

# Препознавање ручно написаног текста (Book Archiver)

Извештај за практично истраживање

# Увод

## Предмет истраживања

Предмет овог пројекта се заснива на испитивању шта је све потребно да би раучар препознао ручно написан текст са слике на папиру без коришћења неких већ готових библиотека које своје решење нуде у пар линија кода (нпр. *easyocr*).

## Циљеви истраживања

Сам циљ пројекта јесте архивирање ручно написаних књига у неки текстуални документ зарад презервације истих.

## Задаци истраживања

Можемо задатак за имплементирање оваквог програма поделити у две обимне целине.

Прва целина представља само тренирање довољно добре конволуционе неуронске мреже за препознавање ручно написаног тескста, док другу целину представља издвајање индивидуалних карактера са слике и прослеђивање тих слика истренираној мрежи.

## Очекивани резултати истраживања

Због недостатка рачунарске снаге које поседујемо не очекујемо да ћемо имати неке довољно успешне резултате које задовољавају стандарне критеријуме.

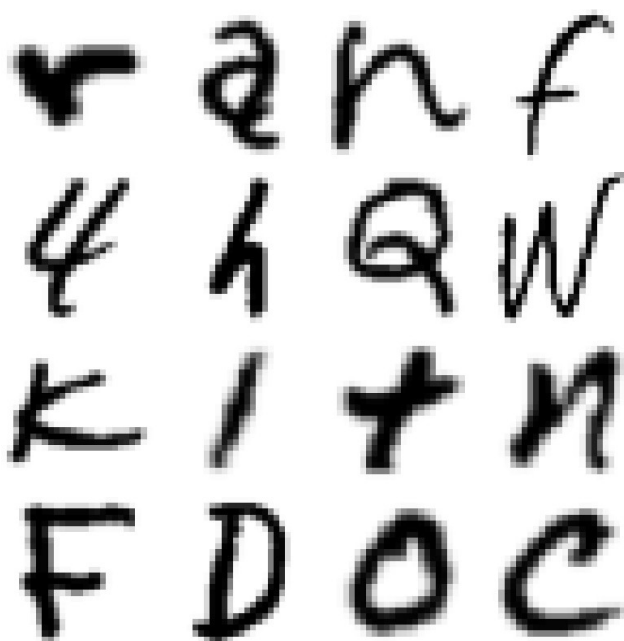
# Методологија

## Коришћени подаци

Користимо један од познатијих тренинг скупова под именом EMNIST (Extended MNIST).

Његов претходник MNIST је био сачињен само од ручно написаних цифара, док је EMNIST сачињен од цифара, малих и великих слова. Сlike су црно-беле и у формату 28\*28 пиксела.

5H3:PHEE2W3JhTF#P3ZKZQHt5PWNMY9ti  
udkkewfY2YcD6KV6G872+wdhD8QV954A  
6R73HARZd407D5rHK9Yx252urdyfA8  
B2B7Cf42KP2451UJj7269KQ4nJVOXHVC  
B6WU0AR2QE0+02CE0WfP0r810CXtncd3  
293CH599e23diTSUBKEC907A=8GXaAc3  
T4451IBU7Edeaf4tG8YKreVL6K9Dncet  
bTG4U0H8NFU3Wn63P097BEP196XG6nW6  
8P2Q1102Hd3rZmk91PmbP2SEZ9K1f4/F  
4u4E41mf83n+7E8S1eK495CKRf3G7Ht3  
1rP058N4Ynf+5D2QBpV0fGfBdtYXJawJ  
P29322ECU22NEfz200A.YAQ9cA1T07Nmc  
C04G8htE07Ar+8dkP89HrVGAanBV12eV  
OXGBBG1UW8A4EXd3911GBfRfZATMHZb9  
P1MvW49T94641S61H0794u25f1k1dUX  
CAF49Vm11009KRCdPAQ9A6arf14C81f  
E4AQ16#e1JX2rR0PE9XG0U44Tf358atY  
TM9CCe3M9Yw6BR056388f9b2210D0JQF  
8529h4Vf9N4fE5NGf241fde9n105fkN6  
C1fX8V1B0t2QKNV/KC9HH5822C0f9022  
1n1FGn59r7213JhA5C2G536TN7SEZUV6  
r9B8CAf0739090f80BYQ8m2+E=Chw99m  
9aFneNALW9W5+9NfSV9NR3KP9571X209  
K2CDmVH4r7BB946BCb361/d51BCat1h  
Z6rtn5730QDI9FVCTUS3DY26ALk31A71  
ntf87W9aet199+nmvB9KEfD956H1Qa2  
54rEm2J16wH761w6+787rJn6w70070h1  
df7G06Tn9796TAM+r04A8w81H1Q7XR5n  
duE20WEPX07Yf9h7SDNRdMW17CwGm1n  
BmXQ0H4FEF/Hm0KvUWwGp1F00XwfX1e  
4Q10FefUAUwe13XyXLtQ53P585h7rICH  
KJ171e57BLC2+ET1G8hU83B026nve1hLH



Сами подаци су подељени по категоријама: ByClass, ByMerge, Balanced, Letters, Digits и MNIST

Такође је свака категорија подељена на тренинг и тест скупове.

Назив категорије	Укупан број слика	Тренинг (train)	Тест (test)	Број класа (карактера)
EMNIST ByClass	814 255	697 932	116 323	62
EMNIST ByMerge	814 255	697 932	116 323	47
EMNIST Balanced	131 600	112 800	18 800	47
EMNIST Letters	145 600	124 800	20 800	26
EMNIST Digits	280 000	240 000	40 000	10
EMNIST MNIST	70 000	60 000	10 000	10

Пре коришћења самог скупа проверамамо да ли постоје дупликати.

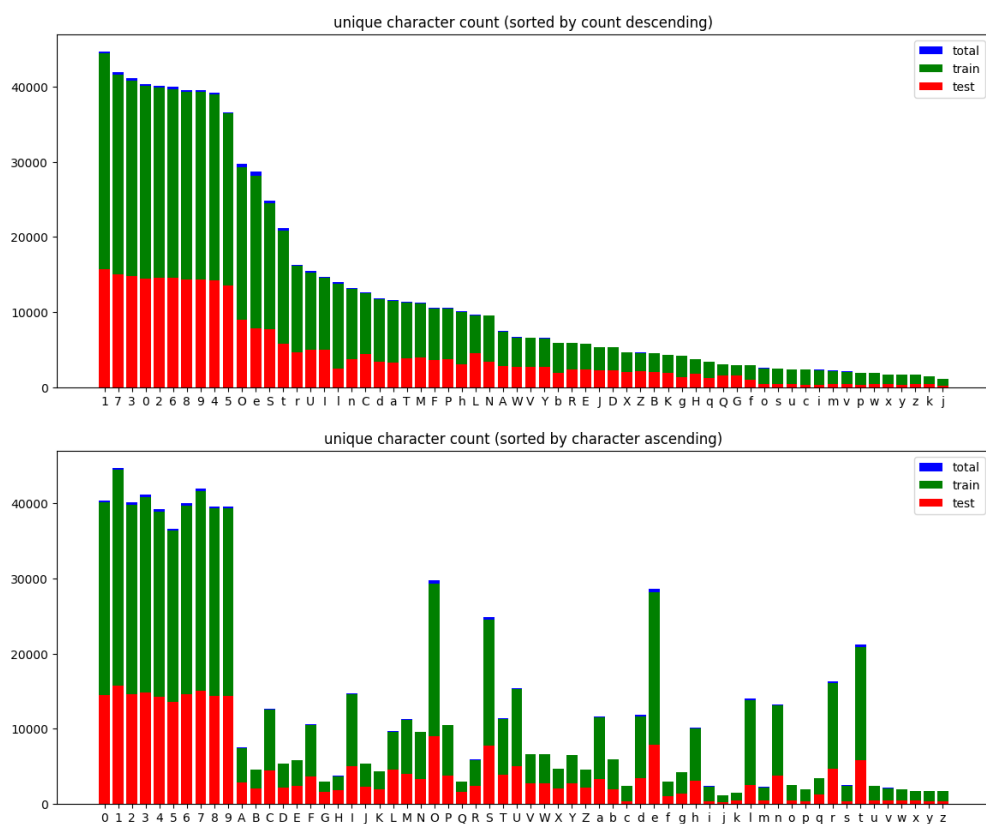
Дупликате проналазимо тако што израчунавамо MD5 хеш за сваку индивидуалну слику, затим мапирамо хеш на индекс слике у сету који касније поредимо са следећим хешом који рачунамо. Ако израчунамо хеш који смо претходно израчунали наишли смо на дупликат.

Анализом сваке од категорија, утврђено је да постоје дупликати у тренинг (*train*) сетовима:

Назив категорије	Број дупликата
EMNIST ByClass	9
EMNIST ByMerge	9
EMNIST Letters	5

Даљом анализом је утврђено да скоро све категорије имају неке слике која нека друга категорија има. На пример ByClass и ByMerge имају велики број слика које су исте. ByClass *train* скуп садржи слике које ByMerge користи у свом *test* скупу, и обрнуто.

Консултовати следећи графикон:



**total** – Приказује колико укупно има јединствених слика у свим скуповима (*train* и *test*).

**train** – Приказује колико укупно има јединствених слика у *train* скуповима.

**test** – Приказује колико укупно има јединствених слика у *test* скуповима.

Учитавањем и избацивањем свих дупликата утврђено је да постоји само 814 241 јединствених слика.

Због међусобног преплитања свих категорија потребно је било самостално поделити податке на тренинг и тест скупове за сваки карактер индивидуално, за сваки карактер је узето: **80%** слика за **тренинг**, а **20%** за **тестирање**.

(консултовати следећу скрипту: [proper\\_train\\_test\\_split.py](#)).

## Претходна истраживања других особа над коришћеним подацима

Анализом рада других који су користили дати скуп нису испитивали податке. Такође већински су сви тренирали мреже на смањеној верзији сета MNIST која садржи само бројеве од 0-10 што је олакшало саму класификацију за мрежу тако да су добијали веома добре резултате (прецизност до чак 90%). Али за наш скуп података EMNIST, нисмо успели да пронађемо рад који је имао успешност већи од 70% због саме комплексности класификације карактера који слично изгледају (попут: „i“, „j“, „1“ затим „S“, „5“, такође „0“, „O“, итд.)

## Методе истраживања

Тренирање саме мреже се сводило на испитивање разних параметра и слојева која библиотека *tensorflow* нуди приликом прављења конволуционе неуронске мреже.

Направљено је неколико верзија мреже коју тренирамо (консултовати датотеку: [utils\\_tf.py](#) у којој можемо видети сваку верзију).

Свака верзија је тренирана на претходним 80% издвојеним подацима и тестирано на преосталим 20%.

Поређењем резултата сваке од верзија мреже утврђено је да нам верзија три (v3) даје најбоље резултате који нам највише одговарају тако да је она коришћена за сва остала испитивања.

Главне параметре које смо узимали у обзир током тренирања мреже су величина уносног тренинг беча (*batch\_size*) и број епоха (*epochs*). Испитивањем је закључено да конволуциона мрежа која има *batch\_size* 1000 или 10000 даје боље резултате од оне која има мањи.

Утврђено је да библиотека *tensorflow* за велике *batch\_size*-еве користи разне методе које смањују буку приликом израчунавања градијента што доприноси самој класификацији категорије. Такође је вођено рачуна да немамо превелики број епоха да не би дошло до превеликог прилагођавања подацима (*overfitting*). Пошто имамо велики број слика за тренирање, 4 или 5 епоха је било сасвим довољно за тренирање довољно добре мреже.

Касније је примећено да је број слика за карактере као што су „j“, „k“, „z“ врло мали. Па смо покушали да допунимо слике коришћењем *tensorflow*-овог *ImageDataGenerator*-а који дату слику ротира, зумира и транслира на разне начине и тако допуњава недостајуће слике.

Нажалост због недостатка рачунарске снаге смо избацили ту методу из оптицаја јер се мрежа јако дуго тренирала. Такође нисмо желели да то допуњавање података заправо одмани учење мреже јер можда буде превише *overfit-овао* на генерисане слике, али не можемо тачно знати. Због довољне величине скупа података само генерисање слика заправо можда ни није било потребно.

Више од 1 дан је био уложен да се намести тренирање на графичкој картици (*tensorflow-cuda*), али нажалост то није било успешно.

Након 2 дана уложених у тестирање параметара која tensorflow библиотека нуди. Одлучено је да се направи окружење које ће кориснику ручно дати да цртањем карактера тестира истренирану мрежу. Тако да смо за финалну мрежу одлучили да користимо цео скуп података (и тест и тренинг податке које смо раније поделили).

У датом пројекту су направљена пар окружења за тестирање истренираних мрежа.

Покретањем сваког од окружења кориснику се нуди листа већ истренираних мрежа коју може изабрати да жели да тестира.

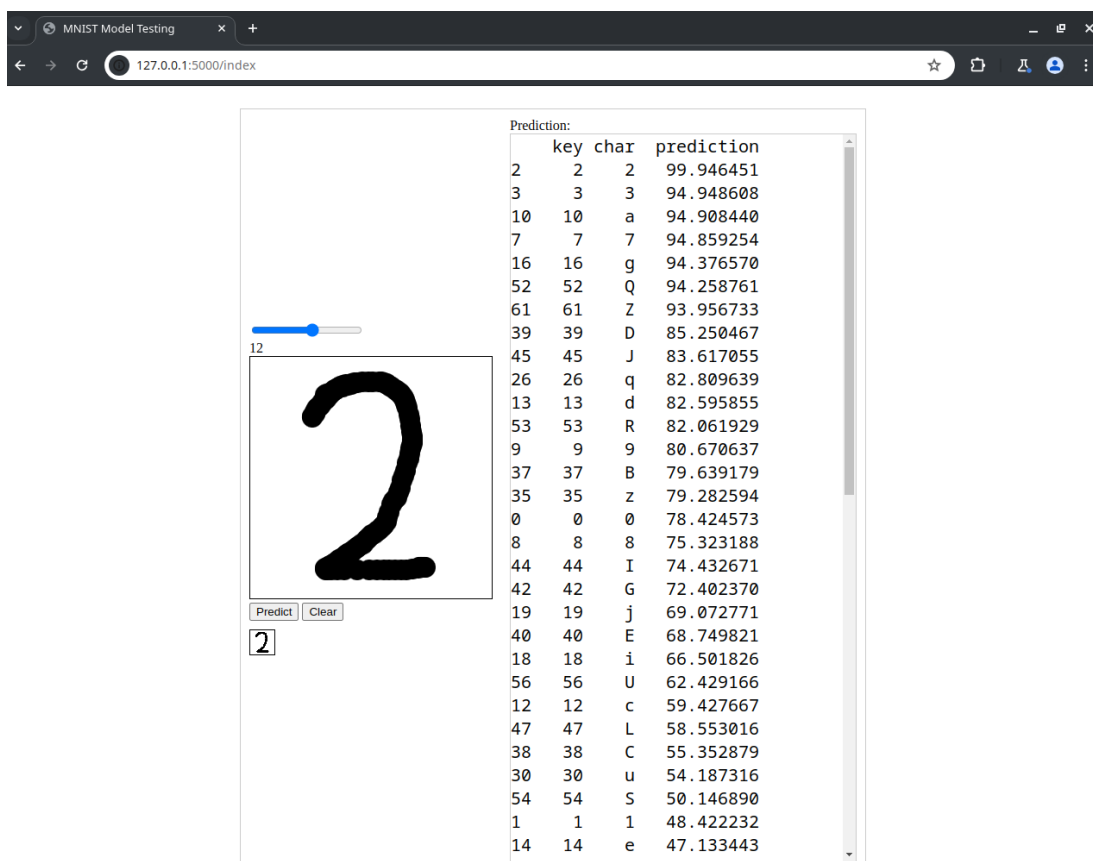
Тренутно најбоља мрежа је под називом: [all\\_v3\\_batch10000\\_epoch5.keras](#)

Окружење за цртање ([test\\_draw.sh](#)).

Након што корисник изабере мрежу покреће се веб сервер на адреси:

<http://127.0.0.1:5000>Након приступа датој веб страници кориснику је омогућено да ручно тестира мрежу цртајући.

Консултовати следећу слику:



Окружење чини једноставан *flask* сервер који нуди дату страницу за цртање.

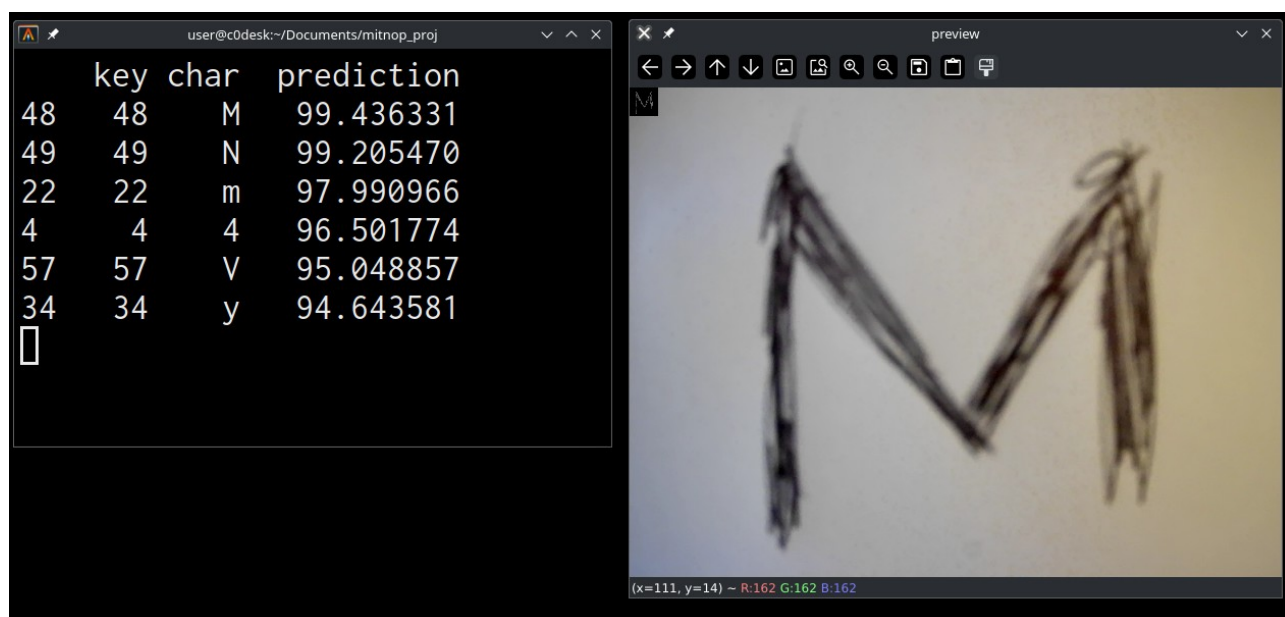
Страница затим шаље нацртану слику мрежи путем *POST request*-а, слика је енкодована у *base64* формату, *flask* сервер даље распакује слику и прослеђује је неурноској мрежи која врши предикцију, израчуната предикција се форматира у табеларни приказ коришћењем библиотеке *pandas* и шаље се назад страници да је може приказати.

За друго окружење је потребна камера. Тренутно постоје две верзије ([test\\_camera.sh](#) и [test\\_camera2.sh](#)).

Обе верзије су направљене коришћењем библиотеке *OpenCV*.

Прва верзија ради предикцију једног карактера са слике.

Консултовати следећу слику:



Функционише на једноставан начин тако што дату слику само *resize*-ује да буде формата 28\*28 пиксела, након тога примени праг (*threshold*) који избаци све светле пикселе из слике коришћењем *color* маске.

Следећа верзија ([test\\_camera2.sh](https://test_camera2.sh)) ради препознавање више карактера.



Доста коплексније функционише, детекција карактера се спроводи коришћењем следећих метода: Прво слику претварамо у црно белу (примењује се *GrayScale* филтер). Затим на ту слику додајемо *GaussianBlur* филтер да би смањили буку у слици. На такву слику примењујемо такозвани *Canny edge* детекцију из које можемо извући контуре (облике). Детековане облике спроводиму неуронској мрежи која ради предикцију.



# Резултати

## Приказ резултата

Најбољу мрежу коју смо успели истренирати је мрежа под називом:

[all\\_v3\\_batch10000\\_epoch5.keras](#)

Резултати се најбоље могу видети покретањем неких од претходно наведених окружења (demo-a).

## Тумачење резултата

У претходним испитивањима утврђено је да се мрежа најчешће збунује за карактере:

„0“, „o“, „O“

„L“, „l“, „1“, „i“, „I“, „J“, „j“

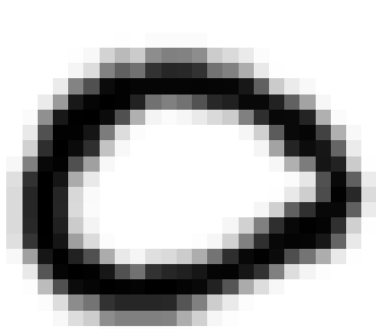
„S“, „5“, „G“, „6“

„9“, „g“

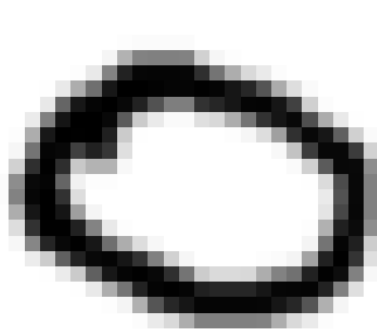
„a“, „2“

Неки од примера:

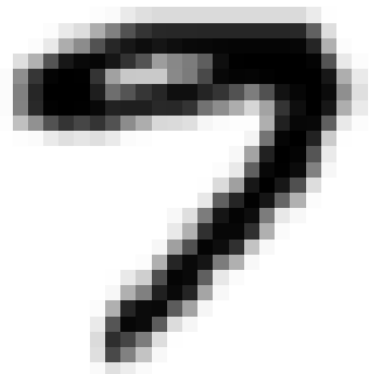
Example 19. Label: 0 -- predict: 0



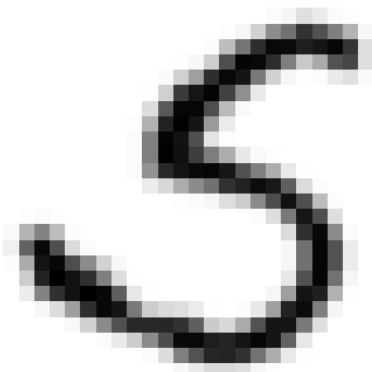
Example 149. Label: 0 -- predict: 0



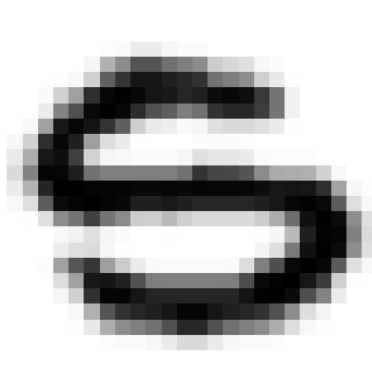
Example 226. Label: 9 -- predict: 7



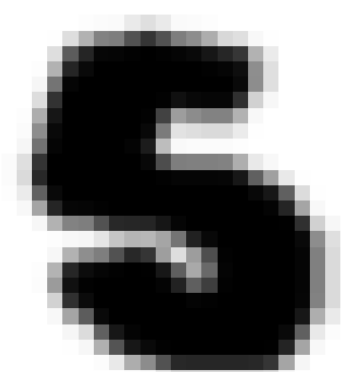
Example 35. Label: 5 -- predict: S



Example 20. Label: 5 -- predict: S



Example 419. Label: 5 -- predict: S



# Закључак

## Анализа испуњења циљева истраживања

Испитано је испуњење циља истраживања, и закључено је да је пројекат делом извршен, међутим, потребно је додатно улагање напора у развој технике детекције текста и подешавање параметара неуронске мреже како би се омогућило ефикасније откривање и препознавање текстуалних елемената.

## Анализа остварења очекиваних резултата истраживања

Резултати које смо добили су далеко од савршених. Но очекивали смо такве резултате јер на расплогању имамо слаб рачунар који није у могућности да користи *tensorflow*-ов *ImageDataGenerator* што би можда побољшало тренирање мреже.

## Могућности за примену истраживања у пракси

Прмену овог истраживања у пракси можемо видети у неком пројекту или индустрији за коју је детекција текста потребна. Као што су примери: препознавање рукописа у банкарским апликацијама, препознавање адреса на поштанским пакетима, и најважније за архивирање историјских књига и докумената.

## Идеје за побољшање и разраду истраживања

Прва идеја која пада на памет за побољшање наше неуронске мреже јесте проналазак већ готове неуронске („супер“) мреже која је истренирана на гомилу дата скупова са највише параметара коју можемо помоћу *transfer learning*-а интегрисати у наше решење.

Друга идеја је набавка јачег хардвеа за самостално тренирање такозване „супер“ мреже. Јачи хардвер би нам омогућио да користимо *tensorflow*-ов *ImageDataGenerator* без потешкоћа. Такође треба узети у обзир побољшање парметара за детекцију контура са слике када примењујемо *Canny edge* детекцију.

# Литература

- [https://www.tensorflow.org/api\\_docs/python/](https://www.tensorflow.org/api_docs/python/)
- [https://en.wikipedia.org/wiki/Canny\\_edge\\_detector](https://en.wikipedia.org/wiki/Canny_edge_detector)
- <https://docs.opencv.org/4.x/index.html>
- [https://docs.opencv.org/4.x/da/d22/tutorial\\_py\\_canny.html](https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html)
- <https://pandas.pydata.org/docs/>