Дигитално процесирање на слика

Скенирање на баркод во слика

Давид Ангеловски - 221090

Вовед

Компјутерската визија е навистина моќна алатка. Без неа, извлекувањето на информации од слика е спор, тежок процес со неконзистентен квалитет зависен на човекова компетентност. Од друга страна, компјутерската способност да обработува и разбира визуелни податоци овозможува брзи и точни одлуки, значително подобрувајќи го ефикасноста и прецизноста на различни процеси.

Во рамките на овој проект, ќе ги искористиме овие технологии за развој на модел за препознавање и скенирање на баркод од слика, демонстрирајќи ја нивната моќ и применливост во реални сценарија.

Главните предизвици за да ја исполниме нашата цел се:

-Наоѓање на баркод во рамките на дадена слика.

-Дешифрирање на истиот за да извадиме некакво значење.

Понатаму ќе дискутираме неколку можни солуции за овие две проблеми.

Баркод

Пред да навлеземе во самите алгоритми сметам дека е важно да се разјасни точно што е баркод и како тој функционира. Баркодот е визуелен, машински читлив претставник на податоци кој обично содржи информации за производ или објект. Составен од серија на паралелни вертикални линии со различна дебелина и разлики помеѓу нив. Особено ни е важен фактот дека баркодот има конзистентен формат(сепак постојат разни стандарни но генерално имаат ист изглед) коишто можеме на повеќе начини да го пронајдеме во слика.

Пронаоѓање баркод во слика

Метод1 - OpenCV

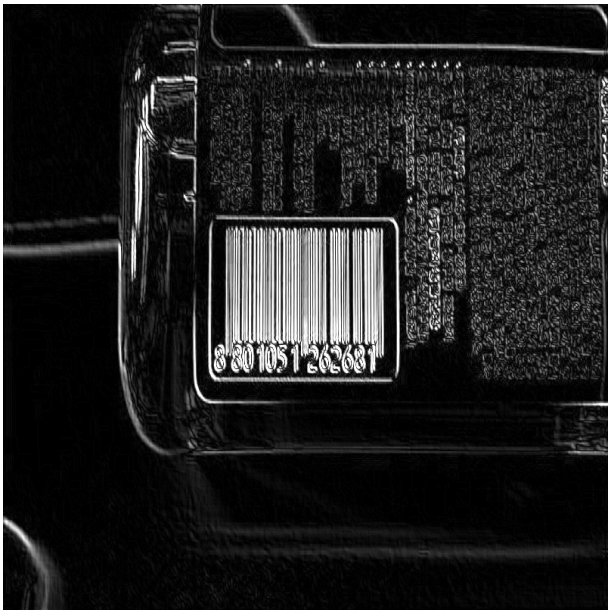
Првиот метод за наоѓање баркод во рамки на една слика се заснова на фактот што, во повеќето случаи, баркодот е во голем контраст со остатокот на сликата. Ова значи дека ние можеме да го истакнеме баркодот користејќи ги основните техники за процесирање на слики кои ни ги дозволува OpenCV библиотеката.

Ќе ја искористиме сликата долу како пример за влез за овој модел, прикажувајќи секој чекор од трансформацијата

Сл. 1: Пример влез слика со баркод

Во првиот чекор сликата е претворена во grayscale за поедноставно процесирање и ги извлекуваме хоризонталните и вертикалните градиенти од сликата користејќи го Sobel операторот. Потоа вертикалниот градиент го одземаме од хоризонталниот и ја наоѓаме апсолутната вредност на резултатот.

Оваа трансформација ја правиме за да се потенцира големиот хоризонтален контраст во баркодот. Бидејќи баркодот содржи наизменични црни и бели линии ова значи дека хоризонталниот градиент ќе ги истакне. Вертикалниот градиент е одземан за да ни остане почиста слика со многу малку шум



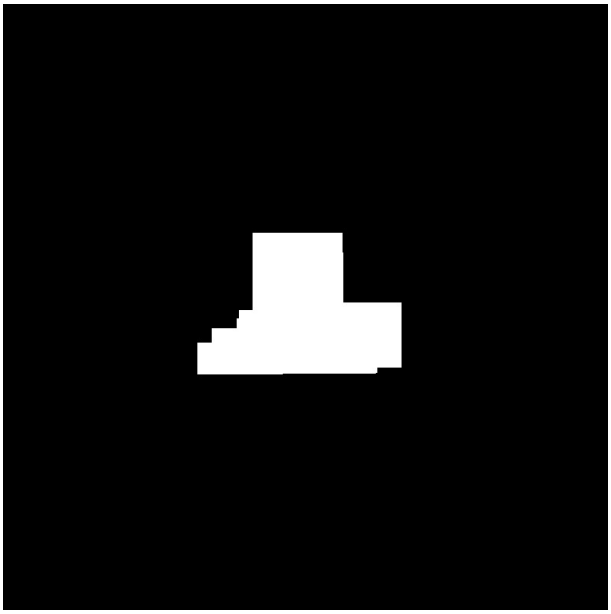
Сл.2: Резултат од наоѓање и одземање на градиенти

Следно ќе извршиме дополнително чистење на шум и други нерелевантни детали користејќи Гаусово замаглување. Потоа поставуваме праг којшто секој пискел со вредност помал од 200 ќе му даде вредност 0, односно ќе стане црн

Оваа трансформација истовремено го нагласува контрастот на баркодот и елиминира други потемни делови од сликата кои не ни се од корист

Сл.3: Резултат на замаглување и праг

Баркодот го конвертираме во една континуирана форма користејќи мофролошко затворање. Исто така ќе извршиме неколку итерации на ерозија и дилација за да елиминираме останати помали форми добиени по затварањето.

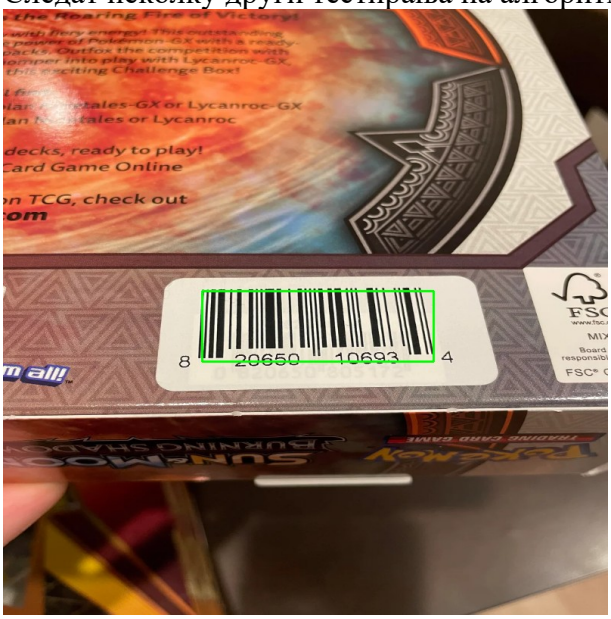


Сл.4: Резултат на морфолошко затворање, ерозија и дилација

Конечно останува само да ги пронајдеме контурите во сликата и да ја извлечеме најголемата, којашто е всушност посакуваниот баркод, и да нацртаме правоаголник (bounding box) околу баркодот користејќи ги излезните координати, висина и ширина.

Сл.5: Пронајден баркод

Следат неколку други тестирања на алгоритмот со други слики земени како влез

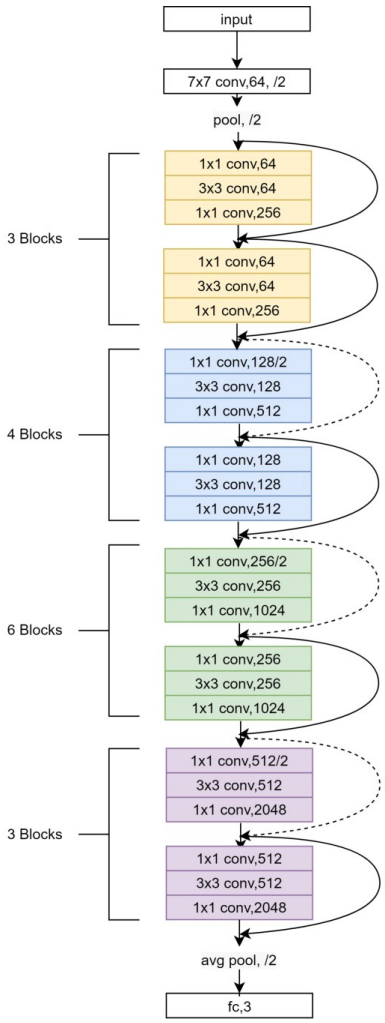


Сл. 6, 7, 8: Тестирање на метод 1

Метод 2 – ResNet

Овој метод, како и следниот, се направени користејќи библиотеката за машинско учење Keras и со тоа се доста покомплексни, поточно се засноваат на концептот на конволуциски невронски мрежи.

Длабоките конволуциски невронски мрежи се одлична алатка за тренирање модели за разни задачи во врска со компјутерската визија. Сепак, имаат мана со тоа што како што стануваат подлабоки мрежите почнува да се губи градиентот на сликите. Затоа била измислена ResNet (Residual Network) мрежата која го избегнува овој проблем со тоа што во неа содржи така наречени “Преостанати Слоеви” (Residual Layers) кои ѝ дозволуваат на мрежата да „прескокнува“ слоеви за време на обуката со тоа што нивниот излез F(x) е поставен на иста вредност со влезот X за да се избегне непотребната трансформација и задржи градиентот. Конкретно во рамки на овој проект ќе се користи ResNet50 мрежата која е длабока 50 слоеви.



Сл. 9: Структура на ResNet50

Архитектурата на нашиот конкретен модел е:

-ResNet50(weights='imagenet', include_top=False, input_shape=(224, 224, 3))

-GlobalAveragePooling2D
-Dense(1024,relu)

-Dropout(0.3)

-Dense(4,sigmoid)

Првиот слој е самата предтренирана мрежа ResNet50, таа прифаќа слики со димензии од 224x224 со 3 канали за боја. Ги користиме тежините добиени од ImageNet базата, врз која била тренирана ResNet мрежата, и параметарот “include_top” го поставуваме на “False” за да можеме да го изоставиме крајниот слој од иницијалната мрежа. Ова е направено бидејќи ResNet е оригинално наменето за класификација на слики но ние ќе го користиме за екстракција на карактеристики (feature extraction) односно пронаоѓање на баркод, па затоа не ни се потребни излезните слоеви.

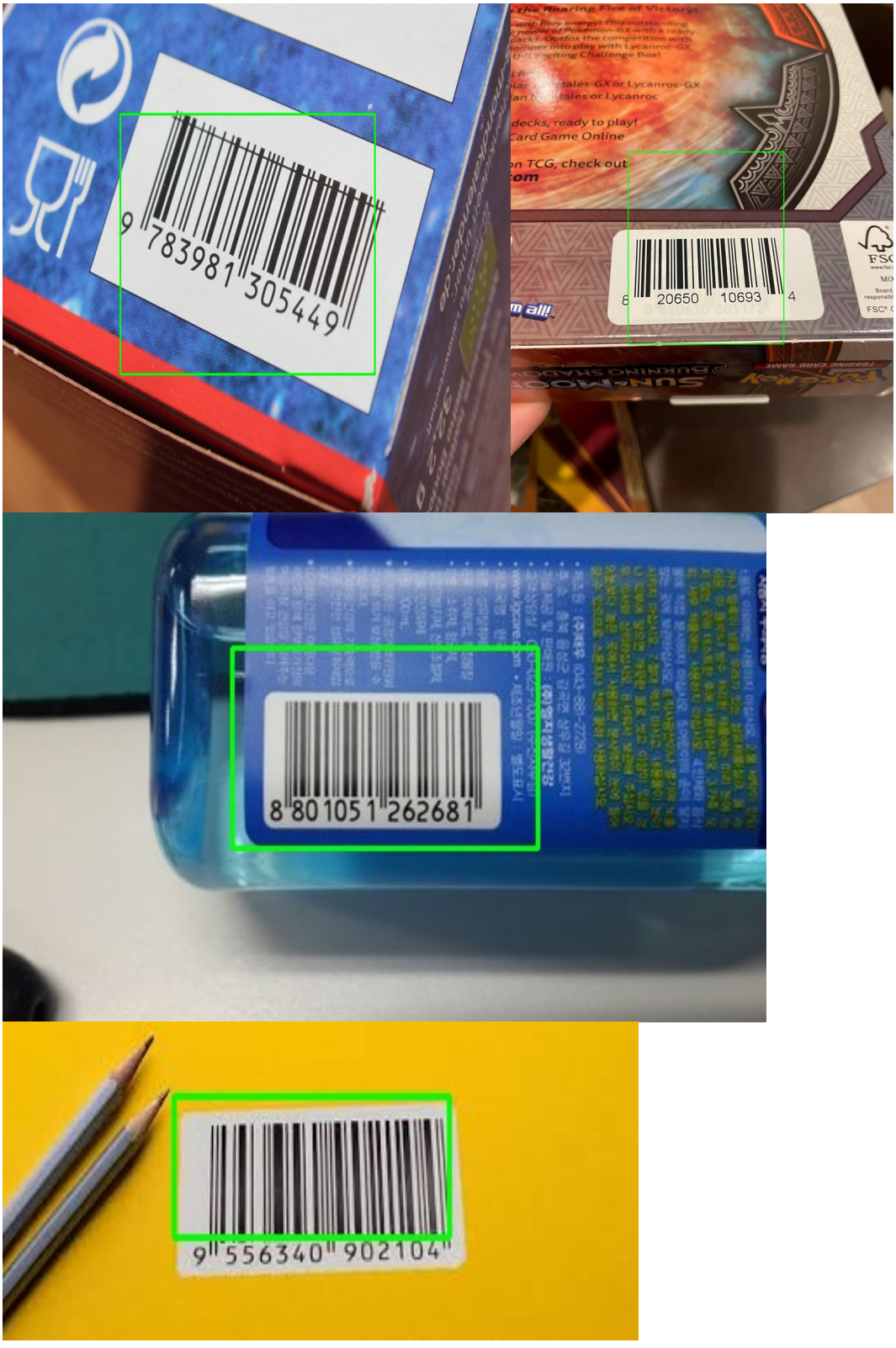
Потоа следните слоеви се дополнителни нагодувања за да можеме да го пронајдеме баркодот во сликата. GlobalAveragePooling2D ја намалува димензионалноста на параметрите за секој канал. Првиот Dense слој се користи за да се научат врските помеѓу вредностите, се користи relu за да се додаде нелинеарност. Dropout служи за да се избегне префитување (overfitting). И вториот Dense слој конечно ни ги дава четирите точки кои ни требаат за да го нацртаме нашиот Bounding Box, sigmoid активациската функција се користи за да се осигура дека вредностите се нормализирани (помеѓу 0 и 1) за да се правилно нацрта Bounding Box.

Истотака слоевите на иницијалната ResNet50 мрежа која ја дополнуваме остануваат непроменети бидејќи веќе се соодветно тренирани однапред.

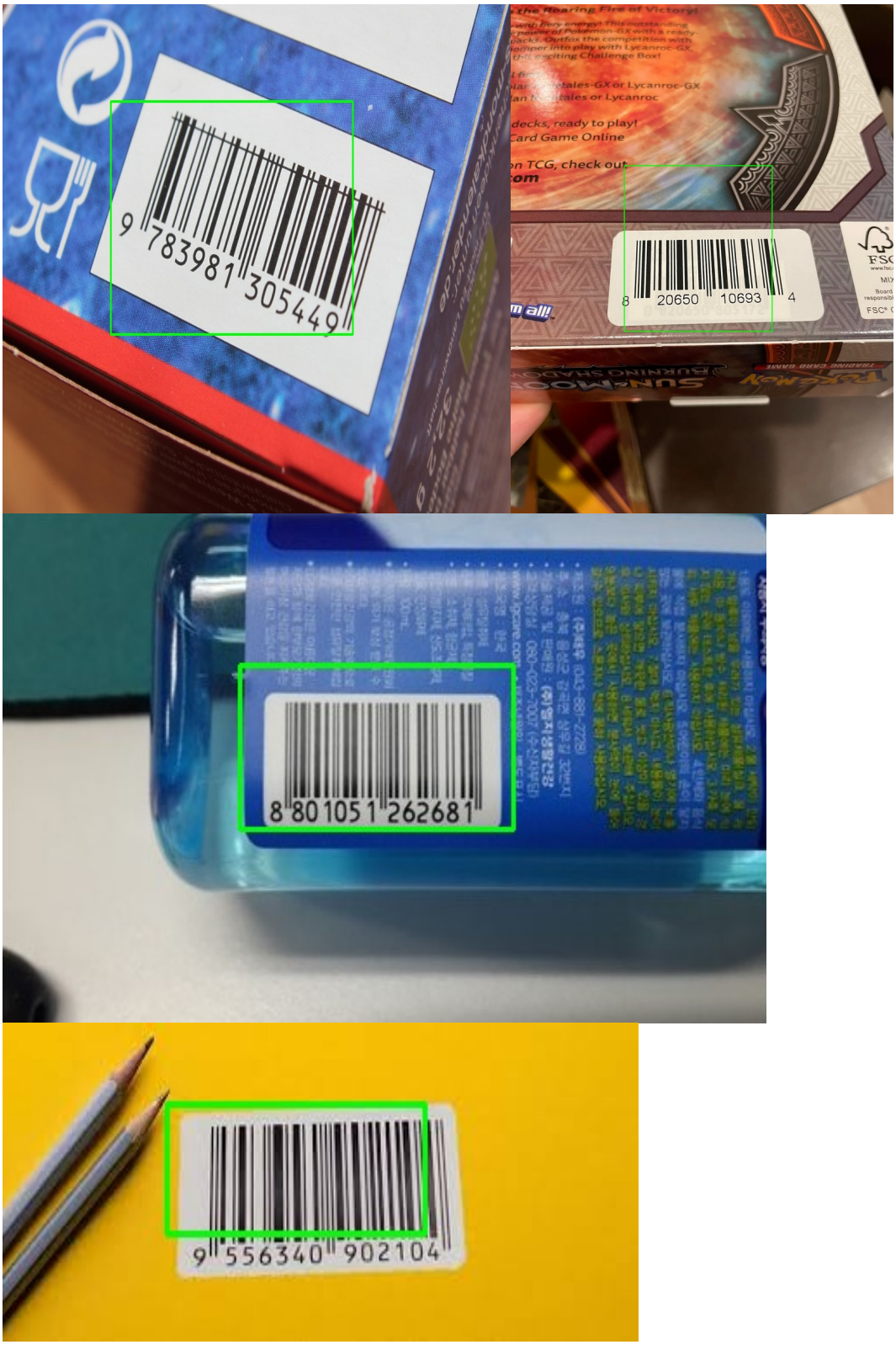
Од аспект на самото тренирање на моделот, се користи adam оптимизаторот којшто е прифатен како најдобар за општа примена во разни невронски мрежи. За loss функција се користи Huber loss со делта од 1.0. Оваа проценка е посебна бидејќи се однесува како MSE(Mean Squared Error) за мали недостатоци и преминува во MAE(Mean Absolute Error) за големи недостатоци. Ова овозможува да се справува со outliers многу полесно отколку константно MSE и истовремено е одржана прецизност на малите поправки како во MSE.

Моделот е трениран врз вкупно 632 слики (80% за тренирање, 10% за валидација и 10% за тестирање) извадени од база која може да се најде на овој линк

Следат неколку примери на излез добиени после 25 и 50 епохи на тренирање на моделот:



Сл. 10, 11, 12,13: Тестирање на метод 2 по 25 епохи

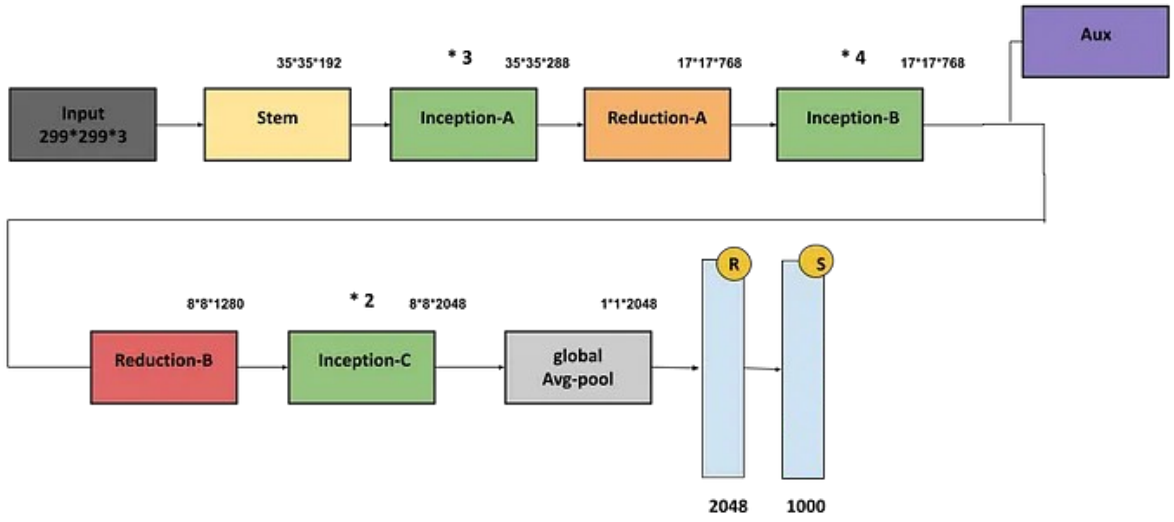


Сл. 14, 15, 16,17: Тестирање на метод 2 по 50 епохи

Метод 3 – InceptionV3

Следниот модел е истотака базирана на веќетренирана конволуциска невронска мрежа. Inception мрежата е сличен како ResNet со тоа што е исто трениран врз ImageNet базата и е оригинално наменет за категоризација, но има многу подлабока и покомплексна архитектура. Inception користи паралелни конволуции во еден слој, односно истата слика ја процесира со различни големини на кернел (1x1,3x3,5x5) паралелно и излезот го комбинира. Вака успева да добие подлабоко разбирање за бараните карактеристики. Во ефект, функционира како повеќе невронски мрежи да работат истовремено.

Дополнително во InceptionV3, која ќе ја искористиме ние, конволуциите со голем кернел се поделени во повеќе конволуции со помали кернели за да се намали процесирачката комплексност и истовремено задржи прецизност на проценките, на пример конволуција со кернел 5x5 ќе се претвори во две конволуции со кернел од 3x3. Дополнително се користат Факторизирани конволуции за да се поедностават проценките уште повеќе. Конволуција со кернел 3x3 би била претворена во две конволуции со големина 1x3 и 3x1.



Сл. 18: Структура на InceptionV3

Архитектурата на нашиот конкретен модел е:

-InceptionV3(weights='imagenet', include_top=False, input_shape=(299, 299, 3))

-GlobalAveragePooling2D
-Dense(2048,relu)

- BatchNormalization()

-Dropout(0.2)

-Dense(4,sigmoid)

Очигледно е слична како што направивме за ResNet50, но со неколку клучни разлики. Првично секако ја користиме основната InceptionV3 мрежа со истите влезни параметри како порано освен големината на сликите. InceptionV3 како влез очекува слики со големина 299x299 за најдобар резултат па затоа тоа го променуваме соодветно. Повторно користиме GlobalAveragePooling2D и Dense со relu активациска функција но го зголемуваме бројот на неврони од 1024 до 2048. Ова е поради различната структура на InceptionV3 мрежата која како излез дава вектор со 2048 димензии, ако директно го намаливме ова на 1024 би изгубиле пола од карактеристиките добиени од сликите па затоа и тоа го променуваме. Следи слојот BatchNormalization. Овој слој служи за да осигураме дека вредностите во невроните се измеѓу 0 и 1. Остатокот на архитектурата е идентнична со последниот модел со тоа што ја намалуваме стапката на Droupout од 0.3 до 0.2.Како излез добиваме повторно 4 вредности кои соодветствуваат на координатите на BoundingBox-от.

Иницијалниот модел повторно не го променуваме, само го надградуваме за наши цели.

При тренирање повторно се користат adam и Huber(delta=1.0) за оптимизација и губење, како и истите 632 слики од истата база спомната погоре со иста распределба за тренирање, валидација и тестирање.

Следат неколку примери за излез после 25 и 50 епохи на тренирање.

Сл. 19, 20, 21,22: Тестирање на метод 3 по 25 епохи

Сл. 23, 24, 25,26: Тестирање на метод 3 по 50 епохи

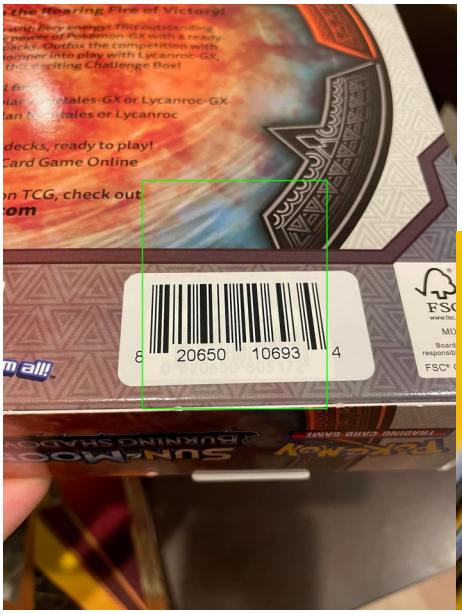
Читање пронајден баркод од слика

За самото читање односно декодирање на пронајдениот баркод повторно ќе ја искористиме pyzbar библиотеката која е wrapper на поопштата Zbar библиотека. Pyzbar е одличен алат за декодирање на баркод бидејќи подржвуа разни формати на баркод (Code 128, EAN, UPC и др.) и функционира без проблем со слики од ниска резолуција со услов дека е сепак видлив целиот баркод. Всушност, при читањето на баркод најважно е да се опфати целата ширина на баркодот бидејќи разликите измеѓу наизменичните црни и бели линии се тие што ја држат информацијата во баркодот. Висината не е до толку важна, само ни олеснува при наоѓање на ширината. Затоа алгоритам за пронаоѓање баркод којшто како резултат дава само една тенка хоризонтална линија која го изминува баркодот е подобар отколу некој којшто го опфаќа 90% од пикселите во баркодот но има изоставено дел од вертикалните линии.

Следат неколку примери за како Pyzbar се справува со резултатните слики на методите кои ги развивме за пронаоѓање на баркод.

Сл. 27,28: Примери од првиот метод

Излез 1:	Излез 2:
Barcode detected:	Barcode detected:
Type: EAN13	Type: EAN13
Data: 0820650106934	Data: 9556340902104




Сл 29,30: Примери од вториот метод по 25 епохи

Излез 1:

Barcode detected:

Type: EAN13

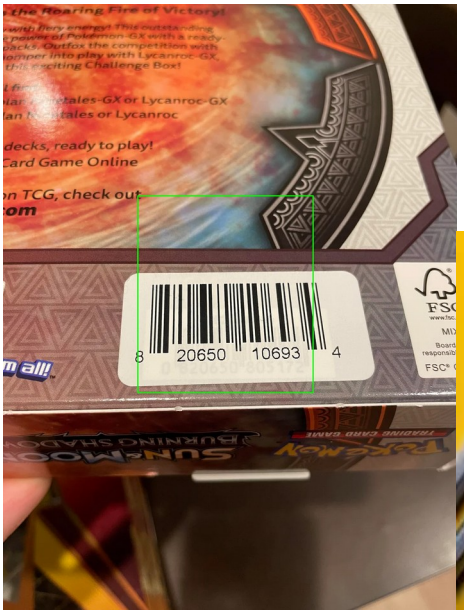
Data: 0820650106934

Излез 2:

Barcode detected:

Type: EAN13

Data: 9556340902104



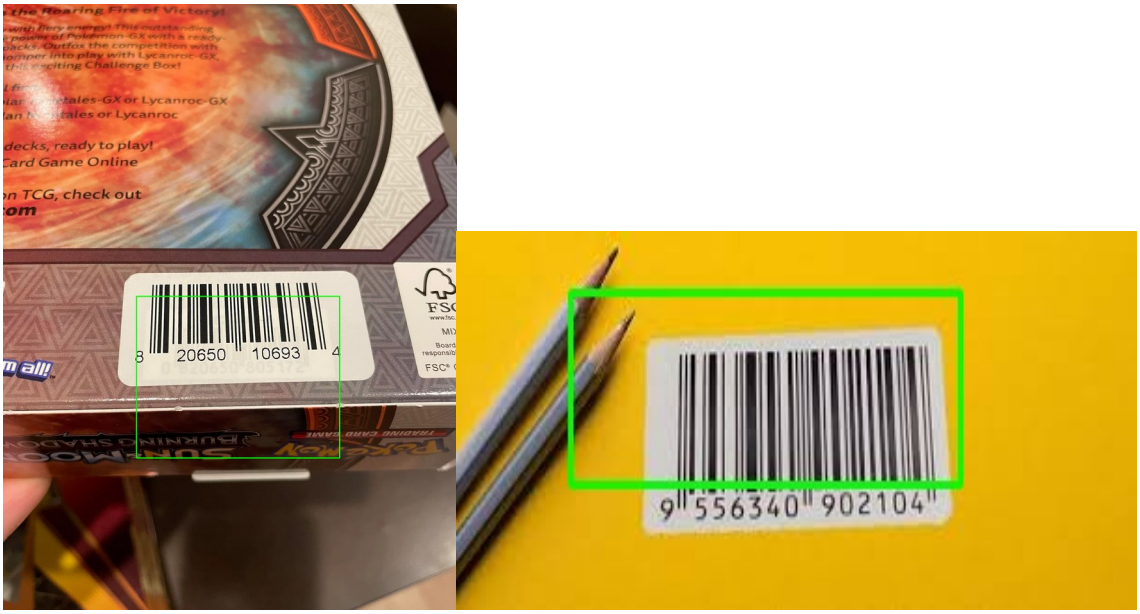

Сл 31,32: Примери од вториот метод по 50 епохи

Излез 1:

No barcode detected.

Излез 2:

No barcode detected.



Сл 33,34: Примери од третиот метод по 25 епохи

Излез 1:

Barcode detected:

Type: EAN13

Data: 0820650106934

Излез 2:

Barcode detected:

Type: EAN13

Data: 9556340902104

Сл 35,36: Примери од третиот метод по 50 епохи

Излез 1:

No barcode detected.

Излез 2:

Barcode detected:

Type: EAN13

Data: 9556340902104

Заклучок

Во овој проект видовме како алгоритми со различни нивоа на комплексност се справуваат со истиот проблем. Првиот метод, којшто е наједноставен, во просек даде најдобра апроксимација на локацијата на баркодот во сликата. Вториот метод, кој може да се каже дека има средна комплексност, прилично добри резултати дава. Особено кога баркодот е изолиран во сликата без многу шум. Додека третиот метод, кој е најкомплексен со најголема длабочина, иако ни дава функционални резултати кои може да се читаат од страна на pyzbar сепак не се целосно софпаѓаат со точниот регион на баркодот во сликата.

Овие резултати можеме да ги објасниме според стратегијата која се користи во секој метод. Првиот метод во реално време ја процесира конкретната слика и креира посебна проценка на каде би се наоѓал баркодот. Во споредба, вториот и третиот модел функционираат во принцип на “предвидување” каде се наоѓа баркодот според нивното тренирање, ова дава многу вариабилни резултати во зависност на сликите во тренирачкото множество, слоевите на моделот, бројот на епохи итн.

Извори

https://tritonstore.com.au/what-is-a-barcode/

https://www.pyimagesearch.com/wp-content/uploads/2015/01/the-ultimate-barcode-detection-guide.pdf

https://universe.roboflow.com/labeler-projects/barcodes-zmxjq/dataset/3

https://wandb.ai/mostafaibrahim17/ml-articles/reports/The-Basics-of-ResNet50---Vmlldzo2NDkwNDE2

https://gghantiwala.medium.com/understanding-the-architecture-of-the-inception-network-and-applying-it-to-a-real-world-dataset-169874795540

https://pyimagesearch.com/2018/05/21/an-opencv-barcode-and-qr-code-scanner-with-zbar/