Вештачка интелигенција 2024/2025

Класификација на рачно нацртани цртежи, со користење на конволуциски невронски мрежи

Изработила: Симона Ристовска 221003

Ментор: Проф. Андреа Кулаков

Содржина

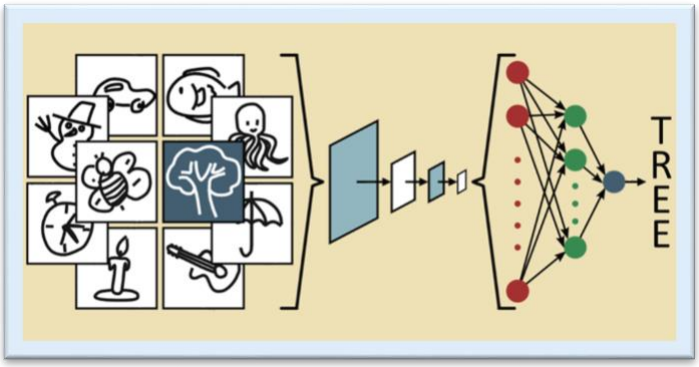
Вовед

Во оваа семинарска работа разработена е имплементацијата и функционалноста за препознавање на рачно нацртани слики ("doodles") со примена на вештачка интелигенција и машинско учење. Користен е модел базиран на длабока конволуциска невронска мрежа (CNN), кој е тренирана врз податоци од Quick, Draw! податочното множество, кое содржи стотици илјади цртежи распределени во повеќе категории. Оваа мрежа е способна ефикасно да учи и препознава карактеристики на сликите, што овозможува висок степен на прецизност при класификацијата.

Главната цел на проектот е да овозможи корисниците да цртаат во реално време на веб-интерфејс, по што моделот автоматски врши препознавање на цртежот и дава повратна информација за категоријата на нацртаниот објект, придружена со степенот(процент) на сигурност на предвидувањето. Дополнително, во рамките на проектот се анализираат перформансите на моделот, со што се идентификуваат неговите предности и евентуалните ограничувања во практична примена.

Почетна идеја

Зошто токму оваа тема?



Слика 1 – Интуитивен приказ на конволуциска мрежа која класифицира рачно нацртани објекти

По завршување на целосниот материјал по предметот “Вештачка интелигенција”, и покрај многубројните интересни теми кои се изучуваа, несомнено најголем впечаток ми оставија невронските мрежи. Бидејќи, предметот ги покрива воведните концепти во вештачката интелигенција и машинското учење, интересот за невронските мрежи особено го изградив на предметот “Вовед во науката за податоци”. Конкретно, идејата за изборот на темата за проектот произлезе од мојата љубопитност за компјутерската визија и машинското учење. Со знаењето од двата предмети, бев фасцинирана од тоа како машините можат да препознаваат слики, препознаваат пишан текст или дури и уметност, па затоа решив да истражам подлабоко во оваа област. Со истражување во оваа област наидов на проектот **"Quick, Draw!"** од Google – забавен, но моќен систем кој користи невронски мрежи за да препознае рачно нацртани објекти. Сакав да направам нешто слично, но поедноставно и во моја верзија, при што ќе можам да научам повеќе за CNN (Convolutional Neural Networks), а подоцна и самостојно да имплементирам модел кој ќе може да препознае рачно цртани слики.

Моделот кој го направив сакав да виде интуитивен и за корисниците кои преку интерактивен веб-интерфејс каде ќе можат да цртаат и веднаш да добијат предвидување за нивниот цртеж.

Први впечатоци

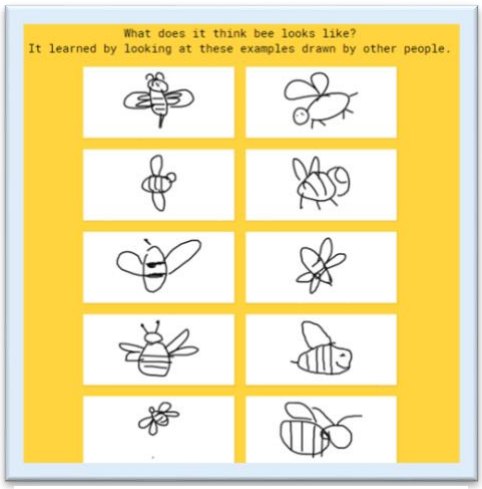
Кои беа првите впечатоци кога ја избрав темата.

Една од главните цели на овој проект беше да се истражи како вештачката интелигенција може да интерпретира симплифицирани слики и да препознава цртежи кои се направени на различни начини од различни корисници. За разлика од традиционалната класификација на фотографии, каде што моделите имаат доволно визуелни информации како бои, сенки и детални форми, препознавањето на рачно нацртани фигури претставува значително поголем предизвик.

Главниот проблем лежи во фактот што луѓето имаат уникатни стилови на цртање, различни нивоа на умешност, и често оставаат недовршени или деформирани линии при брзо скицирање. Ова значи дека истиот објект може да изгледа сосема различно во зависност од тоа кој го цртал, што го отежнува процесот на автоматско препознавање од страна на моделот. За да се справам со овој предизвик, одлучив да тренирам модел кој ќе биде изложен на голем број различни стилови на цртање, што ќе му помогне да ги разбере разликите меѓу начинот на кој луѓето ги визуелизираат ваквите концепти. Исто така, моделот треба да биде доволно робустен за да направи точни предвидувања дури и кога цртежите се несоодветни и неконзистентни.

Податочно множество

За тренинг на моделот, го користам податочното множество од **Quick, Draw!** , кој е еден од најголемите јавно достапни збирки на рачно нацртани слики. Овој датасет содржи милиони цртежи собрани од луѓе низ целиот свет, кои нацртале различни објекти во краток временски период како дел од играта "Quick, Draw!" развиена од Google. Датасетот е составен од **нумерички битмапи со големина 28×28 пиксели во grayscale формат**, што го прави погоден за обработка со Convolutional Neural Networks (CNN), бидејќи ги задржува просторните информации на цртежите. Предизвикот со овој датасет е што цртежите **не се стандардизирани**, што значи дека може да има значителни разлики во стилот на цртање, деталите и квалитетот

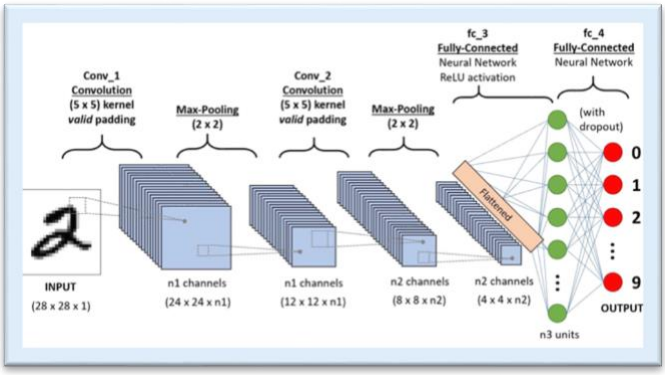


Слика 2 -Примерок од податочното множество користено во проектот

на цртежите – некои можат да бидат јасни, додека други се едвај препознатливи. Токму оваа варијација го прави датасетот идеален за тренирање на модел кој треба да биде робустен на различни начини на изразување**.** За да ја подобрам точноста на моделот, правам **предобработка на податоците(data augmentation)**, вклучувајќи нормализација на пикселите и аугментација (ротирање, превртување и додавање шум), за да го направам моделот пофлексибилен и поотпорен на различни стилови на цртање. Ова гарантира дека мрежата не учи само фиксни шаблони, туку развива разбирање за основните карактеристики на објектите, без разлика како се нацртани**.**

Избор на технологиите

Зошто Конволуциски невронски мрежи (CNN)?



Слика 3 - Архитектура на конволуцските мрежи

При развивањето на овој проект, беше клучно да се избере архитектура која ќе може ефикасно да препознава и класифицира рачно нацртани слики. Традиционалните машински алгоритми како SVM (Support Vector Machines) и KNN (K-Nearest Neighbors), иако брзи и ефикасни за едноставни задачи со структурирани податоци, ваквите модели не се погодни за обработка на слики, бидејќи не можат да ги земат во предвид просторните информации и зависностите помеѓу пикселите. Од друга страна, основните невронски мрежи (Multilayer Perceptron - MLP) ги третираат сите пиксели како независни влезови, што значи дека не можат да препознаваат карактеристики како што се рабови, линии и форми кои се клучни за препознавање на цртежи.

Токму затоа, најдобриот избор беа **Конволуциските невронски мрежи** (CNN) кои се специјално дизајнирани за обработка на визуелни податоци**.** CNN мрежите користат конволуциски слоеви за да идентификуваат локални шаблони во сликите, како рабови и криви, а потоа комбинираат различни нивоа на карактеристики за да формираат повисоко ниво на претставување на објектите. Оваа способност ги прави CNN исклучително моќни за задачи како што е класификација на рачно цртани фигури**,** каде што истите објекти можат да се појават нацртани на различни начини. Дополнително, CNN ефикасно ги намалува вишокот на информации преку **MaxPooling**, што помага да се задржат најзначајните карактеристики на сликите и да се подобри перформансот на моделот без преголем број на параметри. Дополнително конволуциските мрежи работат добро со мали слики како **28x28 px**, што е форматот на Quick, Draw! dataset-от што го

користам**.** За имплементација, избрав **TensorFlow + Keras**, бидејќи е најпопуларната алатка за длабоко учење во моментот.

Архитектурата на моделот

Процес на дизајнирање и оптимизација

При дизајнирањето на моделот, главната цел беше да се создаде **балансиран CNN модел** кој ќе биде доволно длабок за да извлече релевантни карактеристики од рачно нацртаните слики, но истовремено доволно ефикасен за да се тренира брзо и да не троши премногу ресурси при инференција. Почнав со идејата за **едноставен CNN модел со 3-4 Convolutional слоеви**, каде што секој конволутивен слој ќе биде проследен со **MaxPooling**, со цел да се намалат димензиите и да се задржат најважните карактеристики. Првичните тестови покажаа дека **еден или два конволутивни слоеви не се доволни** за сложеноста на податоците во Quick, Draw! датасетот, бидејќи моделот не можеше да разликува слични категории.

```
def build_model(input_shape, num_classes):
    inputs = tf.keras.Input(shape=(28, 28, 1))

    x = layers.Conv2D(64, (3, 3), activation='relu', padding="same")(inputs)
    x = layers.MaxPooling2D((2, 2))(x)

    x = layers.Conv2D(128, (3, 3), activation='relu', padding="same")(x)
    x = layers.BatchNormalization()(x)
    x = layers.MaxPooling2D((2, 2))(x)

    x = layers.Conv2D(256, (3, 3), activation='relu', padding="same")(x)
    x = layers.BatchNormalization()(x)
    x = layers.MaxPooling2D((2, 2))(x)

    x = layers.Flatten()(x)
    x = layers.Dense( units: 512, activation='relu')(x)
    x = layers.Dropout(0.5)(x)

    outputs = layers.Dense(num_classes, activation='softmax')(x)
    model = tf.keras.Model(inputs=inputs, outputs=outputs)
    return model
```

Слика 4 - Функцијата build_model каде се гради конволуциската мрежа

На почеток, започнав со **150 категории и 20 епохи**, при што за секоја категорија користев **1000 примероци**. Овој пристап теоретски требаше да обезбеди **висока точност**, но набрзо се соочив со **ограничувања во хардверските ресурси**. Потоа, се обидов со 50 категории, 1000 примероци по категорија и 12 епохи. Сепак, истово не беше можно да се изврши целосно поради ограничувањата од хардверот. Бидејќи тренирањето на овој обемен модел бараше значителна меморија и пресметковна моќ, се забележаа проблеми со времетраењето на тренингот, како и со достапноста на GPU ресурсите. Како резултат, моделот беше преспор за реална примена**,** што ме принуди да го **редуцирам број на категории** (моментално 20) и да го прилагодам **бројот на примероци по класа (**5000 примероци**)**, за да најдам баланс помеѓу точноста и ефикасноста во согласност со ресурсите кои ги поседувам.

За да го подобрам моделот, го зголемив бројот на Convolutional слоеви на **три**, со филтри кои прогресивно растат (**64 → 128 → 256**), со што мрежата постепено учи од основни карактеристики (линии, рабови) во првите слоеви, до покомплексни форми во подлабоките слоеви. Потоа додадов **Batch Normalization по вториот и третиот слој**, бидејќи при првите тестирања забележав дека активациите во мрежата имаат големи варијации, што го правеше учењето нестабилно. Со ова, моделот почна да се тренира побрзо и со поголема стабилност**,** постигнувајќи повисока точност во помалку епохи (моментално 8).

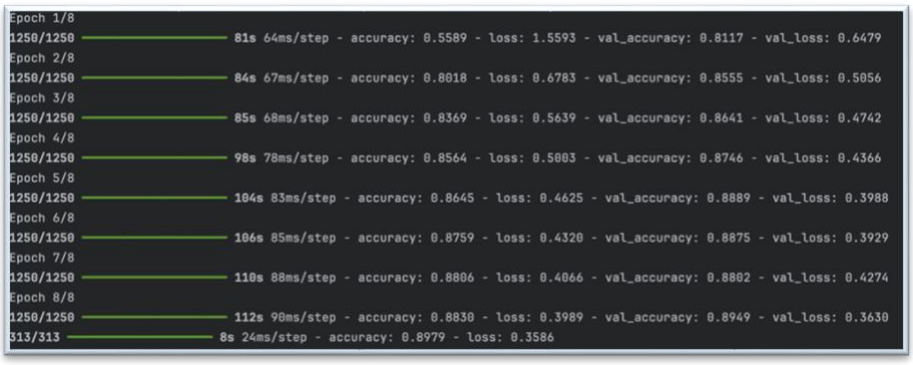
Дополнително, бидејќи податоците беа релативно ограничени и содржеа голема варијабилност во стилот на цртање, се појави ризик од overfitting, особено во Dense слоевите каде што бројот на параметри драстично расте. За да го спречам ова, додадов **Dropout слој** со веројатност 0.5, кој помогна да се редуцира зависноста од специфични неврони, што резултираше со подобра генерализација при предвидувањата на нови, невидени цртежи.

Првично, користев **learning rate од 0.001**, но забележав дека моделот многу бавно конвергира, па направив експерименти со адаптивни оптимизатори како Adam, што значително ја подобри стабилноста. Исто така, се одлучив за **batch size од 64**, што обезбеди добра рамнотежа помеѓу брзината на тренинг и стабилноста на градиентите.

На крајот, последниот слој во моделот е **Dense слојот со 512 неврони** и активација ReLU, проследен со финален **излезен слој со softmax активација**, кој ги предвидува веројатностите за секоја класа. Овој дизајн постигна **најдобар баланс помеѓу сложеноста и брзината**, при што моделот остана доволно лесен за да може брзо да се користи во реално време, а сепак доволно моќен за да препознава различни стилови на цртање со висока точност.

Перформанси и метрики (benchmarks)

Напредок на тренирањето и перформанси на моделот



*Слика 5 – Следење на метрики за секоја епоха поединечно за време на **тренингот***

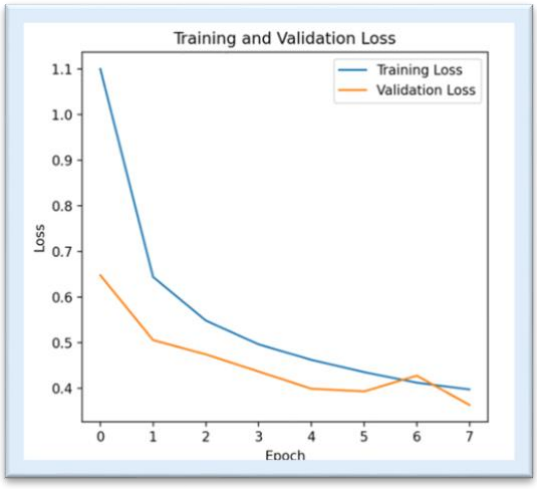
Слика бр.5 го прикажува напредокот на тренирањето низ **8 епохи**, вклучувајќи ја точноста (**accuracy**), загубата (**loss**) и перформансите на моделот при валидација (**validation accuracy/loss**). Почетната точност на моделот е **55.89%**, со загуба од **1.5593**, додека валидациската точност е значително повисока на **81.17%**, што укажува дека моделот од самиот почеток добро се приспособува кон податоците за тестирање. Како што напредуваат епохите, точноста постепено се подобрува, достигнувајќи **88.83%** на последната епоха, додека конечната валидациска точност е **89.49%**. Истовремено, загубата значително се намалува, што покажува дека моделот станува сè попрецизен во класификацијата.

Графици за точност и загуба при тренирањето и валидацијата

*Слика 6 – График за точност на **тренирачкото** и **валидациското** множество*

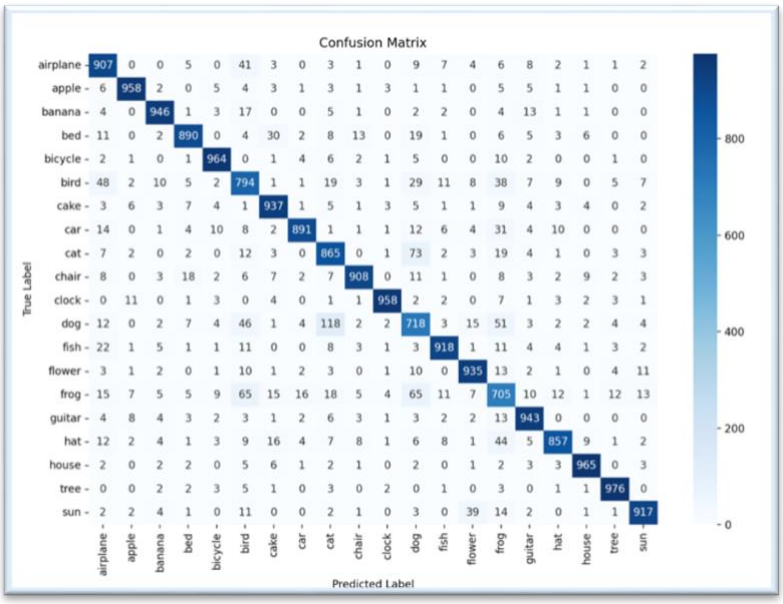
Овој график го прикажува растот на **точноста** при тренирањето и валидацијата, додека десниот го покажува намалувањето на **загубата**. Може да се забележи дека и **тренинг** и **валидацискатa** точност брзо се зголемуваат во првите неколку епохи, а потоа се стабилизираат околу **90%**, што укажува на добра генерализација на моделот.

Од друга страна, загубата на овој график постојано опаѓа, што значи дека моделот станува сè попрецизен во своите предвидувања. Важно е да се напомене дека линиите за тренинг и валидација се **приближни**, што значи дека нема значително пренавикнување (**overfitting**), што е одличен знак за стабилноста на моделот.



*Слика 7 – График за загубата на **тренирачкото** и **валидациското** множество*

Матрица на збунетост и точноста на моделот по класи



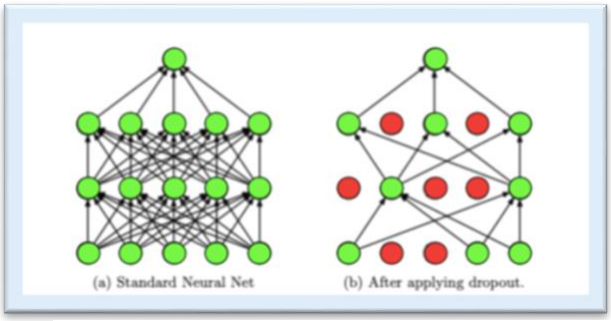
Слика 8 – Матрица на збунетост

Слика бр.7 ја прикажува **матрицата на збунетост**, која визуелно ги анализира точните и погрешните предвидувања на моделот за секоја од 20-те класи (категории) на цртежи. Дијагоналните полиња претставуваат точни предвидувања, каде што моделот правилно ја препознал категоријата, додека останатите полиња покажуваат каде моделот ги згрешил класите. На пример, цртежите на "авион" (airplane) биле точно предвидени 907 пати, но во неколку случаи биле класифицирани како „птица“ или „сонце“. Ова укажува дека некои категории имаат слични карактеристики и моделот може да ги меша. Генерално, високите бројки на предвидувања по дијагонала покажува дека моделот добро ги разликува категориите, но сепак има мали грешки кај категории кај кои потешко се воочува разликата.

Техники за спречување на overfitting и забрзување на конвергенција

Dropout

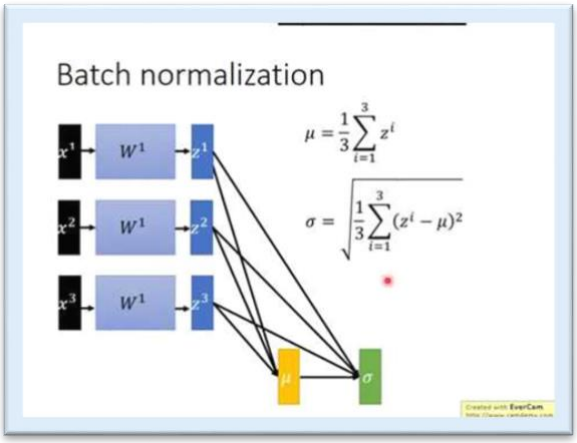
Една од клучните техники за спречување на overfitting искористени во овој проект е **Dropout** слојот. Dropout претставува моќна регуларизациска техника со која за време на процесот на тренирање, случајно се деактивираат дел од невроните во мрежата со одредена веројатност. Ваквото случајно



Слика 9 -Графички приказ за Dropout

исклучување спречува одредени неврони да доминираат и да се потпираат само на одредено множество на карактеристики, што ја намалува преголемата специјализација на моделот и му овозможува да развива поголема генерализација. Во конкретната архитектура на моделот, Dropout слојот е поставен по целосно поврзаниот (Dense) слој и има зададена веројатност на исклучување од 50%, што значи дека половина од невроните во тој слој се привремено отстранети за време на секој чекор на учење. Ова овозможува на моделот да ги научи робустните и општи својства на влезните податоци наместо само тесни специфичности, со што значително се зголемува неговата ефикасност на нови, невидени податоци.

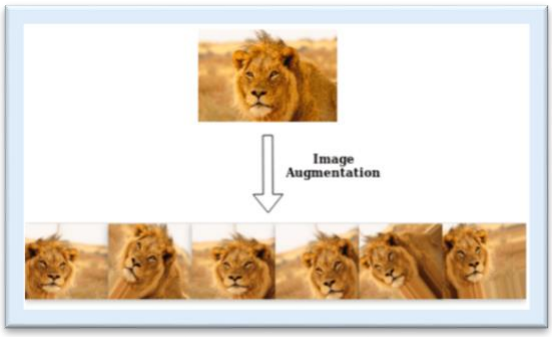
Batch Normalization



Слика 10 -Графички приказ за Batch

Втора значајна техника која е употребена за да се намали ризикот од overfitting е **Batch Normalization**. Овој метод има клучна улога во подобрување на стабилноста и ефикасноста на процесот на учење на длабоките невронски мрежи. Batch Normalization функционира преку нормализирање на излезите на претходниот слој во секој мини-бач (mini-batch), така што средната вредност се поставува на нула, а стандардната девијација на еден. Со ова се елиминира проблемот на внатрешни коваријантни поместувања (internal covariate shift), кои можат да ја забават конвергенцијата и да предизвикаат нестабилност во процесот на учење. Во имплементираниот модел, Batch Normalization е вклучена веднаш после конволутивните слоеви, со што се подобрува не само брзината на учењето, туку и способноста за генерализација на мрежата.

Data Augmentation



Слика 11 -Графички приказ за Data Augmentation

Третата, многу значајна техника применета во овој проект, е **Data Augmentation**. Оваа техника подразбира систематско проширување на тренинг сетот преку создавање на дополнителни, вештачки модифицирани примероци од постоечките податоци. Data Augmentation вклучува примена на случајни трансформации како што се ротација, зумирање, транслација и рефлектирање на сликите. Со примена на вакви трансформации, се генерираат

нови примероци кои ги задржуваат суштинските карактеристики на оригиналните податоци, но ја зголемуваат нивната различност. Оваа зголемена разноликост помага на моделот да учи поопшти и поцврсти карактеристики, наместо специфични детали од оригиналниот датасет, што значително ја подобрува неговата способност за генерализација на нови, непознати податоци.

Корисничкиот интерфејс

Како се поврзува моделот?

За да ја направам апликацијата интерактивна, имплементирав веб-апликација каде што корисникот може да црта и веднаш да добие предвидување од моделот. Фронт-ендот е изграден со HTML, CSS и JavaScript, при што Canvas API се користи за цртање. Бек-ендот е базиран на Flask, кој овозможува комуникација со CNN моделот изграден со TensorFlow/Keras**.**

Функционирањето е едноставно, корисникот црта во Canvas, JavaScript ја зема сликата и ја конвертира во **Base64 формат**, по што се испраќа до Flask серверот преку POST барање. Серверот ја обработува сликата, ја конвертира во **28x28 grayscale** и ја праќа кон CNN моделот. Моделот враќа предвидување, кое потоа се прикажува на веб. Оваа архитектура овозможува брза обработка и real-time препознавање на цртежите.

Референци

- Видео содржина - https://youtu.be/hfMk-kjRv4c?si=223x4AcWoTW1Sb1l
- Видео содржина - https://youtu.be/AsYczuDBGp0?si=ms1k519aODLQdufJ
- Статија - https://medium.com/towards-data-science/doodling-with-deep-learning-1b0e11b858aa
- Линк до податочното множество - https://github.com/googlecreativelab/quickdraw-dataset