Mihailo Vukorep (IN 40/2021)

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.

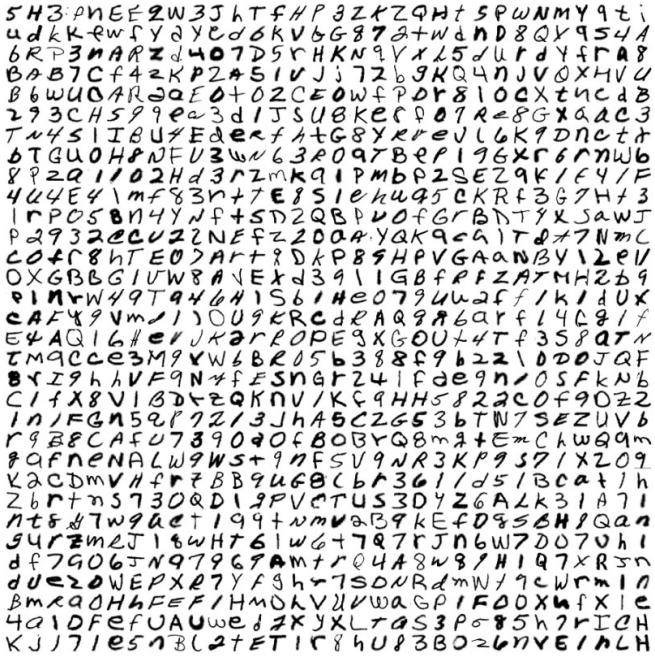
Очекивани резултати истраживања

Због недостатка рачунарске снаге не очекујемо да ћемо имати неке довољно успешне резултате који 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. Слике су црно-беле и у формату 28*28 пиксела.



Сами подаци су подељени на категорије: ByClass, ByMerge, Balanced, Letters, Digits и MNIST Такође, свака категорија је подељена на тренинг и тест скупове.

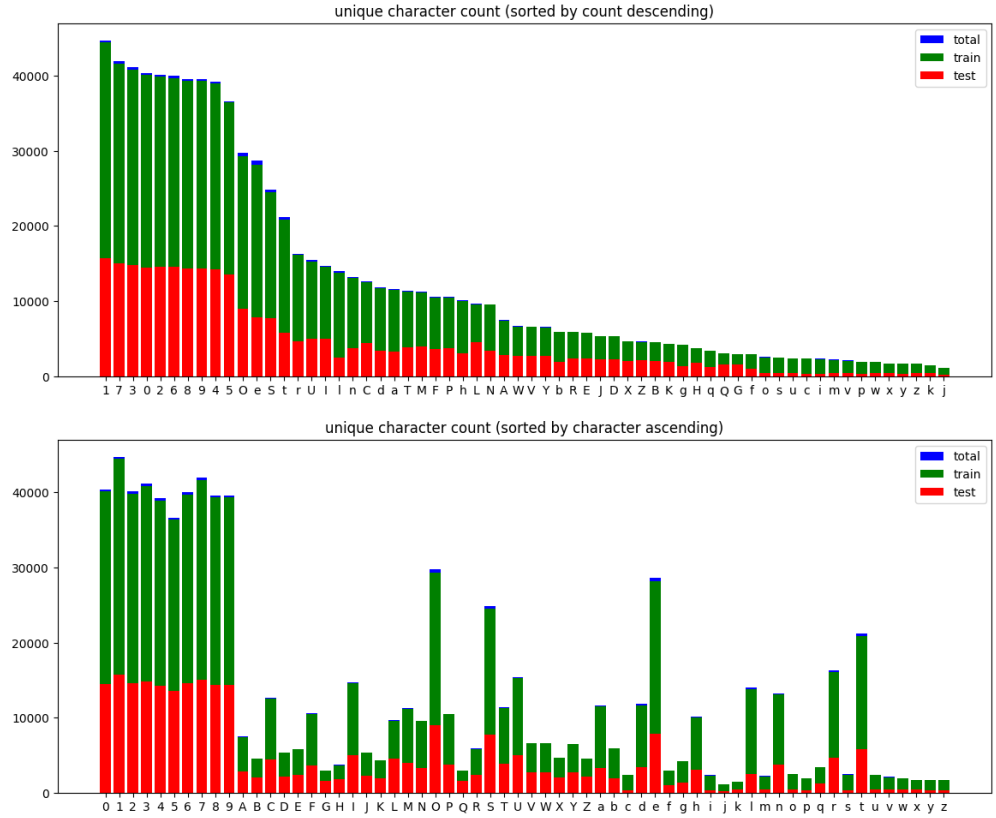
Назив категорије	Укупан број слика	Тренинр (train)	Тест (test)	Број класа (карактера)
EMNIST ByClass	814 255	697 932	116 323	62
EMNIST ByMerge	814 255	697 932	116 323	47
EMNIST Balanced	131 600	112 800	18 800	47
EMNIST Letters	145 600	124 800	20 800	26
EMNIST Digits	280 000	240 000	40 000	10
EMNIST MNIST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

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

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, видимо да они нису испитивали податке. Такође, већински су 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 већу од 70% због саме комплексности класификације карактера који слично изледају (попут: „i“, „j“, „1“; затим „S“, „5“; или „0“, „O“)

Методе истраживања

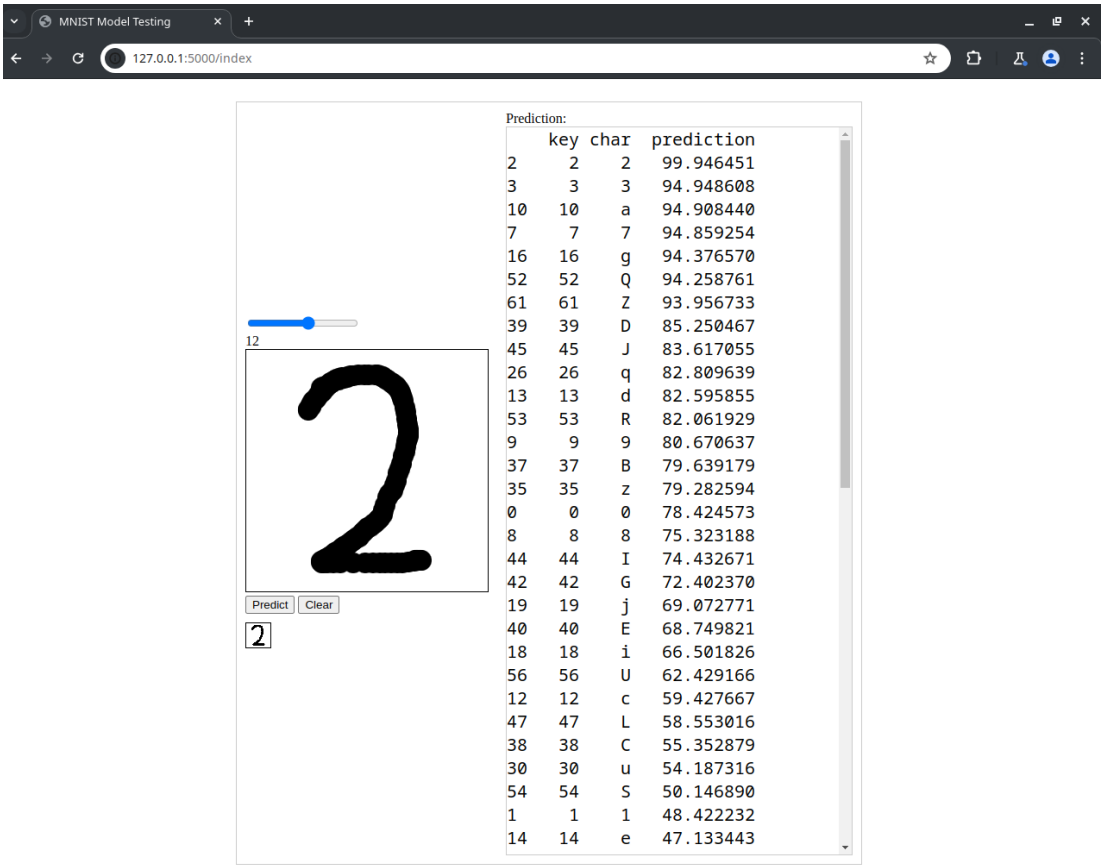
Тренирање саме мреже се сводило на испитивање разних параметара и слојева које библиотека *tensorflow* нуди приликом прављења конволуционе неуронске мреже.
Напрвљено је неколико верзија мреже коју тренирамо (консултовати датотеку: *utils_tf.py* у којој можемо видети сваку верзију).
Свака верзија је тренирана на претнходних 80% издвојених података и тестирана на преосталих 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 10 000 даје боље резултате од оне која има мањи.
Утвеђено 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 поремети 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 окружења за тестирање истренираних мрежа. Покретањем сваког од њих кориснику се нуди листа већ истренираних мрежа које може изабрати 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, где је кориснику омогућено да ручно тестира мрежу цртајући на платну.

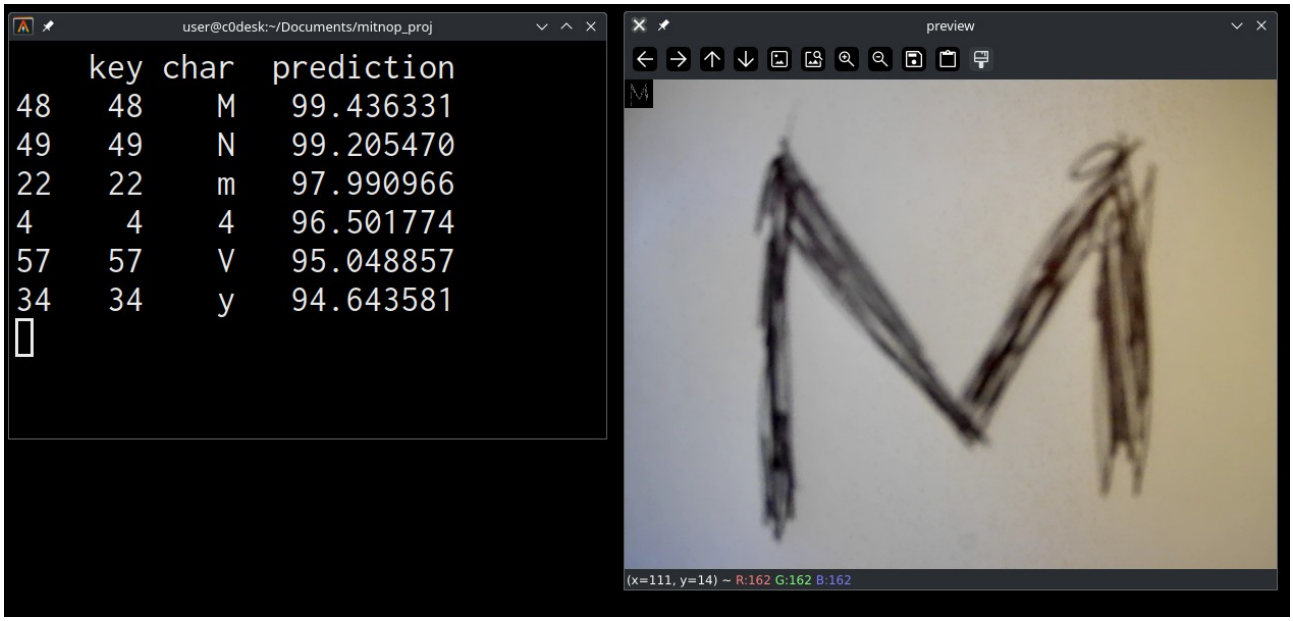
Консултовати следећу слику:



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*** ***test_camera2.sh***).
Обе верзије су направљене коришћењем библиотеке *OpenCV*. Прва 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***) ради препознавање више карактера.

Доста коплексније функционише јер се детекција карактера 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 спроводимо неуронској мрежи која ради предикцију.

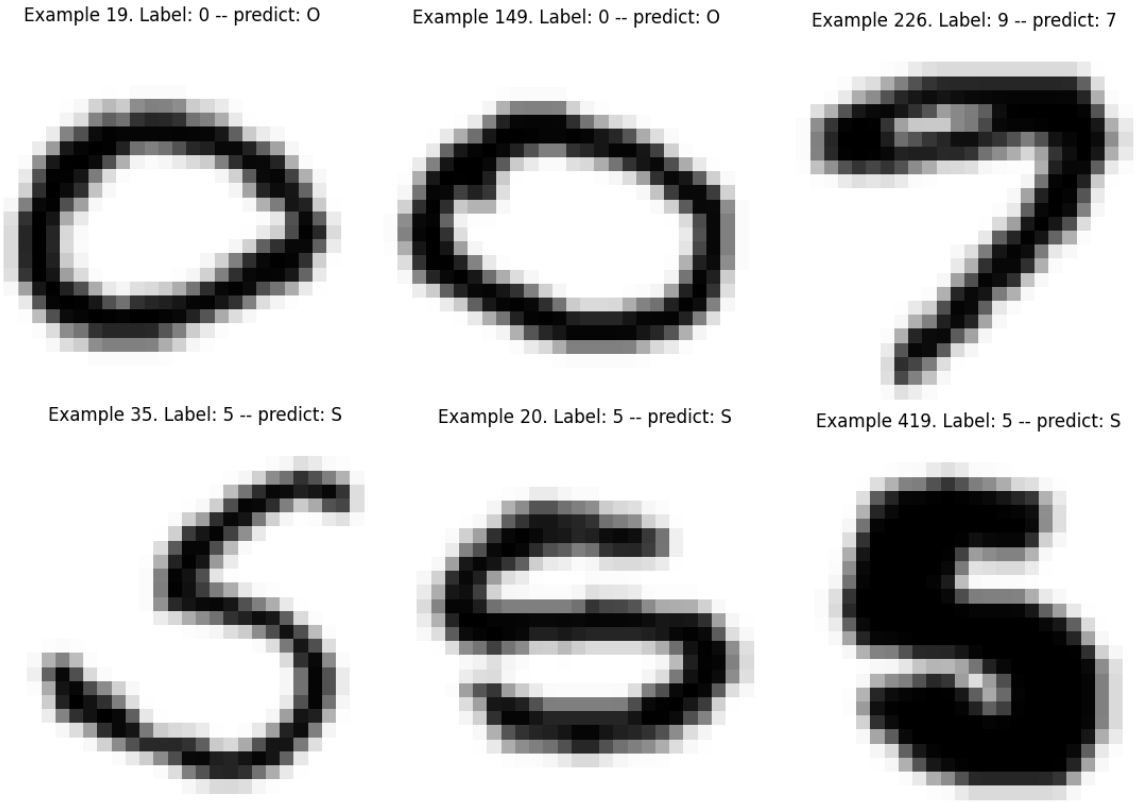
Резултати

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 претнходним испитивањима утврђено је да се мрежа најчешће збуни за карактере:
„0“, „o“, „O“
„L, „l“, „1“, „i“, „I“, „J“, „j“
„S“, „5“, „G“, „6“
„9“, „g“
„a“, „2“



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 расплоагању имамо слаб рачунар који није у могућности да користи *tensorflow*-ов ImageDataGenerator што би можда побошљшало тренирање мреже.

Могућности за примену истраживања у пракси

Примену овог истраживања у пракси можемо видети у неком пројекту или индустрији за коју је детекција текста потребна. Као што су примери: препознавање рукописа у банскарскип 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 learninga*-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://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://pandas.pydata.org/docs/
- https://flask.palletsprojects.com/en/3.0.x/