection of corners on towels and cloths. The implemented modular approach allows for the straightforward integration of different deep learning algorithms. The system enhances understanding and application of these methods in practical scenarios, contributing to the development of innovative solutions in the field of deep learning.

Keywords: Demonstration cell, Deep learning, Algorithm integration, Practical applications, Classification, Anomaly detection, Polyp counting, Traffic signs, Grasp point detection

1 Uvod

Metode globokega učenja so v zadnjih letih postale izredno močno orodje za reševanje raznolikih praktičnih problemov. Navkljub temu pa je njihova

demonstracija širši publiki pogosto težavna, saj se razvoj in demonstracija algoritmov vršita povsem ločeno od končnih sistemov, kjer taki algoritmi dejansko rešujejo praktične probleme. Zaradi tega širša publika pogosto spregleda potencial takih algoritmov za uporabo v aplikativnih domenah.

V ta namen smo ustvarili demonstracijsko celico s katero lahko enostavno prikažemo delovanje najnovejših metod globokega učenja v kontekstu različnih praktičnih problemov. Celica sestoji iz ohišja, kamer, strojne opreme, programske opreme za njen nadzor ter iz vrsto demonstracijskih programov z algoritmi globokega učenja. Glavni cilj pri zasnovi celice je bila enostavna uporaba na eni strani ter na drugi strani enostavna integracija različnih vrst algoritmov globokega učenja, ki so pogosto razviti za raziskovalne namene ter zahtevajo specifično programsko okolje za pravilno delovanje. V ta namen je bilo razvito posebno programsko ogrodje skupaj s petimi različnimi demonstracijskimi programi, kjer vsak rešuje specifičen praktičen problem ter jih je mogoče demonstrirati na realnih predmetih postavljenimi v celico.

2 Opis sistema

Demonstracijska celica je sestavljena iz ohišja, strojne opreme ter namensko ustvarjenje programske opreme za poganjanje poljubnih metod globokega učenja v demonstracijskem načinu.

2.1 Ohišje in strojna oprema

Celotna demonstracijska celica obsega 60 x 60 x 120 cm veliko ogrodje sestavljeno iz ALU profila, ki je razdeljeno na dva prekata: i) glavni demonstracijski prekat ter ii) manjši prekat višine 20 cm za strojno opremo na dnu celice. Na vrhu celice se nahajajo nosilci za montažo senzorjev, ki so usmerjeni navzdol, ter 6 programsko nadzorovanih LED luči. Celica vsebuje 12MP kamero *Allied Vision Alvium 1800 U-1240c* za zajem slik ter *Kinect Azure* za zajem slik z globinsko informacijo, sočasno pa omogoča enostavno montažo dodatnih kamer z nosilci pritrjenimi na ALU profile. Prekat za strojno opremo vsebuje modul za napajanje in nadzor luči ter majhen glavni računalnik, sestavljen iz procesorja Intel Core i5-11600KF 3.90GHz, 16 GB pomnilnika ter grafične enote NVIDIA GeForce RTX 3060 z 12GB pomnilnika. Celica vsebuje še 24" LCD zaslon na dotik za upravljanje in vizualizacijo demonstracijskih programov. Celotna demonstracijska celica je prikazana na Sliki 1.

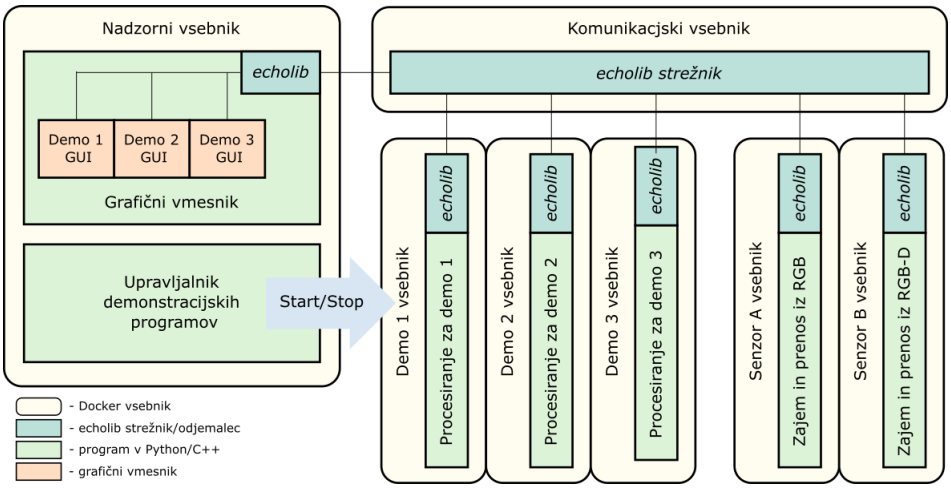
Slika 1: Demonstracijska celica.
Vir: lasten.

2.2 Programska oprema

Pri razvoju programske opreme smo zasledovali dva pomembna cilja: i) enostavno uporabo z zaslonom na dotik, ter ii) enostavno integracijo različnih vrst demonstracijskih programov. V ta namen smo razvili namenski grafični vmesnik enostaven za upravljanje na dotik, ter celotno programsko okolje zasnovali modularno na osnovi sistema vsebnikov. Sistem je tako razdeljen na: i) grafični vmesnik z modulom za upravljanje z demonstracijskimi programi, ii) na podsistem za zajem in prenos slik, iii) na posamezne demonstracijske programe, ter iv) na komunikacijski podsistem. Arhitektura programske celice je prikazana na Sliki 2.

Vsak od štirih podsistemov je implementiran v svojem vsebniku, s čimer povsem zamejimo odvisnosti od knjižnic na vsak podsistem. Tak pristop primarno razrešuje problem odvisnosti knjižnic pri različnih demonstracijskih programih, kjer je lahko vsak program implementiran na različen način in z uporabo različnih programskih

jezikov in orodij. Na primer, prvi demonstracijski program je lahko implementiran v programskem jeziku Python z ogrodjem PyTorch 1.9, ki zahteva knjižnico CUDA 10, medtem ko je drugi program lahko implementiran v jeziku C++ z ogrodjem TensorFlow v2, ki zahteva knjižnico CUDA 11. Ker je v našem sistemu vsak demonstracijski program implementiran v svojem vsebniku, pa tako povsem izločimo potrebo po sočasni prisotnosti med seboj nekompatibilnih knjižnic. V praksi implementiramo vse podsisteme z orodjem Docker.



Slika 2: Programska arhitektura demonstracijske celice.
Vir: lasten.

Podsistem za zajem in prenos slik je prav tako implementiran v ločenem vsebniku Docker. Tako omogočimo enostavno uporabo različnih tipov kamer, ki zahtevajo vsak svoje gonilnike ter ogrodja. Za demonstracijsko celico smo implementirali dva vsebnika: i) vsebnik za prenos slik iz senzorja *Allied Vision Alvium 1800*, ter ii) vsebnik za prenos slik iz senzorja *Kinect Azure*. Prvi omogoča zajem RGB slik, medtem ko drugi omogoča zajem RGB-D slik ter dodatno tudi informacijo iz inercijskega senzorja.

Kamere, demonstracijski programi ter podsistem za upravljanje demonstracijskih programov z grafičnim vmesnikom med seboj komunicirajo preko dodatnega komunikacijska podsistema. Komunikacija je mogoča na podlagi vzorca objavi-naroči (ang. *publish-subscribe*), podobno kot pri sistemu ROS. Komunikacijski podsistem smo implementirali v ločenem Docker vsebniku z odprtokodno knjižnico

echolib[1], ki je prenosljiva ter lahka implementacija protokola objavi-naroči. Posameznimi podsistemi tako komunicirajo med seboj preko različnih tem (npr. tema za prenos slik, tema za prenos rezultatov za prikaz, tema za nadzor demonstracijskega programa preko grafičnega vmesnika, itd.). Tak način komuniciranja omogoča tudi sočasen prenos slik iz več kamer, kjer lahko vsaka kamera prenaša slike na svoji temi, nato pa se demonstracijski program odloči katero kamero bo uporabil.

Pomemben del predstavlja tudi podsistem z grafičnim vmesnikom ter modulom za upravljanje z demonstracijskimi programi. Grafični vmesnik smo zasnovali na podlagi odprtokodne knjižice[2], ki omogoča uporabo ogrodja OpenGL preko programskega jezika Python. Sam grafični vmesnik je relativno preprost ter primarno omogoča preklapljanje med posameznimi demonstracijskimi programi. Preklapljanje med demonstracijskimi programi nadzira modul za upravljanje z demonstracijskimi programi, ki je implementiran v programskem jeziku Python. Na vsako zahtevo po prikazu specifičnega demonstracijskega programa tako modul požene in ustavi posamezne vsebnike z demonstracijskimi programi.

2.2 Implementacija demonstracijskega programa

Posamezen demonstracijski program ter njegov grafični vmesnik sta implementirana ločeno od preostalih podsistemov. S tem omogočimo enostavno dodajanje novih algoritmov. Glavni del programa predstavlja aplikacija, ki se avtomatsko požene ob zagonu vsebnika, ter se mora preko komunikacijskega podsistema *echolib* povezati s kamerami ter z grafičnim vmesnikom. V vsebniku se tako izvede priklop na zajem in prenos slik iz specifičnega senzorja ter ustrezna obdelava slik, kar se nato pošlje v glavni grafični vmesnik za izris na zaslon. Del tega procesa je tudi prikaz slike ter rezultatov detekcij, ki se izvede že znotraj demonstracijskega vsebnika ter tako omogoča enostavno integracijo različnih obdelav in prikazov rezultatov.

Del demonstracijskega programa je tudi grafični vmesnik za nadzor posameznih funkcij procesiranja (npr. gumb za zagon detekcije, nastavitev upragovanja, itd.), ki pa ni implementiran znotraj demonstracijskega vsebnika, ampak se dinamično vključi v glavni grafični vmesnik ob zagonu programa. Zaradi tega poteka

[1] https://github.com/vicoslab/echolib
[2] https://github.com/petermlakar/opengl_gui

komunikacija med grafičnim vmesnikom za demonstracijski program ter njegovim zajemom in procesiranjem slik preko komunikacijskega podsistema *echolib*.

3 Demonstracijski programi

Predstavljen sistem je bil zasnovan za enostavno integracijo različnih demonstracijskih programov za prikaz delovanja algoritmov globokega učenja. Implementirali smo pet spodaj opisanih programov.

3.1 Klasifikacija lesenih desk

Demonstracijski program predstavlja praktičen primer klasifikacije lesenih desk v kakovostne razrede, ki je bil predstavljen v (Muhovič, Tabernik, & Skočaj, 2020). Za model smo izbrali nevronsko mrežo EfficientNet-B4, ki smo jo naučili za razlikovanje med 10 različnimi tipi lesenih površin (primeri na Sliki 3). Za učenje smo uporabili 290 slik zajetih na istem demonstracijskem sistmu (29 slik za vsako kategorijo) ter učili model za 200 epoh z optimizatorjem Adam ter funkcijo izgube binarne prečne entropije.

Slika 3: Deset kategorij kakovosti lesa.
Vir: lasten.

Pri napovedovanju (ter tudi pri učenju) se iz slike predhodno izreže posamezno desko z enostavnim upragovanjem (ozadje je vedno črne barve) ter poravna in poveča na velikost 1512 x 536 pikslov. Grafični vmesnik demonstracijskega programa vsebuje gumb za izvedbo klasifikacije, ter prikaz rezultata v obliki segmentacije posamezne deske ter številko razpoznanega razreda deske (Slika 5). Klasifikacijski program je implementiran v programskem okolju Python z ogrodjem PyTorch v1.7.1 ter knjižnico CUDA v11.2 in CuDNN v8 na osnovi sistema Ubuntu 18.04.

3.2 Detekcija površinskih anomalij

Za demonstracijo nenadzorovanih metod detekcije površinskih anomalij, kjer model učimo na izključno dobrih primerih, smo izbrali metodo DREAM (Zavrtanik, Kristan, & Skočaj, 2021), ter jo aplicirali na detekcijo anomalij na ploščicah. Metoda DRAEM, predstavlja rekonstrukcijski tip nenadzorovanih metod učenja normalnega izgleda, kjer se z arhitekturo kodirnik-dekodirnik izvede rekonstrukcijo slike, nato pa z dodatnim diskriminativnim kodirnik-dekodirnikom, naučenim na sintetičnih napakah, poskuša detektirati odstopanja od normalnega izgleda, t.j., anomalije. Za učenje smo pripravili 300 slik ploščic brez napak, ter model učili 50 epoh z optimizatorjem Adam.

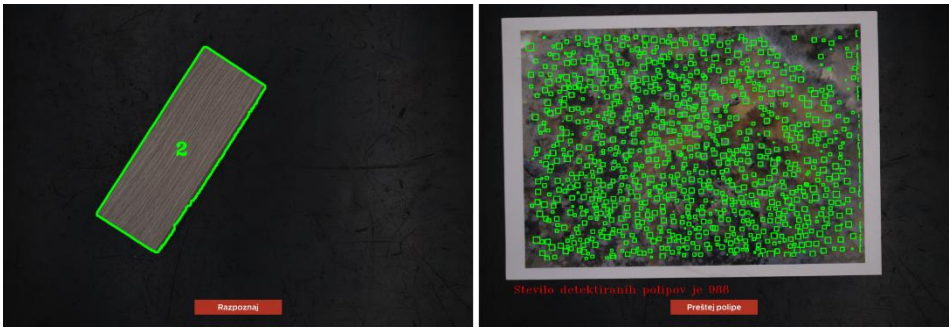
Slika 4: Detekcija anomalij na ploščicah. Levo: dober primer, desno: anomalija (verjetnost anomalije v oklepajih).
Vir: lasten.

Pri napovedovanju (ter tudi pri učenju) se iz slike predhodno izreže posamezno ploščico z enostavnim upragovanjem ter poravna in poveča na velikost 480 x 480 pikslov. Grafični vmesnik demonstracijskega programa vsebuje gumb za izvedbo detekcije, ter prikaz rezultata v obliki segmentacije posamezne ploščice ter indikator prisotnosti anomalije v obliki barve segmentacije, kot je prikazano na Sliki 4 (rdeča – anomalija, zelena – brez anomalije). Detektor anomalij je implementiran v programskem okolju Python z ogrodjem PyTorch v1.7.1 ter knjižnico CUDA v11.2 in CuDNN v8 na osnovi sistema Ubuntu 18.04.

3.3 Štetje polipov

Za demonstracijski program smo implementirali tudi model PoCo (Zavrtanik, Vodopivec, & Kristan, 2020) za štetje polipov. Metoda PoCo temelji na arhitekturi

U-Net, ter je sestavljena iz kodirnika in dekodirnika, ter vmesnimi povezavami. Metoda šteje polipe na podlagi segmentacije, kjer se okoli polipov segmentira krog v odvisnosti od velikosti polipov, nato pa se z pomočjo transformacije razdalje detektira in prešteje prekrivajoče se polipe z uporabo konsenza razdalij (ang. *distance consensus points*). Model je bil naučen na 37 slikah s 32.685 označenimi polipi.



Slika 5: Demonstracija klasifikacije lesa (levo) ter štetja polipov (desno)
Vir: lasten.

Ker se objekti naravno nahajajo le v morju, le teh ni mogoče fizično prikazati na demonstratorju, zato za demonstracijo delovanja prikažemo natisnjeno sliko s zajetimi polipi. Pri napovedovanju se zato iz slike odstrani ozadje z enostavnim upragovanjem ter sliko s polipi poravna in poveča na velikost 2080 x 1470 pikslov. Grafični vmesnik demonstracijskega programa vsebuje gumb za izvedbo štetja, ter prikaz rezultata v obliki številke detektiranih polipov (Slika 5). Program za štetje polipov je implementiran v programskem okolju Python z ogrodjem TensorFlow v2.6.2 ter knjižnico CUDA v11.2 in CuDNN v8 na osnovi sistema Ubuntu 18.04.

3.4 Detekcija prometnih znakov

Implementirali smo tudi demonstracijski program za prikaz algoritma detekcije prometnih znakov predstavljenim v (Tabernik & Skočaj, 2019). Detektor sloni na metodi Faster/Mask R-CNN (He, Gkioxari, Dollár, & Girshick, 2017), naučeni za detekcijo 200 različnih kategorij prometnih znakov. Model je bil naučen na podatkovni množici s preko 5.000 slik zajetih na slovenskih cestah, ki vsebuje preko 10.000 označenih prometnih znakov. Za hrbtenico smo vzeli arhitekturo ResNet50, ki omogoča izvajanje v realnem času. Model je bil učen 95 epoh, kjer so bile slike zmanjšane, tako da je bila najmanjša stranica velika 840 pikslov.

Slika 6: Detekcija prometnih znakov.
Vir: lasten.

Grafični vmesnik demonstracijskega programa vsebuje gumb za izvedbo detekcije, ter prikaz rezultata v obliki očrtanih pravokotnikov okoli prometnih znakov. Dodatno izrišemo tudi zaznano kategorijo v obliki slike tipičnega predstavnika te kategorije, kot je to prikazano na Sliki 6. Program napovedovanja je implementiran v programskem okolju Python z ogrodjem Detectron in Caffe2 ter knjižnico CUDA v11.1 in CuDNN v8 na osnovi sistema Ubuntu 16.04.

3.5 Detekcija vogalov brisač in krp

Implementirali smo tudi demonstracijski program za detekcijo vogalov brisač in krp, ki se lahko uporablja kot detektor točk prijema za robotske aplikacije. Za detekcijo kotnih robov smo aplicirali model CeDiRNet (Tabernik, Muhovič, & Skočaj, 2023), ki sloni na dvostopenjski arhitekturi z regresijo smernih vektorjev na prvi stopnji, ter lokalizacijo točk na drugi stopnji. Metoda je bila naučena na preko 5.000 učnih slikah s prikazanimi brisačami v raznolikih konfiguracijah ter svetlobnimi pogoji.

Naučena metoda vsebuje tudi možnost uporabe globinske informacije zato za ta demonstracijski program omogočamo uporabo tako navadne RGB kamere (*Allied Vision*) kot tudi RGB-D kamere (*Kinect Azure*). V ta namen grafični vmesnik poleg gumba za zagon detekcije vsebuje tudi opcijo za izbiro vira kamere. Rezultat detekcije prikažemo v obliki detektiranih točk, dodatno pa prikažemo tudi smer, ki določa potencialen kot pristopa za robotsko roko, kot je to prikazano na Sliki 7. Detektor vogalov je implementiran v programskem okolju Python z ogrodjem PyTorch v1.13.1 ter knjižnico CUDA v11.7 in CuDNN v8 na osnovi sistema Ubuntu 20.04.

Slika 7: Detekcija vogalov na brisači.

Vir: lasten.

4 Zaključek

V članku smo predstavili demonstracijsko celico, ki omogoča preprost prikaz delovanja metod globokega učenja v različnih praktičnih aplikacijah. Celica, ki združuje strojno in programsko opremo, ponuja modularno okolje za enostavno integracijo različnih algoritmov globokega učenja. Z izvedbo petih demonstracijskih programov, ki segajo od klasifikacije do detekcije, demonstracijska celica ponuja vpogled v uporabo globokega učenja v resničnih scenarijih. Razviti sistem prispeva k širšemu razumevanju potenciala in praktične uporabe metod globokega učenja, kar odpira vrata inovativnim rešitvam v različnih aplikacijskih domenah. S tem članek spodbuja nadaljnje raziskave in razvoj za praktično demonstracijo metod globokega učenja v aplikativnih domenah.

Zahvala

To delo je bilo delno podprto s strani raziskovalnih projektov ARIS J2-3169 (MV4.0) in J2-4457 (RTFM), kot tudi s strani raziskovalnega programa P2-0214.

Viri in literatura

He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask R-CNN. (str. 2961–2969). International Conference on Computer Vision.

Muhovič, J. N., Tabernik, D., & Skočaj, D. (2020). O klasifikaciji slik v ne-enolično določljive razrede. (str. 355-358). Zbornik devetindvajsete mednarodne Elektrotehniške in računalniške konference ERK 2020.

Tabernik, D., & Skočaj, D. (2019). Deep Learning for Large-Scale Traffic-Sign Detection and Recognition. *IEEE Transactions on Intelligent Transportation Systems*.

Tabernik, D., Muhovič, J. N., & Skočaj, D. (2023). Lokalizacija in ocenjevanje lege predmeta v treh prostostnih. *Zbornik dvaintridesete mednarodne Elektrotehniške in računalniške konference ERK 2023*.

Zavrtanik, V., Kristan, M., & Skočaj, D. (2021). DRAEM -- A discriminatively trained reconstruction embedding for surface anomaly detection. *International Conference on Computer Vision*.

Zavrtanik, V., Vodopivec, M., & Kristan, M. (2020). A segmentation-based approach for polyp counting in the wild. *Engineering Applications of Artificial Intelligence*.