Fakulteta za matematiko, naravoslovje
in informacijske tehnologije

Računalništvo in informatika

Projektni seminar

Aplikacija za prepoznavanje licitacije pri bridžu

Uporabniška dokumentacija

Avtorji:
Dani Zugan
Žan Peternelj
Jani Suban

Mentor:
dr. Peter Rogelj

16. september 2023

Kazalo

1 Uvod

Ta dokument je uporabniška dokumentacija aplikacije za prepoznavanje licitacije pri bridgeu. Aplikacija je uporabljena znotraj vmesnika z ukazno vrstico (ang. Command-line interface ali CLI). Aplikacija je sestavljena iz dveh delov: prepoznavanje licitacije ter izboljšava modela za prepoznavanje licitacije.

2 Namestitev

Aplikacijo enostavno namestimo tako, da ustvarimo novo mapo, kjer bo shranjena aplikacija. Za tem se prenese .zip datoteko iz spletnega naslova https://www.student.famnit.upr.si/~89222015/Bridge/projekt.zip. Datoteko se nato ekstrahira in aplikacija je pripravljena za uporabo, če imamo že nameščene vse potrebne knjižnice.

2.1 Zahtevane knjižnice

Preden začnemo uporabljati aplikacijo je potrebno še namestiti vse knjižnice, ki so potrebne za uporabljanje aplikacije. Preden začnemo z nameščanjem knjižnic, ki so potreben za uporabo aplikacije, je potrebno preveriti če imamo nameščen Python. To storimo, tako da poženemo sledeči ukaz v terminalu/PowerShellu/CMD (Ukazni poziv):

```
python —version //python –V
```

Če ukaz vrne napako ali različica Pythona je starejša od 3.8, je potrebno na novo namestiti Python. Ko imamo nameščeno pravilno različico Pythona lahko namestimo knjižnice, ki so potrebne za pravilno delovanje aplikacije. Knjižnice, ki jih aplikacija potrebuje za delovanje so sledeče: OpenCV, Ultralytics in PyTorch. Knjižnice namestimo s sledečim ukazom:

```
pip install opencv–python>=4.6.0 ultralytics torch>=1.8.0
```

S tem korakom smo uspešno zaključili z namestitvijo aplikacije na računalnik in jo lahko začnemo uporabljati.

3 Uporaba

Aplikacija uporabniku omogoča dva načina uporabe. Prvi način uporabe je prepoznavanje licitacije igre. Drugi način uporabe pa je izboljšava modela za prepoznavanje licitacije. V nadaljevanju tega poglavja bosta prestavljena oba načina uporabe. Vsi predstavljeni ukazi predpostavljajo, da je trenutna lokacija uporabnika mapa, v kateri je aplikacija.

3.1 Prepoznavanje licitacije

Prvi način upore aplikacije je prepoznavanje licitacije igre. Ta način uporabe aplikacije je primarni način, katerega uporabnik bo uporabljal večino časa. Aplikacija se požene s sledečim ukazom:

```
./main.py [ime_slike] [st_mize]
```

Aplikacija izdela .pbn dokument, v katerem je zabeležena licitacija igre. Pri tem je potrebno pbn datoteko popraviti, saj so imena igralcev napačna. Druga omejitev aplikacije je orientacija vhodne slike. Aplikacija predpostavlja, da so karte igralca na poziciji Sever (North ali N) so na vrhu slike ter karte igralca na poziciji Vzhod (East ali E) so na desni strani slike.

3.2 Izboljšava modela

Drugi način uporabe aplikacije je izboljšava modela za prepoznavanje kart, ki so uporabljene za licitacijo. Ta del aplikacije uporabimo za boljše prepoznavanje kart, ki jih uporabljamo pri licitiranju.

Prvi korak pri izboljšavi modela je zajem video posnetkov vseh kart za licitacijo. Videoposnetek zajamemo tako, da postavimo telefon ali video kamero nad karto ter zajamemo par sekund dolg posnetek. To storimo za vsako karto, pri tem pa telefon ali video kamero ne premikamo.

Enkrat, ko imamo vse video posnetke vseh kart, lahko iz video posnetka izrežemo, slike karte iz vseh sličic video posnetka. To storimo s sledečim ukazom:

```
./image_extractor.py [video_posnetek] [ime_karte]
```

Ime karte se mora držati sledečih pravil:

1. vsa imena kart morajo biti zapisana z velikimi črkami,
2. križ se označi s4 C, srce se označi s H, karo se označi z D, pik se označi s S in NT se označi z NT,
3. karte, ki vsebujejo številke, se pišejo skupaj.

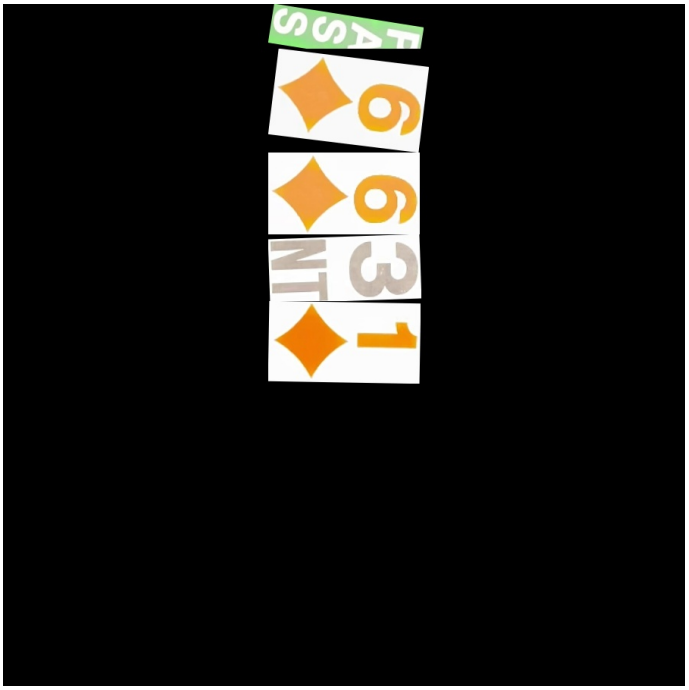
Torej na primer karta 6 karo, se zapiše kot D6. Ko ukaz poženemo se prikaže okno s prvo sličico videoposnetka. Na sličici izberemo sredino oznake na karti z dvojnim levim klikom na miški. To je potrebno storiti za vse karte.

Naslednji korak je izdelava učne in testne množice. To storimo na tak način, da se združi 5 različnih sik kart v kolaž, ki bo uporabljen za učenje in validacijo nevronske mreže. To se stori s sledečim ukazom:

```
./random_card_collage.py
```

Ta ukaz izdela 10000 kolažev kart ter labele za vsako karto na vsakem kolažu. Primer izdelanega kolaža je viden na Sliki 1. Preden nadaljujemo z deljenjem slik kolaža na validacisko in učno množico je potrebno pretvoriti trenutne labele kart v obliko, ki jo uporablja YOLO arhitektura nevronske mreže. To storimo s sledečim ukazom:

```
./CSV_to_YOLO.py
```



Slika 1: Primer kolaža

Za tem, ko imamo pretvorjene labele v YOLO obliko, moramo ločiti labele od slik. To storimo, da slike premaknemo v mapo images, s sledečim ukazom:

```
mv ./Output/*.jpg ./Output/images/
```

in labele premaknemo v mapo labels, s sledečim ukazom

```
mv ./Output/*.txt ./Output/labels/
```

To deluje le na računalnikih, ki uporabljajo Linux operacijski sistem. Če uporabljamo računalnikom z Windowsom pa uporabimo sledeči ukaz, da premaknemo slike:

```
move ./Output/*.jpg ./Output/images/
```

in sledeči ukaz, da premaknemo labele:

```
move ./Output/*.txt ./Output/labels/
```

Zdaj imamo pravilno ločene labele od slik, kot to zahteva YOLO arhitektura nevronskih mrež. Ampak za boljše učenje nevronske mreže je potrebno razdeliti podatke (slike in labele) še v učno in validacijsko množico. To je storjeno s sledečim ukazom:

```
./splitter.py
```

Ukaz razdeli množico kolažev v razmerju 7:3, kjer je 70% kolažev uporabljenih za učenje nevronske mreže in 30% za validacijo nevronske mreže.

Enkrat, ko imamo izdelano testno in validacijsko množico, lahko začnemo s ponovnim učenjem modela z novimi vhodnimi podatki. Na napravah, ki uporabljajo Linux kot operacijski sistem, je to storjeno s sledečim ukazom:

```
nice nohup yolo detect train data=coco128.yaml //
model=./Model/best.pt epochs=10 imgsz=900 lr0=0.01 &
```

Če pa uporabljamo računalnik, ki uporablja operacijski sistem Windows, program poženemo s sledečim ukazom:

```
start /min /high yolo detect train data=coco128.yaml^
model=./Model/best.pt epochs=10 imgsz=900 lr0=0.01
```

Slika 2: Rezultati učenja modela

Edine parametre, ki jih lahko uporabnik spreminja, so epochs in lr0. Število epoch pove, kolikokrat se ponovi proces učenja. Drugi parameter, ki ga lahko uporabnik nadzoruje, je lr0, ki omogoča uporabniku določati hitrost učenja nevronske mreže. Pri tem odsvetujemo, da uporabniki spreminjajo hitrost učenja.

Ko se ponovno učenje nevronske mreže konča, so vsi modeli izdelani po vsakem epochu shranjeni v svoji mapi train, kot je to razvidno iz Slike 2. Za pravilno prepoznavanje kart je potrebno premakniti zadnjo izdelano različico modela, premakniti v mapo Model. Na Linux računalniku je to storjeno s sledečim ukazom:

```
cp -f ./runs/detect/[zadnja_razlicica]/weights/best.pt ./Model/
```

Na Windows računalniku pa to naredimo s sledečim ukazom:

```
copy /y ./runs/detect/[zadnja_razlicica]/weights/best.pt ./Model/
```