Универзитет Св. Кирил и Методиј – Скопје

Факултет за информатички науки и компјутерско инженерство

Дигитално процесирање на слика

Летен Семестар 2023/24

Тема: Background Subtraction

Ментор:

проф. д-р Ивица Димитровски

Студент:

Љубица Дамјановиќ 221173

Јуни, 2024

Содржина

Вовед

Во денешниот дигитален свет, обработката на слики игра клучна улога во многу апликации, вклучувајќи безбедност, надзор, виртуелна реалност и создавање на содржини. Еден од најзначајните предизвици во оваа област е изолацијата на објекти од нивната позадина, процес познат како background subtraction (одземање на позадина).

Проектот "Background Subtraction" има за цел да развие и имплементира модели за автоматско предвидување на маската на дадена слика, овозможувајќи изолирање на објектот од неговата позадина. Со користење на алгоритми за машинско учење и компјутерска визија, овој проект ќе овозможи отстранување на позадината, оставајќи го само релевантниот објект.

Концепт

Background subtraction (одземање на позадина) е техника што се користи во обработката на слики и компјутерската визија за да се одделат елементите на преден план (објектите од интерес) од позадината во слики. Оваа техника игра клучна улога во различни области како што се графички дизајн, фотографија, е-трговија и дигитален маркетинг. Кога ќе се отстрани позадината, предметот зазема централно место, што резултира со поостра, попрофесионална слика.

Постојат неколку методи за да се постигне ова. Најчесто користена техника е рачно избирање, кое вклучува користење на софтвер за уредување слики како што е Adobe Photoshop за следење на прегледот на објектот. Овој процес бара прецизност и внимание на деталите за да се обезбеди прецизно сечење. Дизајнерите користат алатки како алатката Пенкало или алатката Magnetic Lasso за прецизно следење на објектот и создавање патека што го одвојува од позадината. Откако ќе се избере, позадината може лесно да се отстрани или замени.

Друг метод за отстранување на заднината е да се користат автоматизирани алгоритми. Овие алгоритми користат напредни техники за компјутерска визија за откривање и разликување помеѓу пикселите на преден план и заднина. Тие ги анализираат факторите како што се бојата, текстурата и откривањето на рабовите за да ги одредат границите на објектот. Алгоритмите за машинско учење, како што се невронските мрежи, може да се обучат на големи збирки на податоци за да се подобри точноста на отстранувањето на позадината. Овие автоматизирани техники се особено корисни кога се работи со слики кои имаат јасен контраст помеѓу предметот и позадината.

Примени

Background subtraction се користи широко во различни полиња и апликации, вклучувајќи:

- Видео надзор: Откривање и следење на подвижни објекти, како што се луѓе или возила, во безбедносните снимки.
- Интеракција човек-компјутер: Овозможување препознавање на гестови и интеракција со изолирање на корисниците од позадина.
- Следење на сообраќајот: Идентификување и следење на возила за анализа и управување со сообраќајот.
- Следење на објекти: Следење на движењето на предметите во спортот, следењето на дивиот свет и роботиката.
- Уредување слики и видео: Отстранување на позадини за композитирање или специјални ефекти во медиумско производство.

Модели за предвидување на маска

За целите на овој проект, креирани се 3 модели кои ќе предвидуваат маски на објектите во сликите со помош на библиотеката PyTorch и програмскиот јазик Python.

PyTorch е целосно опремен framework за градење модели за длабоко учење, што е тип на машинско учење што вообичаено се користи во апликации како што се препознавање слики и обработка на јазици (NLP).

Тензорите (tensors) се основен тип на податоци во PyTorch, сличен на повеќедимензионална низа, што се користи за складирање и манипулирање со влезовите и излезите на моделот, како и со параметрите на моделот. Тензорите се слични на низите на NumPy, освен што тензорите можат да работат на графичкиот процесор (GPU) за да го забрзаат пресметувањето. Тензорите ги користиме токму во овој проект за тренирање на моделите.

Податочното множество кое е искористено за тренирање на овие модели е достапно тука, но во овој случај се искористени само 480p фолдерите. Ова податочно пножество се состои од 3455 парови: слика со маска. Резултатите од сите модели се различни, но тој модел што е избран за да служи на демото не е најбеспрекорен од причина што тренирачкото множествно само по себе не е доволно големо за да се добие одличен модел. Секако за да се добијат подобри резултати потребно е да се искористи многу поголемо податочно множество. Ова податочно множество е поделено на следниот начин: 80% од податоците се ставени во тренирачкото множество, додека пак останатите 20% се ставени во тестирачкото множество.

Моделите се тренирани на графичката карта NVIDIA RTX 4060 Ti со 16GB VRAM на локален компјутер.

Модел 1 – U-Net

unet_model се заснова на U-Net архитектурата, која е широко користена за задачи за семантичка сегментација, вклучително и одземање на позадината. Архитектурата U-Net се состои од мрежа на енкодер-декодер со прескокнувачки врски, што овозможува прецизна локализација и зачувување на контекстот за време на сегментацијата.

Објаснување на архитектурата на моделот:

Патека за склучување договор (енкодер)

Енкодерот го доловува контекстот и ги извлекува карактеристиките на високо ниво од влезната слика преку серија конволуциони операции и слоеви за намалување на примероците.

1. Почетен конволуционерен слој (self.inc):
   - Применува операција за двојно свртување на влезната слика, зголемувајќи ја длабочината на карактеристиките.
   - Се состои од два конволутивни слоеви 3x3 проследени со нормализација на серија и активирање ReLU.
2. Намалување на примероци на слоеви (self.down1 до self.down4):
   - Секој слој за намалување на примероците содржи операција на максимално здружување проследено со операција за двојна конволуција.
   - Ги намалува просторните димензии додека го зголемува бројот на канали за функции за да се доловат хиерархиските карактеристики.

Експанзивна патека (декодер)

Декодерот ги презема примероците од мапите на карактеристики за да ја генерира конечната маска за сегментација, инкорпорирајќи ситно-гранулирани детали и искористувајќи ги прескокнувачките врски од екодерот.

1. Преземање примероци на слоеви (self.up1 до self.up4):
   - Користи или билинеарно земање примероци или транспонирана конволуција за зголемување на просторните димензии.
   - Конкатанира мапи на карактеристики од соодветниот енкодер за да се зачуваат просторните информации и да се обезбеди прецизна локализација.

Излезен слој

Излезен конволутивен слој (self.outc):

- Ја произведува последната маска за сегментација со еден канал што ја претставува веројатноста секој пиксел да припаѓа на класата на преден план.
- Се состои од 1x1 конволутивен слој.

Пропусница нанапред (Forward Pass)

За време на додавањето нанапред:

1. Влезната слика се обработува преку енкодерот, извлекувајќи хиерархиски карактеристики.
2. Мапите на карактеристики од енкодерот се поврзуваат со оние од декодерот за да се зачуваат просторните информации.
3. Конечната маска за сегментација се генерира од излезниот слој, што ја претставува веројатноста секој пиксел да припаѓа на класата на преден план.

Контролен пункт (Checkpointing)

Методот use_checkpointing овозможува контролна точка, техника за заштеда на меморија која ги менува пресметките за меморија за време на обуката. Применува контролни точки на секој модул во мрежата.

Во овој случај кога имаме помало податочно множество, овој модел даде најдобри резултати. Овој модел е трениран во 50 и во 150 епохи и добиени се следните прецизности:

```
Testing model with 50 epochs:
Accuracy: 0.8807700569257969
Testing model with 150 epochs:
Accuracy: 0.31210544180763006
```

Моделот кој е трениран во 50 епохи има многу поголема прецизност од моделот кој е трениран во 150 епохи поради overfitting. Ова се има случено поради тоа што податочното множество е мало и со зголемување на епохите всушност се губи конзистентноста – моделот веќе конвергирал во првите 50 епохи и со понатамошното тренирање тој почнал да дивергира.

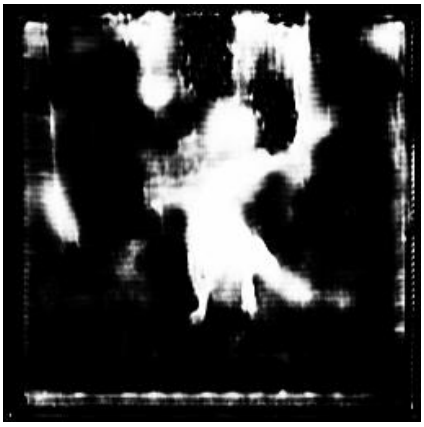
Примери од предвидени маски од моделот со 50 епохи:

result_mask_0

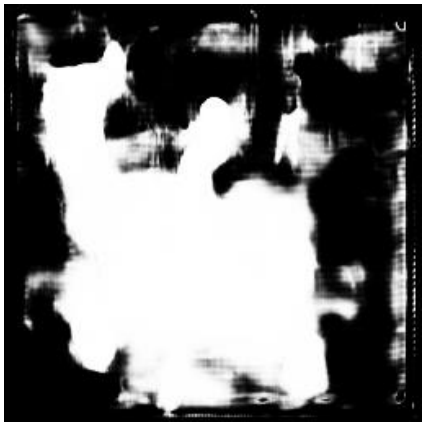
result_mask_30

result_mask_100

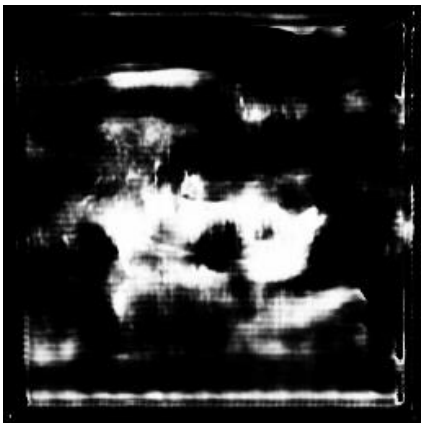
Примери од предвидени маски од моделот со 150 епохи:



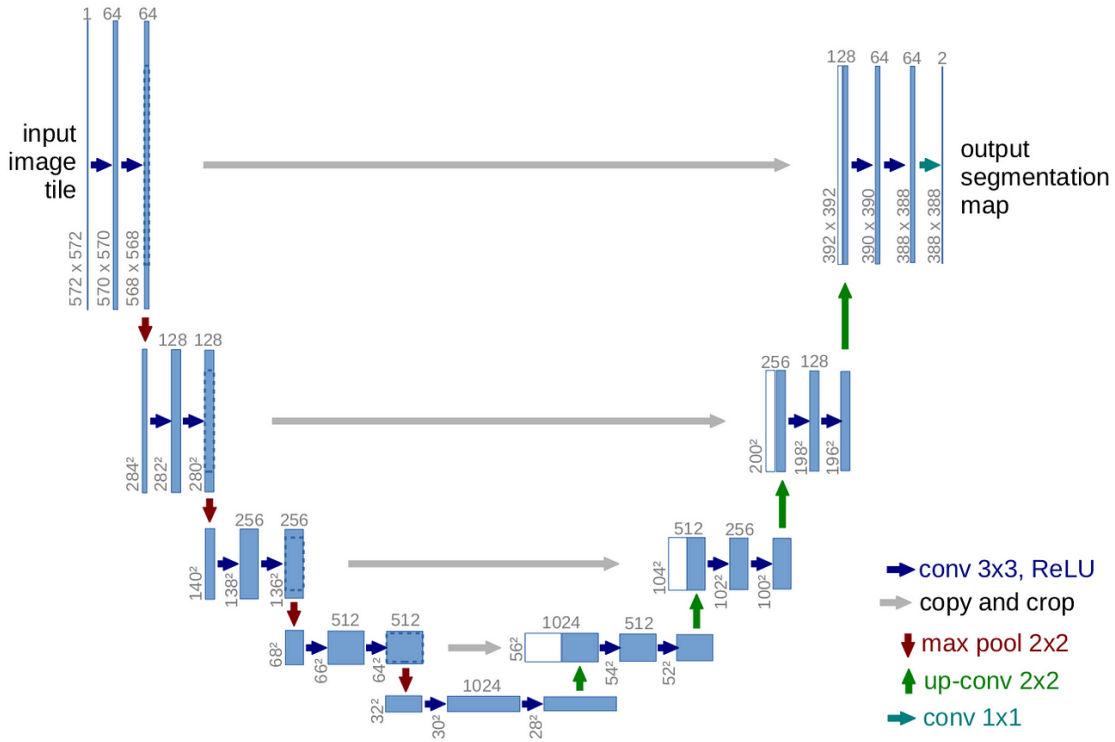
result_mask_0



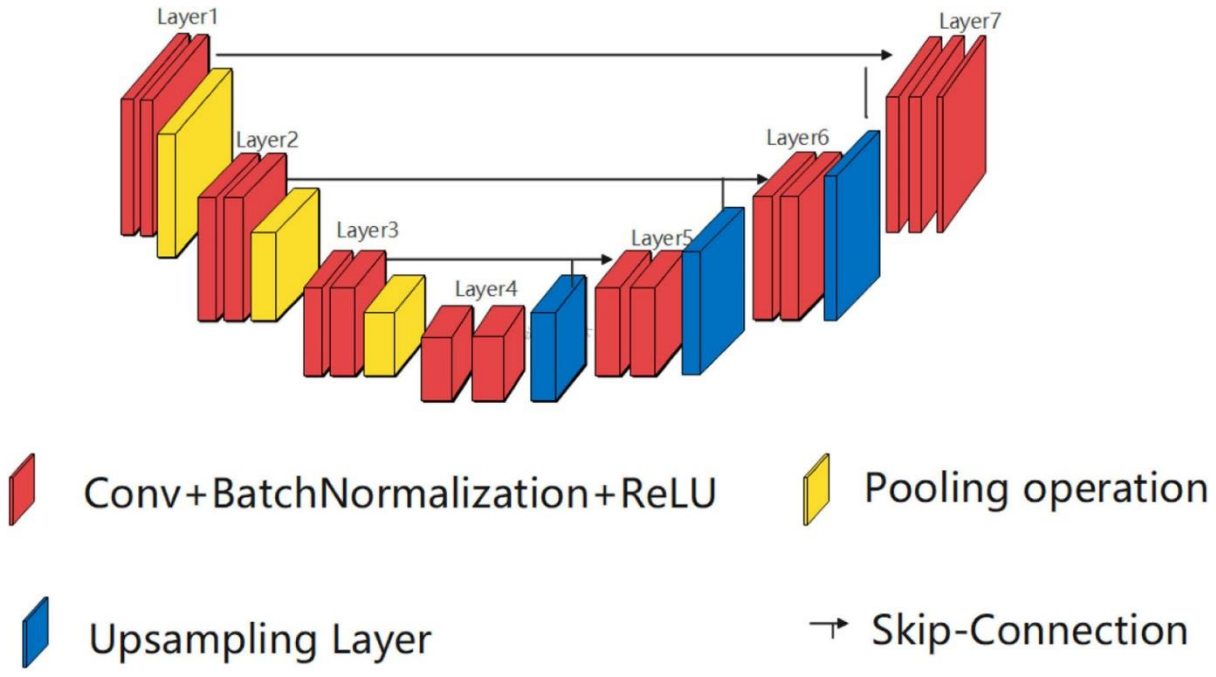
result_mask_30



result_mask_100



Приказ на слоевите на U-Net архитектурата



Приказ на слоевите на U-Net архитектурата

Модел 2 – U^2-Net

u2net_model се базира на архитектурата U^2-Net, која е напредна верзија на U-Net дизајнирана за задачи за сегментација на слики, како што е одземање на позадина. Моделот U^2-Net ја подобрува U-Net со инкорпорирање на вгнездена U-Net структура, овозможувајќи подобро извлекување на карактеристиките и пофини детали во сегментацијата.

Објаснување на архитектурата на моделот:

Патека за склучување договор (енкодер)

Енкодерот доловува контекстуални информации преку серија на конволутивни и здружени слоеви, прогресивно намалувајќи ги просторните димензии додека ја зголемува длабочината на карактеристиките.

1. Почетни конволуциони слоеви:
   - self.c1 и self.c2: Конволуциони слоеви кои ја обработуваат влезната слика (3 канали за RGB) и произведуваат 64 мапи со карактеристики.
   - self.p1: Максимален здружен слој кој ги преполовува просторните димензии.
2. Следни конволуциони слоеви:
   - self.c3 до self.c8: Овие слоеви го продолжуваат процесот на конволуција и здружување, со длабочини на мапата на карактеристики од 128, 256 и 512, додека просторните димензии се преполовуваат на секој чекор на здружување.
3. Слоеви на тесно грло:
   - self.c9 и self.c10: Формирање на тесното грло со 1024 мапи на карактеристики, доловувајќи апстрактни карактеристики на високо ниво.

Експанзивна патека (декодер)

Декодерот ги обновува примероците од мапите на карактеристиките назад до оригиналните димензии на сликата, овозможувајќи прецизна локализација на сегментираните објекти.

1. Upsampling и конволуциони слоеви:
   - self.u6 до self.u9: Транспонирани конволуциони слоеви (деконволуција) кои ги удвојуваат просторните димензии.
   - self.c11 до self.c18: Конволуциони слоеви кои ги усовршуваат карактеристиките со подобрени примероци, постепено намалувајќи ја длабочината на карактеристиките (1024 до 512, 512 до 256, 256 до 128, 128 до 64).
2. Прескокнувачки врски:
   - Прескокнувачките врски ги поврзуваат мапите на карактеристики од енкодерот до соодветните слоеви во декодерот, зачувувајќи фини детали.

Излезен слој

Завршна конволуција:

- self.output: Слој на конволуција 1x1 што ги намалува мапите на карактеристиките на 1, создавајќи излез од еден канал.
- Се применува функцијата за активирање torch.sigmoid, која дава вредности помеѓу 0 и 1 за бинарна сегментација.

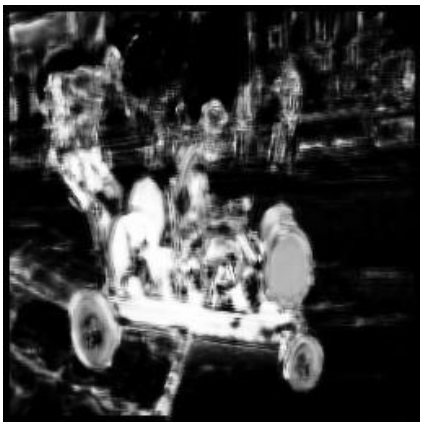
Пропусница нанапред (Forward Pass)

Во додавањето напред:

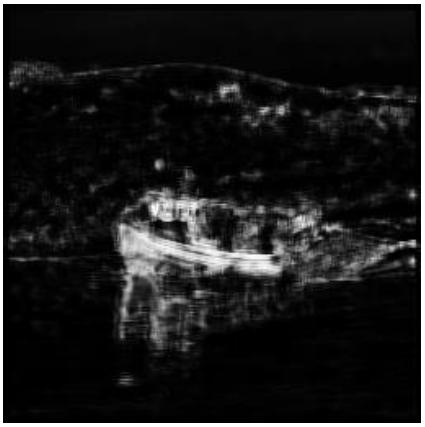
1. Влезната слика се обработува низ енкодерот, доловувајќи го контекстот и намалувајќи ги димензиите.
2. Карактеристиките на високо ниво се заробени во слоевите на тесно грло.
3. Декодерот ги зголемува примероците на функциите, со прескокнувачките врски што повторно воведуваат фини детали.
4. Конечниот излез е бинарна маска што го истакнува објектот во преден план.

Овој модел е одличен, но во случај кога имаме поголемо податочно множество. Овој модел е трениран во 50 епохи и добиена е прецизност од само 0.3959493592184112. Примери за предвидени маски од овој модел од тестирачкото множество:

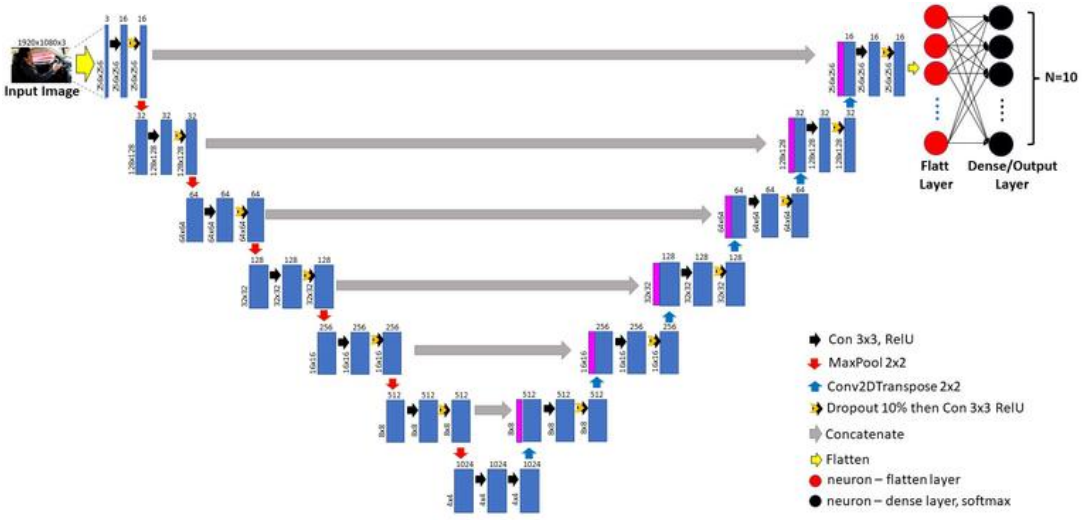
result_mask_0



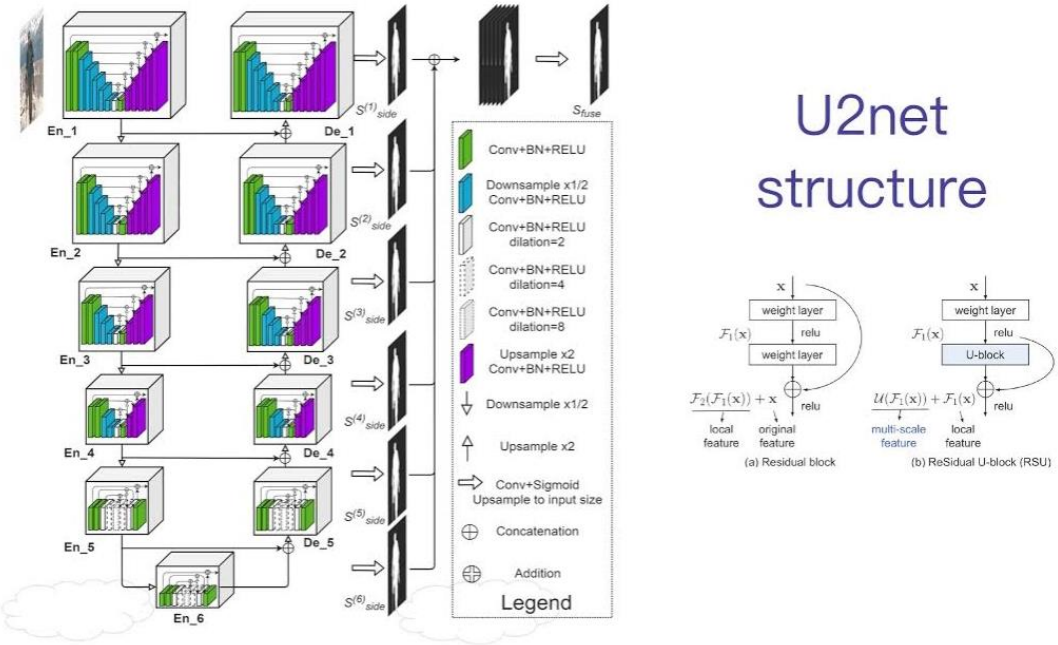
result_mask_30



result_mask_100



Приказ на слоевите на U^2-Net архитектурата



Приказ на слоевите на U^2-Net архитектурата

Модел 3 – DeepLabV3

DeepLabV3 е најсовремена архитектура на конволуционална невронска мрежа за семантичка сегментација на слики. Тој се надоврзува на неговиот претходник, DeepLabV2, со воведување на неколку подобрувања, вклучително и атрозна конволуција, Atrous Spatial Pyramid Pooling (ASPP) и структура на енкодер-декодер со прескокнувачки врски.

Еве детален преглед на архитектурата DeepLabV3:

1. Атрозна конволуција:

- DeepLabV3 користи атрозно извиткување (исто така познато како проширена конволуција) за да го зголеми приемното поле на филтрите без да го зголеми бројот на параметри.
- Atrous convolution овозможува мрежата да фати контекстуални информации во различни размери, што е од клучно значење за прецизна сегментација.

2. Atrous Spatial Pyramid Pooling (ASPP):

- ASPP е клучна компонента на DeepLabV3, дизајнирана за ефикасно снимање на контекстуални информации во повеќе размери.
- Се состои од паралелни атрозни конволуциони слоеви со различни стапки на дилатација, што му овозможува на моделот да ги доловува карактеристиките во повеќе размери.
- Излезите од овие конволуциони слоеви се конкатанирани и фузирани за да создадат сеопфатна претстава на влезната слика во различни размери.

3. Архитектура на енкодер-декодер:

- DeepLabV3 користи архитектура на енкодер-декодер за да ги усоврши резултатите од сегментацијата и да произведе ситно-гранулирани предвидувања.
- Модулот за енкодер извлекува карактеристики на високо ниво од влезната слика користејќи претходно обучена backbone мрежа (на пр. ResNet, MobileNet).
- Модулот за декодер ги зголемува примероците на мапите на карактеристики до оригиналната резолуција и вклучува прескокнувачки врски од енкодерот за зачувување на просторните информации.

4. Финален слој на класификација:

- Конечниот слој за класификација на DeepLabV3 вообичаено се состои од 1x1 конволутивен слој со softmax активација за да се создадат веројатности од класа по пиксели.
- За задачите за бинарна сегментација, како што се одземање на позадина или откривање на објект, бројот на излезни канали во последниот слој се прилагодува на 1, а сигмоидно активирање често се користи за да се произведе бинарна маска што укажува на присуство или отсуство на целната класа.

5. Тренирање и заклучок:

- DeepLabV3 вообичаено се тренира со користење на анотирани сетови на податоци на ниво на пиксели.
- За време на инференција, обучениот модел зема влезна слика и произведува маска за сегментација со предвидувања според пиксели.
- Може да се применат техники за пост-обработка како условни случајни полиња (CRF) за да се усовршат резултатите од сегментацијата и да се подобри точноста.

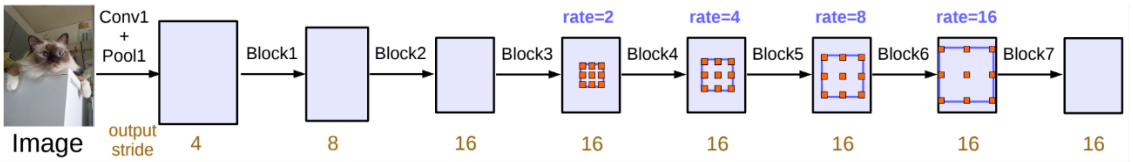
Апликации:

- DeepLabV3 е широко користен во различни задачи за компјутерска визија, вклучително и семантичка сегментација, откривање објекти, матирање на слики и анализа на медицинска слика.
- Неговата способност да доловува контекстуални информации од повеќе размери го прави особено ефикасен за задачи кои бараат прецизна локализација и детална сегментација.

Во овој проект, овој модел е дотрениран со истото податочно множество, така што од torch.hub е земен овој модел со ставка `pretrained=True` и е поставен како прв слој, но повторно се случи overfitting. Овој модел е трениран во 50 и во 100 епохи и добиени се следните прецизности:

```
Testing model with 50 epochs:
Accuracy: 0.0809347773001771
Testing model with 100 epochs:
Accuracy: 0.13323644903342985
```

Резултатите од овој модел се прелоши, така што овој модел не е искористен во демото.



Приказ на слоевите на DeepLabV3 архитектурата

Демо – апликација

За демо – апликацијата го искористен U-Net моделот кој е трениран во 50 епохи. Демо – апликацијата сама по себе е React апликација, каде frontend делот е дизајниран со TypeScript компоненти, додека пак од друга страна, за backend-от е искористен framework-от FastAPI кој што се справува со HTTP барањата и одговорите, а за серверот е искористен ASGI серверот uvicorn. Сето ова функционира на следниот начин:

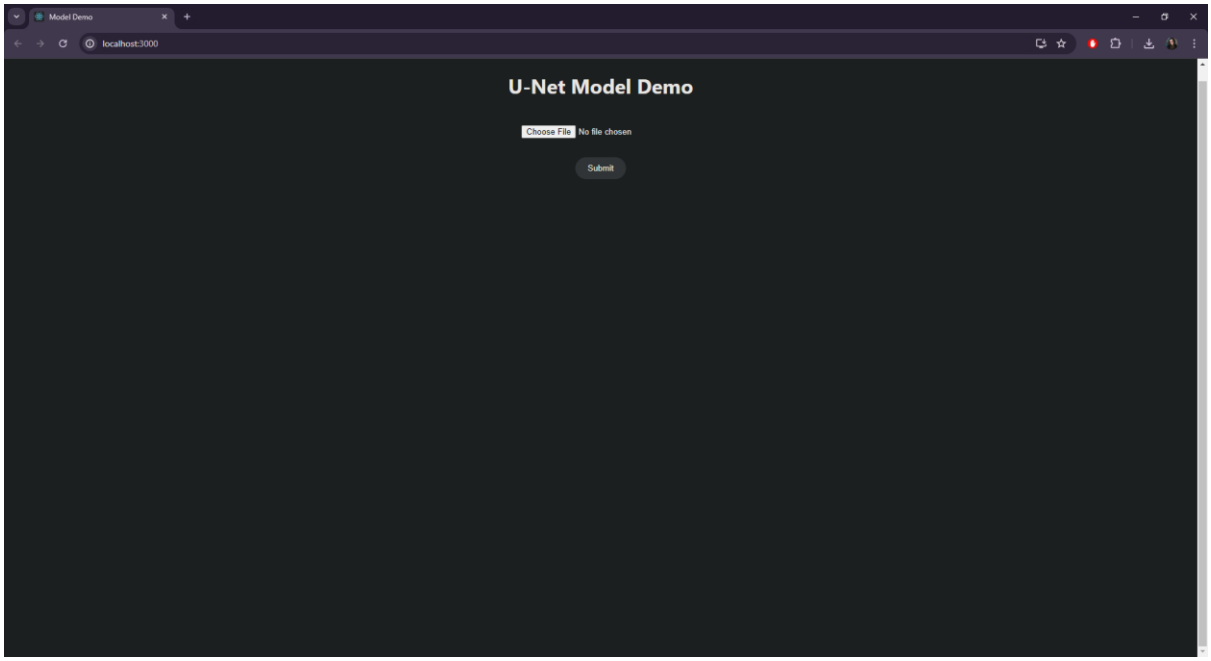
1. Клиентот ја поставува сликата:
   - Клиентот (React frontend) поставува датотека со слика преку барање POST до крајната точка `/predict/`.
2. Backend-от ја процесира сликата:
   - Поставената датотека со слика се чита и се претвора во PIL слика.
   - PIL сликата потоа се претвора во NumPy низа.
   - Оваа NumPy низа се пренесува на функцијата `result` од модулот `evaluate`, кој врши одземање на позадината.
   - Обработената слика се враќа во формат base64 за да може да се покаже на frontend-от.
3. Се враќа резултатот:
   - Backend-от ја враќа обработената слика во формат base64 на клиентот (React frontend).
4. Клиентот ја прикажува сликата:
   - Клиентот (React frontend-от) ја декодира низата base64 и ја прикажува резултатната слика.

За да се стартува демото треба да се извршат следните команди во терминал:

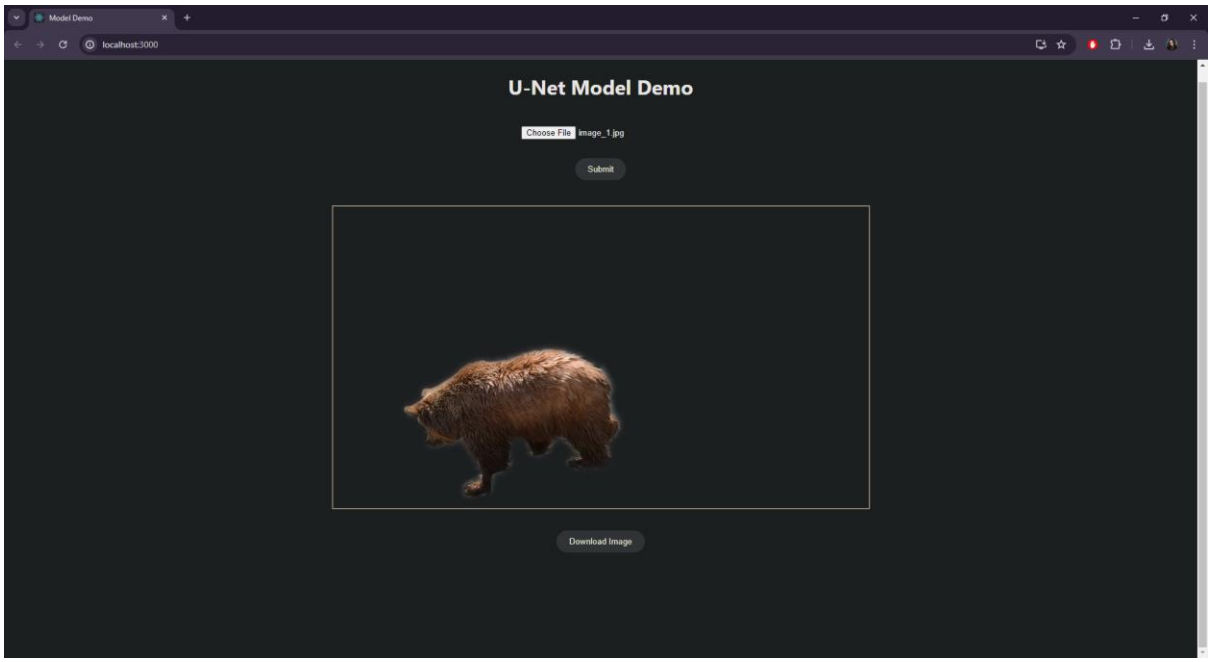
1. Во root фолдерот, треба да се изврши командата:
   - uvicorn main:app --reload
2. Во фолдерот `unet-demo` треба да се изврши командата:
   - npm start

Ќе го тестираме демото со оваа слика:

Изглед на frontend делот:



По кликање на копчето Choose File, селектирање на слика и кликање на копчето Submit:



Доколку кликнеме на копчето Download Image ќе ни се симне следната слика:

Користена литература

https://en.wikipedia.org/wiki/U-Net

https://medium.com/axinc-ai/u2net-a-machine-learning-model-that-performs-object-cropping-in-a-single-shot-48adfc158483

https://paperswithcode.com/method/deeplabv3

https://www.nvidia.com/en-us/glossary/pytorch/