Испитна питања из предмета Дубоко учење

1. ***[DL01a\_Uvod\_u\_duboko\_ucenje]* Шта је дубоко учење у контексту машинског учења и вештачке интелигенције? Које су главне карактеристике класичних система вештачке интелигенције, машинског учења и дубоког учења? Графички илустровати и објаснити.**

Дубоко учење (енг. Deep learning) је грана машинског учења која се бави развојем алгоритама и модела који опонашају рад људског мозга, користећи вештачке неуронске мреже са више слојева. Овај приступ се назива „дубоко” јер се састоји од више слојева вештачких неурона који постепено извлаче сложене репрезентације података.

A picture containing text, screenshot, font, design

Description automatically generated

Као што се на слици види, дубоко учење је део машинског учења које је опет део вештачке интелигенције. Вештачка – шах, унапред дефинисана правила..

Машинско – класификација спам мејлова. Дубоко учење – Препознавање слике

Формалнија разлика:

A picture containing text, screenshot, font, diagram

Description automatically generated

У процес **система вештачке интелигенције** је максимално укључен човек, од почетка до краја се пише програм који ради по правилима и унапред дефинисаним хеуристикама

**Машинско учење** – Учење на основу неких примера, битан део је препроцесирање које обавља човек док остали учења из података и извлачење информација ради алгоритам машинског учења.

**Дубоко учење** – End to end learning – структура сама учи из података тако што екстрактује обележја.

1. **[DL01b\_Image\_Classificaon\_Pipeline] Шта су хиперпараметри и која методологија поделе података на подскупове се препоручује за налажење оптималних вредности?**

Хипарпараметри су параметри који се користе за подешавање модела дубоког учења, али се њихове вредности не учествују директно из података. Они контролишу понашање самог модела, као што су број слојева у неуронској мрежи, број неуронских јединица у сваком слоју, брзина учења итд. Најчешће коришћена методологија за поделу података је тзв. подела на **тренинг** **валидациони** и **тест** скуп.

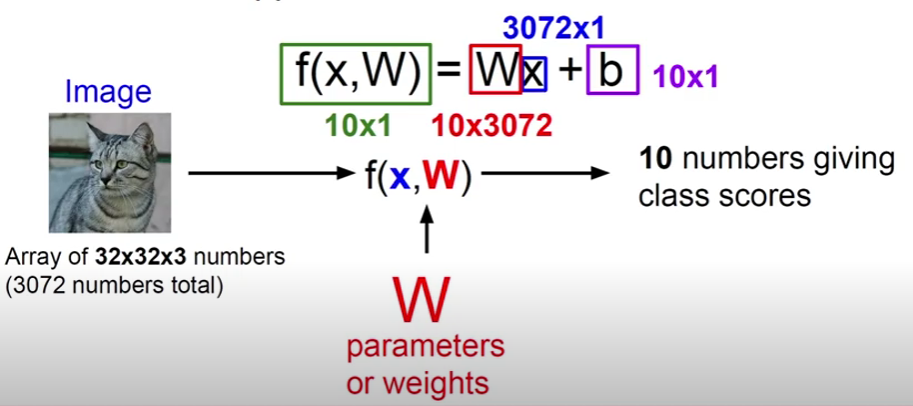
**Тренинг** **скуп**: Скуп података који се користи за тренирање модела. Овај скуп треба да буде довољно репрезентативан за све класе или категорије које модел треба да научи.

**Валидациони скуп**: Након сваке епохе или итерације током тренинга, користи се валидациони скуп да би се проценио перформансе модела и извршила оптимизација хипопараметара. Овај скуп помаже у спречавању пренаучености (енг. **overfitting**) и омогућава прилагођавање модела.

**Тест скуп**: Када је модел потпуно трениран и оптимизован, користи се тест скуп за коначно процењивање перформанси. Овај скуп садржи податке које модел није видео током тренинга или валидације, пружајући објективну процену способности модела да се генерализује на нове, непознате податке.

1. **[DL01b\_Image\_Classificaon\_Pipeline] Коју математичку операцију врши линеарни класификатор? Којих димензија су параметри који се користи за класификацију слика димензија 32x32x3 у 10 класа.**

Слика 32x32x3 се може репрезентовати као вектор од 3072 броја. Линеарни класификатор је функција **f(x,W) = Wx + b** где је W матрица која има колона колико и класа, а врста колико је вектор велики a x вектор. Уз линеарни класификатор постоји и праг окидања који је вектор величине броја класе (за сваку класу по 1).



1. **[DL01c\_Loss\_Funcons\_and\_Opmizaon]** **Која је улога регуларизационог члана код функције грешке? Како се рачуна за случај L1 и L2 регуларизације.**

Улога је да се спречи пристрасност (енг. **Overfitting**), односно да модел превише добро ради на тренинг скупу података а да су му на тест скупу перформансе јако лоше.

A picture containing text, font, white, typography

Description automatically generated

У случају L1 (лапласијанска) регуларизације, регуларизациони члан се рачуна као L1 норма (апсолутна вредност) параметара/ тежина модела

У случају L2 (еуклидовска) регуларизације, регуларизациони члан се рачуна као L2 норма (квадратни корен из суме квадрата) параметара/тежина модела.

1. **[DL01c\_Loss\_Funcons\_and\_Opmizaon]** **Шта представља и како се рачуна излаз somax слоја? Како се на основу њега рачуна функција грешке (енгл. *cross-entropy loss*).**

Излаз слоја Softmax представља вероватноће за сваку од могућих класа. Излазни вектор из претходног слоја се нормализује користећи експоненцијалну функцију. Нормализоване вредности се деле сумом свих нормализованих вредности. Ово се ради како би се осигурало да све вероватноће сумирају на јединицу.

A picture containing font, diagram, line, white

Description automatically generated

На основу излаза Softmax слоја, функција грешке се често рачуна користећи категоријалну унакрсну ентропију (categorical cross-entropy). Ова функција мери разлику између стварних класа и предвиђања неуронске мреже. Функција грешке се израчунава просто речено као логаритам вероватноће класе која тачна.

1. **[DL01c\_Loss\_Funcons\_and\_Opmizaon]** **На који начин се врши обучавање (промена параметара) методом градијентног спуста (енгл. *gradient descent method*)? Шта представља стохастички градијентни спуст (енгл. *Stohastic Gradient Descent - SGD*).**

Обучавање методом градијентног спуста (gradient descent) је поступак којим се мењају параметри модела у циљу минимизовања функције грешке. Ова метода користи градијент функције грешке у односу на параметре како би се одредио смер измене.

Општи поступак градијентног спуста обухвата следеће кораке:

1. Иницијализација параметара: Почетне вредности параметара се насумично иницијализују.
2. Пропагација унапред: Улазни подаци се прослеђују кроз модел, а израчунава се излаз.
3. Израчунавање градијента: Израчунава се градијент функције грешке у односу на параметре користећи backpropagation алгоритам. Овај корак укључује пропагацију грешке уназад кроз модел и израчунавање деловодних извода функције грешке по сваком параметру.
4. Ажурирање параметара: Параметри се ажурирају у супротном смеру градијента како би се смањила функција грешке. Величина корака ажурирања контролише се помоћу параметра који се назива стопа учења (learning rate).
5. Понављање поступка: Кораци 2-4 се понављају све док се не достигне услов заустављања, на пример, одређени број итерација или критеријум конвергенције.

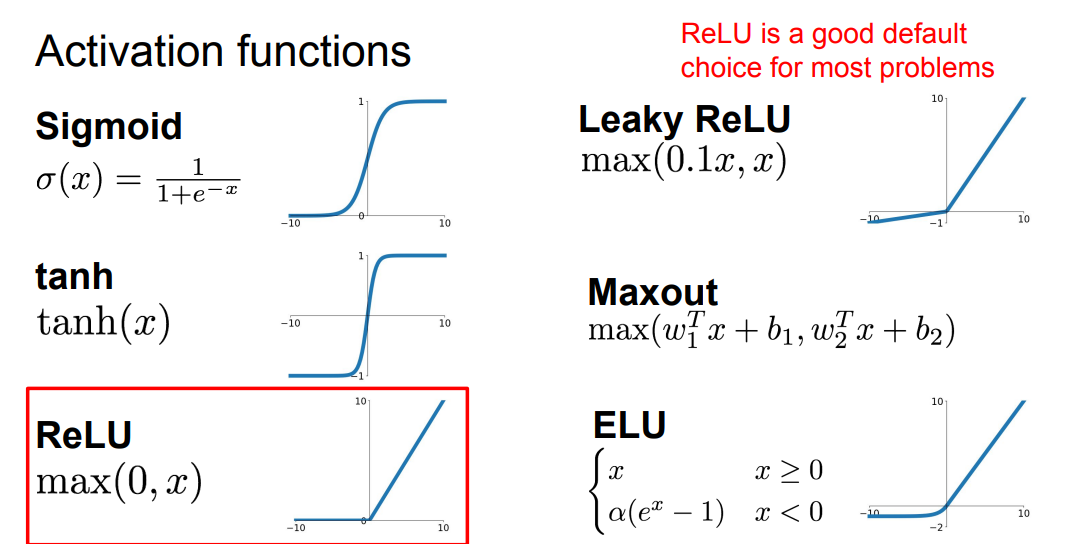
Стохастички градијентни спуст (Stochastic Gradient Descent - SGD) је варијанта градијентног спуста у којој се у свакој итерацији користи само један узорак (или мали подсет узорака) из скупа за тренирање за израчунавање градијента и ажурирање параметара. Ово се разликује од стандардног градијентног спуста који користи све узорке из скупа за тренирање. Коришћење једног узорка или мањег подсета узорака чини стохастички градијентни спуст бржим за израчунавање градијента и ажурирање параметара, али може бити мање прецизан од стандардног градијентног спуста.

1. **[DL02\_Neural\_Networks\_and\_Backpropagraon]** **Написати математичку функцију по којој се рачуна излаз неуронске мреже са једним скривеним слојем који користи ReLU активациону функцију.**

A black text on a white background

Description automatically generated with low confidence

1. **[DL02\_Neural\_Networks\_and\_Backpropagraon]** **Нацртати графике следећих активационих функција: сигмоидална, тангенс хиперболички, ReLU, Leaky ReLU и ELU.**



1. **[DL02\_Neural\_Networks\_and\_Backpropagraon] Нацртати граф израчунавања (енгл.**

***computational graph*) за функцију 𝑓𝑓 = (𝑥𝑥 + 𝑦𝑦)𝑧𝑧 и одредити парцијалне изводе 𝜕𝜕𝑓𝑓 , 𝜕𝜕𝑓𝑓 , 𝜕𝜕𝑓𝑓**

**𝜕𝜕𝜕𝜕 𝜕𝜕𝜕𝜕 𝜕𝜕𝑧𝑧 када су вредност улаза 𝑥𝑥 = 3, 𝑦𝑦 = −2, 𝑧𝑧 = 5.**

Derivative calculator

A picture containing text, line, diagram, font

Description automatically generated

1. **[DL02\_Neural\_Networks\_and\_Backpropagraon]** **Графички илустровати четири шаблона тока градијената (енгл. *patterns in gradient flow*).**

A picture containing text, diagram, screenshot, font

Description automatically generated

1. **[DL03\_Convoluonal\_Neural\_Networks] Графички представити и описати архитектуре LeNet5 мреже. Који све типови слојева се у њој појављују и која је њихова улога?**

A picture containing text, diagram, font, line

Description automatically generated

1. Улазни слој (Input Layer): Слично 32x32 пиксела представљајући слику цифре.
2. Конволуциони слој (Convolutional Layer, C1): Користи скуп конволуционих филтера за обављање операција конволуције на улазној слици. Ови филтери детектују локалне особине као што су ивице и облици
3. Subsampling слој (Subsampling Layer, S2): Примењује pooling на излазима конволуционог слоја. Ова операција смањује просторне димензије слике и доприноси инваријанцији на померање.
4. Конволуциони слој (Convolutional Layer, C3): Овај слој такође користи конволуционе филтере, али се сада примењују на излазима subsampling слоја. Овај слој примењује више филтера и омогућава детекцију вишег нивоа особина.
5. Subsampling слој (Subsampling Layer, S4): Исто као и претходни subsampling слој, али се сада примењује на излазима C3 конволуционог слоја.
6. Потпуно повезани слој (Fully Connected Layer, C5): Овај слој се састоји од потпуно повезаних неурона. Сваки неурон у овом слоју је повезан са свим неуронима из претходног слоја.
7. Излазни слој (Output Layer, F6): Коначни потпуно повезани слој који представља излазни вектор вероватноћа за сваку класу.

Улога конволуционих слојева (C1, C3) је да детектују локалне особине и облике на слици. Subsampling слојеви (S2, S4) смањују димензије слике и помажу у идентификацији битних особина. Потпуно повезани слојеви (C5, F6) врше класификацију на основу издвојених фичера.

1. **[DL03\_Convoluonal\_Neural\_Networks]** **Која је улога и на који начин се формира излаз конволуционог слоја код конволуционих неуронских мрежа (илустровати на примеру)? Колика се мапа на излазу добија уколико конволуциони слој садржи 6 филтара 5x5x3, а на улазу се доведе слика димензија 32x32x3?**

Улога конволуционог слоја у конволуционим неуронским мрежама је да изврши операцију конволуције над улазним подацима и филтерима. Ова операција има за циљ да детектује карактеристичне вредности у улазним подацима и формира активациону мапу као излаз слоја.

Када се на улаз конволуционог слоја доведе слика димензија 32x32x3, а слој садржи 6 филтара димензија 5x5x3, излазни подаци ће бити форматирани као мапа димензија 28x28x6. Ово се дешава јер се сваки филтер пролази кроз улаз са примењивањем операције конволуције, што резултује мапом од 6 канала димензија 28x28. Тако да на излазу имамо 6 таквих мапа, што доводи до укупне димензије излаза 28x28x6.

A picture containing text, diagram, screenshot, line

Description automatically generated

Илустрација:

Улаз (32x32x3) --> Конволуција (6 филтара 5x5x3) --> Излаз (28x28x6)

1. **[DL03\_Convoluonal\_Neural\_Networks]** **Која је улога и на који начин се формира излаз *pooling* слоја код конволуционих неуронских мрежа (илустровати на примеру)? Колика се мапа на излазу добија уколико се на улаз *max* *pooling* слоја величине кернела 3x3 са померањем (енгл. *stride*) 2, доведе мапа димензија 32x32x3?**

A picture containing text, screenshot, diagram

Description automatically generated

Пулинг слој има за циљ смањити димензије активационих мапа и сумирањем информација из региона улазне мапе да креира излазну мапу мањих димензија. Најчешће коришћена врста пулинга је максимално пулирање (енгл. max pooling).

Процес максималног пулирања се врши следећим начином:

За сваки регион улазне мапе који је прозор димензија 3x3 (у случају кернела 3x3), излазни пиксел се рачуна као највећа вредност у том региону.

Димензије излазне мапе се рачунају на основу размере улазне мапе и величине померања (stride). У случају када се користи кернел 3x3 са померањем 2, димензије излазне мапе се рачунају на следећи начин:

Ширина излазне мапе: (ширина улазне мапе - величина кернела) / померање + 1 = (32 - 3) / 2 + 1 = 15

Висина излазне мапе: (висина улазне мапе - величина кернела) / померање + 1 = (32 - 3) / 2 + 1 = 15

Број канала на излазу остаје исти (3 у овом случају).

Дакле, у овом примеру би се добила излазна мапа димензија 15x15x3. Мапа се смањује у ширини и висини, док број канала остаје непромењен. Путем пулинг слоја, мрежа учи да изабере најзначајније информације из сваког региона улаза, што може допринети инваријантности на промене улошка и специфичностима мапе.

1. **[DL04\_Training\_Neural\_Networks\_I]** **Због чега се користи и на који начин се спроводи *batch* нормализација активационих мапа.**

Batch нормализација је техника која се користи за нормализацију активационих мапа. Улога batch нормализације је умирити расподелу активација у скупу података који се обради у једном проласку кроз мрежу. То омогућава стабилније и брже обучавање мреже. Процес batch нормализације се извршава након израчунавања активација конволуционог или потпуно повезаног слоја, и прелази кроз следеће кораке:

1. Израчунавање средње вредности и стандардне девијације за сваки канал
2. Нормализација активација
3. Нормализоване активације се растерете користећи параметре који се оптимизују током тренинга мреже.
4. **[DL04\_Training\_Neural\_Networks\_I]** **Које све врсте нормализација постоје. Графички представити шта која од њих нормализује?**

Постоје различите врсте нормализације у контексту машинског учења и дубоког учења. Ево неколико примера:

1. **Batch Normalization**: Нормализује активације по сваком узорку у бечу.
2. **Layer Normalization**: Нормализује активације по сваком слоју унутар једног узорка.
3. **Instance Normalization**: Нормализује активације по сваком узорку унутар једног слоја.
4. **Group Normalization**: Нормализује активације по групама у слоју, где групе могу бити дефинисане према одређеном критеријуму (на пример, по пикселима у конволуционом слоју).

A picture containing diagram, line, design

Description automatically generated

1. **[DL05\_Training\_Neural\_Networks\_II] Који су проблеми код стохастичког градијентног спуста (енгл. SGD) и како се превазилазе коришћењем момента?**

Проблеми kod стохастичког градиjентног спуста (SGD) укључују:

1. Спора конвергенција: SGD може бити спор при конвергенцији ка оптималном решењу, посебно у случају функција грешке са дубоким, тесним или нерегуларним минимумима.
2. Осетљивост на избор параметра: Параметри као што су величина корака (learning rate) и инцијални параметри модела могу значајно утицати на перформансе SGD. Погрешан избор параметара може довести до споре конвергенције или стагнације у локалним минимумима.
3. Заглављивање у локалним минимумима: SGD може бити склоњен да се заустави у локалним минимумима функције грешке, недостижући оптимално глобално решење.

Момент (енгл. momentum) је техника која се користи да превазиђе ове проблеме. Она доприноси бржој конвергенцији и избегавању заглављивања у локалним минимумима. Ефекат момента се постиже тако што се уведе нова величина (момент) која акумулира претходне градијенте и додатно утиче на ажурирање параметара модела. Конкретно, момент промени промену параметара у складу са претходним градијентима и брзином која се контролише параметром момента.

Примена момента у SGD има следеће предности:

1. Бржа конвергенција
2. Превазилажење локалних минимума
3. Стабилније ажурирање параметара
4. **[DL05\_Training\_Neural\_Networks\_II] У чему је специфичност Несторовог момента у односу на стандардни момент и због чега може да да боље резултате?**

**Несторов момент** је модификација стандардног момента која се користи у алгоритмима оптимизације, посебно у контексту стохастичког градијентног спуста. Специфичност Несторовог момента узима се у обзир градијент како би прилагођавао параметe мреже.

Код стандардног момента, тренутни градијент се комбинује са претходним кораком ажурирања, при чему се параметри мреже помножени претходним кораком кретања "моментумом" додају тренутном градијенту. Ово помаже у убрзању конвергенције алгоритма оптимизације, али може довести до превеликих корака у ситуацијама када се апроксимација градијента у близини минимума покаже нетачном.

Несторов момент решава овај проблем тако што прво користи претходни корак ажурирања да "предвиди" следећи корак и тада израчунава тренутни градијент. На основу тог предвиђања, ажурира се параметар мреже. Ова техника омогућава да се избегну превелики кораци у близини минимума и побољшава конвергенцију.

Због ове специфичности, Несторов момент може да пружи боље резултате од стандардног момента, посебно у ситуацијама када постоји велика варијација у градијентима или када се апроксимација градијента показује нетачном у близини минимума.

1. **[DL05\_Training\_Neural\_Networks\_II] Навести који све алгоритми за оптимизацију се користе и који се сматра најбољим (најбржим)?**

Постоје разни алгоритми за оптимизацију у контексту тренирања неуронских мрежа. Неки од најпопуларнијих алгоритама укључују:

1. Стохастички градијентни спуст (SGD)
2. Моментум
3. Несторов моментум
4. Adagrad
5. RMSprop
6. Adam

Aлгоритми попут Адама (Adam) су често се називају "најбржим" због њихове ефикасности и добрих резултата у великом броју задатака тренирања мрежа. Адам комбинује предности Mоментума и RMSprop-a и обично добија добре резултате без потребе за ручним тјунирањем хиперпараметара.

1. **[DL05\_Training\_Neural\_Networks\_II] Објаснити технику одбацивања (енгл. *dropout*) и навести зашто се користи. На који начин се врши у току обучавања, а како касније у току експлоатације мреже?**

Техника одбацивања (dropout) је регуларизацона техника која се користи у тренирању неуронских мрежа и примењује се на скривене слојеве мреже и има за циљ смањити overfitting.

У току обучавања, при проласку примера кроз мрежу, сваки неурон у скривеним слојевима (укључујући улазни слој) са вероватноћом p (често постављеном на 0,5) се случајно искључује или "одбацује".

У току експлоатације мреже, када се користи за предвиђање нових примера, одбацивање се не примењује. Међутим, излазни слој се скалира са вероватноћом p како би се компензовао губитак нерона. Ово је потребно ради одржавања сличног очекиваног излаза у току обучавања и експлоатације.

1. **[DL05\_Training\_Neural\_Networks\_II]** **Објаснити технику проширивања података (енгл. *data augmentation*) и навести зашто се користи. На који начин се врши у току обучавања, а како касније у току експлоатације мреже?**

**Техника проширивања података** је поступак где се постојећи скуп података ученика проширује генерисањем нових примера података путем различитих трансформација или модификација. Ова техника се користи у обучавању машинског учења и дубоког учења са циљем повећања разноврсности података и уопштавања модела.

У току обучавања, проширивање података се примењује на скуп тренинг података. Примери података се модификују на различите начине, као што су ротација, зумирање, промена осветљења, замагљивање итд. Овим се генеришу нови примери који су слични оригиналним, али су у неком смислу "различити" и пружају већу разноврсност.

Касније, у току експлоатације мреже (тј. при коришћењу обученог модела за предвиђање), проширени подаци се обично не користе. Само оригинални, неизмењени подаци се подносе моделу за добијање предвиђања. Проширивање података се користи искључиво за обогаћивање тренинг скупа и побољшање перформанси модела.

1. **[DL06\_CNN\_Architecture]** **Шта представља и како се обавља пренесено учење (енгл. *transfer learning*)?**

**Пренесено учење** се односи на технику машинског учења где се знање стечено обучавањем једног модела на одређеном задатку преноси и примењује на други сродни задатак. Уместо да обучавате модел од нуле на новом задатку, трансферно учење користи унапред обучено знање из и зворног задатка да би се побољшало учење и учинак на циљном задатку.

Процес трансферног учења обично укључује следеће кораке:

**Pre-training**: Модел се обучава на великом скупу података за изворни задатак, као што је класификација слика, где учи да извуче релевантне карактеристике и обрасце из података. Овај корак пре обуке се често изводи на великом, општем скупу података као што је ImageNet.

**Fine-tuning**: Унапред обучени модел се затим прилагођава или фино подешава на мањем скупу података специфичном за циљни задатак. Последњих неколико слојева претходно обученог модела се често замењују или поново обучавају, док се ранији слојеви, који обухватају више генеричких карактеристика, држе замрзнути или ажурирани са нижом стопом учења.

1. **[DL06\_CNN\_Architecture]** Који су основни архитектурни елементи карактеристични за VGGNet?

VGGNet је архитектура неуронске мреже позната по својој једноставности и ефикасности. Главни архитектонски елементи карактеристични за ВГГНет су:

* 1. **Конволуциони слојеви**: VGGNet се састоји од низа конволуционих слојева наслаганих један на други. Ови слојеви изводе операције конволуције да би издвојили локалне карактеристике из улазне слике. VGGNet обично користи 3х3 конволуционе филтере са кораком од 1, а број филтера се повећава са дубином мреже.
  2. **Max Pooling слојеви**: Након сваког скупа конволуционих слојева, VGGNet примењује Max Poolingслојеве. Max Poolingсмањује просторне димензије мапа обележја уз задржавање најважнијих информација. VGGNet користи Max Poolingса прозором 2х2 и кораком од 2, што преполови просторне димензије.
  3. **Потпуно повезани слојеви**: При крају мреже, VGGNet се састоји од потпуно повезаних слојева. Ови слојеви узимају карактеристике претходних конволуционих слојевима и трансформишу их у вероватноће класа. Потпуно повезане слојеве обично прати софтмакс функција активације за вишекласну класификацију.
  4. **ReLU**: VGGNet користи функцију активације исправљене линеарне јединице (**ReLU**) након сваког конволуционог и потпуно повезаног слоја. **ReLU** уводи нелинеарност у мрежу, омогућавајући јој да научи сложене обрасце и побољшавајући способност мреже да генерализује.
  5. **Дубока архитектура**: VGGNet карактерише дубока архитектура са великим бројем слојева. Оригинални VGGNet има 16 или 19 слојева, у зависности од варијанте. Дубина мреже омогућава јој да ухвати сложене карактеристике и научи хијерархијске репрезентације.

1. **[DL06\_CNN\_Architecture]** **Графички представити *inception* модул који представља основу GoogLeNet архитектуре и објаснити идеју која је ту примењена.**

A picture containing text, screenshot, line, diagram

Description automatically generated

* Победник ImageNet такмичења 2014. године са грешком од 6.7%
* 22 Слојева
* Без тотално повезаних слојева
* 5м параметара, што је 12х мање од AlexNet

Конкретно, **inception** модул користи комбинацију 1x1 конволуције, 3x3 конволуције, 5x5 конволуције и 3x3 макс-пулинг операције. Ова комбинација омогућава моделу да изучава фичере на различитим резолуцијама и различитим нивоима апстракције паралелно, а затим их конкатенира. 1x1 конволуција се користи за смањење димензионалности, док се 3x3 и 5x5 конволуције користе за детекцију различитих фичера на различитим грануларностима. Макс-пулинг се користи за извлачење најзначајнијих фичера.

1. **[DL06\_CNN\_Architecture]** Графички представити структуру резидуалног блока који се користи код ResNet архитектуре и описати предности његовог коришћења.

A picture containing text, diagram, font, screenshot

Description automatically generated

У резидуалном блоку, улазни сигнал пролази кроз два слоја конволуције (3x3 конволуције), док се истовремено пролази и преко скаларне везе (shortcut connection) која преноси неизменјену информацију са улаза на излаз блока. Крајњи излаз блока је добијен сабирањем излаза конволуцијских слојева и скаларне везе. Овај излаз се затим прослеђује на следећи резидуални блок или на крајњи излаз мреже.

Предности коришћења резидуалног блока у ResNet архитектури су:

Превазилажење проблема деградације: Дугачке невронске мреже су склоње деградацији перформанси, где додавање додатних слојева може довести до губитка тачности. Резидуални блок омогућава ефикасно учење диференцијалних сигнала, што олакшава учење дубоких мрежа и супериорну перформансу.

Оптимизација тренирања: Резидуални блок омогућава лакше преносење информација кроз блокове без губитка или измештања. Ово олакшава тренирање мреже и спречава проблеме као што су прекомерно учење и проблеми са градијентима.

Побољшана репрезентација фичера: Резидуални блок омогућава мрежи да научи различите нивое апстракције и репрезентације фичера, што доводи до боље различитости и универзалности у решавању различитих задатака.

Скалирање и примена: Резидуални блок се може лако скалирати и применити у различитим архитектурама и задацима. Могуће је додавати више блокова за дубље мреже и применити их у различитим комбинацијама, што омогућава велику флексибилност у дизајнирању мреже.

Ове предности чине резидуални блок битним елементом у ResNet архитектури и омогућавају успешно тренирање дубоких невронских мрежа.

1. **[DL07\_Detecon\_and\_Segmentaon]** **Која четири базична задатка рачунарског вида? Скицирати и описати укратко сваки од њих.**

A screenshot of a computer

Description automatically generated with low confidence

**Computer Vision Tasks**

* 1. **Classification** – Проблем класификације је одредити која класа се налази на слици, односно препознати којој класи припада објекат који је на слици.
  2. **Semantic Segmentation** – Семантичка сегментација је тежи проблем и она је фокусирана на пикселе, односно говори који пиксел припада којој површи.
  3. **Object Detection** – Детекција објеката на слици, могућност детектовања више објеката на слици, препознавање којој класи детектовани објекат припада и уовичавање површи на којој је детектован објекат
  4. **Instance Segmentation** – Се односи на пикселе и означава све пикселе који припадају детектованом објекту. Такође могућност детектовања више објекта, односно сегментација више објеката на слици

1. **[DL07\_Detecon\_and\_Segmentaon]** **На који начин се формирају потпуно конволуционе мреже за семантичку сегментацију? Која два основна елемента оне поседују?**

Потпуно конволуционе мреже за семантичку сегментацију се формирају модификовањем конволуционих неуронских мрежа да би се извршила класификација по пикселима и сегментација слике. Они поседују два основна елемента:

* Конволуцијски слојеви: ФЦН користе конволуционе слојеве за издвајање карактеристика из улазне слике. Ови слојеви примењују различите филтере на слику како би издвојили локалне карактеристике уз очување просторних информација.
* Транспоновани конволуцијски слојеви: Ови слојеви се користе за повећање узорковања излаза конволуционих слојева да би се добила мапа сегментације са истом резолуцијом као и улазна слика. Транспоновани конволуцијски слојеви су способни да науче да разматрају пикселе у више резолуција и да изврше прецизно мапирање пиксела у сегмент.

Другим речима, поседују downsampling и upsampling елементе, један за смањење резолуције и класификацију, други за повећање резолуције ради сегментације односно класификације пиксела.

1. **[DL07\_Detecon\_and\_Segmentaon]** Која је улога и на који начин се спроводи транспонована конволуција (енгл. *transpose convolution*)?

Основна намена транспоноване конволуције је да покрене поступак усмерен на повратак са нискорезолуционог простора на вишерезолуциони простор, односно да омогући прозиводњу излаза већих димензија од улаза.Ова операција је слична операцији конволуције, само што за разлику од конволуције се улазна слика множи транспонованим тежинама (филтером).

Транспонована конволуција има различите примене у обради слика, укључујући сегментацију слика, генерисање слика високе резолуције, реконструкцију

Користећи транспоновану конволуцију, могуће је повратити информације које су изгубљене приликом усклађивања резолуције и обавити детаљнију сегментацију или генерацију слика. Ова операција је битна компонента у GAN и архитектурама као што је U-Net.

1. **[DL07\_Detecon\_and\_Segmentaon]** Која је основна архитектура конволуционе мреже за класификацију и локализацију? Графички илустровати и назначити разлику у односу на конволуциону мрежу за класификацију.

Користи се вектор(4096 – Алекснет) и на основу њега се добијају информације. Значи класична архитектура с тим што се тренира мрежа да на основу 4096 вектора имамо две главе једна за препознавање класе а друга за локализацију. Оне имају два лоса, они се касније сабирају...

A picture containing text, screenshot, diagram, font

Description automatically generated

1. **[DL07\_Detecon\_and\_Segmentaon]** Скицирати и објаснити принцип рада *R-CNN* мреже за детекцију објеката. Који су недостаци ове архитектуре?

A picture containing text, screenshot, diagram

Description automatically generated

Алгоритам извлачи регионе од интереса из слике (око 2000). Такви региони се resize-ују на величину 224х224 а затим сви пропуштају кроз конволуциону мрежу. Након пропуштања кроз мрежу врши се класификација региона помоћу SVM-a. Проблем ове архитектуре је што је веома спора јер је потребно провући кроз мрежу 2000 региона за сваку слику.

1. **[DL07\_Detecon\_and\_Segmentaon]** Скицирати и објаснити принцип рада *Fast R-CNN* мреже за детекцију објеката. Шта је главна разлика у односу на обичну *R-CNN* мрежу?

A picture containing text, screenshot, design

Description automatically generated

1. **[DL07\_Detecon\_and\_Segmentaon]** Скицирати и објаснити принцип рада *Faster R-CNN* мреже за детекцију објеката. Шта је главна разлика у односу на *Fast R-CNN* мрежу?

A picture containing text, screenshot, parallel, design

Description automatically generated

1. **[DL07\_Detecon\_and\_Segmentaon]** Објаснити принцип рада једнофазних детектора објеката (енгл. *Single-Stage Object Detectors*) као што су *YOLO*, *SSD* и *RetinaNet*. У чему је главна разлика у односу на Faster R-CNN мрежу?

Једнофазни детектори објеката, као што су YOLO (You Only Look Once), SSD (Single Shot MultiBox Detector) и RetinaNet, представљају архитектуре које обављају детекцију објеката у једном проходу кроз мрежу, за разлику од двофазних детектора као што је Faster R-CNN.

Принцип рада једнофазних детектора објеката се може описати на следећи начин:

YOLO: YOLO делити улазну слику на мрежну матрицу и предвиђа области и класе објеката у истом проходу. Мрежа YOLO је састављена од конволуционих слојева који предвиђају класе и границе около објеката. Ово омогућава брзу и ефикасну детекцију објеката, али може бити мање прецизна у односу на двофазне детекторе.

SSD: SSD такође користи конволуциону мрежу, али укључује и додатне конволуционе слојеве са различитим резолуцијама. Ово омогућава детекцију објеката различитих величина и пропорција. SSD предвиђа класе и границе објеката на различитим нивоима резолуције, што му омогућава да брже и прецизније детектује објекте.

RetinaNet: RetinaNet је архитектура која решава проблем несиметричне расподеле објеката различитих величина у скупу података. Она укључује мрежу са две гране: одговорну за предвиђање класа објеката и одговорну за предвиђање граница около објеката. У циљу решавања проблема несиметричне расподеле, користи се алгоритам постављања тежине (Focal Loss), који даје већи значај ретким класама објеката.

Главна разлика између једнофазних детектора и двофазних детектора као што је Faster R-CNN је у брзини и ефикасности. Једнофазни детектори објеката обављају детекцију у једном проходу кроз мрежу, што их чини бржим и погоднијим за реално временске примене. С друге стране, двофазни детектори као што је Faster R-CNN користе два посебна прохода - један за генерисање предлога региона и други за класификацију и регресију. Ово обезбеђује већу прецизност у детекцији, али може бити спорије у односу на једнофазне детекторе.

1. **[DL08\_Recurrent\_Neural\_Networks]** Која је главна идеја код рекурентних неуронских мрежа која их разликују од „обичних“ неуронских мрежа? За које проблеме се ове мреже користе?

Главна идеја која стоји иза рекурентних неуронских мрежа (РНН) која их издваја од „обичних“ неуронских мрежа је њихова способност да обрађују секвенцијалне податке, при чему излаз зависи не само од тренутног улаза већ и од претходних стања мреже.

РНН-ови су дизајнирани да хватају контекстуалне зависности у секвенцијалним подацима и генеришу излазе на основу информација из претходних улаза. Обично се користе за задатке као што су обрада природног језика, препознавање говора, генерисање текста, анализа осећања, предвиђање временских серија и друге задатке који захтевају разумевање контекста и динамике у секвенцијалним подацима.

Кључна компонента РНН-а је рекурентни слој, који има повратне везе које омогућавају да се претходно стање користи у обради тренутног улаза. Ово омогућава мрежи да задржи информације о претходним стањима и да их користи за доношење одлука или генерисање резултата. Ова способност чини РНН моћним алатом за моделирање секвенцијалних података и разумевање њиховог контекста и зависности.

У поређењу са „обичним“ неуронским мрежама, главна разлика РНН-ова је њихова способност да рукују секвенцијалним подацима и хватају временске зависности, што их чини веома погодним за задатке који укључују временске серије или секвенцијалне улазе.

1. **[DL08\_Recurrent\_Neural\_Networks]** Скицирати начин рада једноставне рекурентне мреже за превођење текста са једним скривеним слојем и написати математичку формулу по којој се формира излаз овакве мреже?

A picture containing text, screenshot, diagram, font

Description automatically generated

1. **[DL08\_Recurrent\_Neural\_Networks]** Графички илустровати и објаснити на који начин се врши обучавање рекурентних неуронских мрежа коришћењем *backpropagration* алгоритма? Како се назива ова варијанта *backpropagration* алгоритма?

Да бисмо обучили рекурентне неуронске мреже (РНН) користећи пропагацију уназад, користимо варијанту која се зове бацкпропагатион кроз време (БПТТ). Процес обуке укључује следеће кораке:

Проширивање унапред: Улазни подаци се обрађују кроз РНН, генеришући излаз у сваком временском кораку.

Прорачун губитка: Излаз се упоређује са циљним излазом, а функција губитка мери разлику.

Проширивање уназад: Градијент се израчунава тако што се пропагира уназад кроз време од последњег до првог временског корака.

Ажурирање параметара: градијенти се користе за ажурирање параметара РНН-а, минимизирајући губитак.

БПТТ управља временском природом РНН-ова тако што размотава мрежу кроз време. Међутим, БПТТ може бити рачунарски скуп и пати од проблема нестајања градијента. За решавање ових проблема користе се технике као што су одсецање градијента и специјализоване РНН архитектуре (као што су ЛСТМ и ГРУ).

A diagram of a flowchart

Description automatically generated with low confidence

1. **[DL08\_Recurrent\_Neural\_Networks]** Који проблеми су карактеристични за обичне рекурентне неуронске мреже и које друге архитектуре су уведене у циљу превазилажења?

Уобичајени проблеми повезани са редовним рекурентним неуронским мрежама (РНН) укључују проблем нестајања градијента и потешкоће у учењу дугорочних зависности. Да би се превазишли ови проблеми, уведено је неколико алтернативних архитектура. Неке од ових архитектура су:

Дуготрајна краткорочна меморија (ЛСТМ): ЛСТМ је врста рекурентне неуронске мреже која укључује меморијске ћелије и капије за контролу тока информација. Он се бави проблемом градијента који нестаје и омогућава боље учење дугорочних зависности.

Гатед Рецуррент Унит (ГРУ): ГРУ је још једна врста рекурентне неуронске мреже слична ЛСТМ. Такође користи капије да регулише проток информација и скривена стања. ГРУ се често сматра лакшом и бржом алтернативом ЛСТМ-у, док је и даље ефикасан у учењу дугорочних зависности.

Трансформер: Првобитно развијен за задатке обраде природног језика, Трансформер архитектура је била успешна у домену секвенцијалног учења. Уместо рекурентних слојева, Трансформатори користе механизам пажње за моделирање зависности између улазних и излазних података. Трансформери су показали снажне перформансе у различитим задацима, укључујући генерисање текста и превођење језика, ефикасно хватајући дугорочне зависности.

Ове архитектуре имају за циљ да превазиђу изазове повезане са редовним РНН-овима и побољшају моделирање дугорочних зависности. Они су значајно унапредили поље понављајућег учења и омогућили моделима да ефикасно хватају зависности у секвенцијалним подацима

1. **[DL09\_Generave\_Models]** Графички илустровати и описати архитектуру аутоенкодера? На који начин се врши њихово обучавање? Која функција грешке (енгл. *loss*) се користи?

Autoenkoder je veštačka neuronska meža koja se koristi za učenje efikasnog kodiranja. Imaju isti broj ulaznih i izlaznih neurona i nekoliko skrivenih slojeva. Uče da rekonstruišu ulaz i to dovode na izlaz. Glavna osobina autoenkodera je učenje fičera ili osobina iz ulaznih neoznačenih podataka.

A diagram of a function

Description automatically generated with low confidenceA picture containing circle, diagram, line, pattern

Description automatically generated

Postoje 2 dela autoenkodera:

* **Enkoder** – deo mreže koji uči efikasno enkodiranje podataka
* **Dekoder** – deo mreže koji uči efikasno dekodiranje (rekonstrukciju) podataka koje mu enkoder daje

**Bottleneck (latentni prostor)** – srednji skriveni sloj koji je mnogo manje dimenzionalnosti od ulaznog i izlaznog  sloja i sadrži kompresovanu reprezentaciju podataka. Svi slojevi su potpuno povezani.

**Obučavanje**: Počinjemo skupom ulaznih podataka **X** koje enkoder mapira u skup fičera **Z**. Vrši se redukcija dimenzionalnosti pa je Z uvek znatno manji od X jer predstavlja najbitnije fičere iz skupa X. najbitniji fičeri su oni koji pamte bitne faktore varijacija u podacima. Druga mreža koju dodajemo (istog tipa) je dekoder i on nam daje izlaz istih dimenzija kao što je ulaz X. Koristi se **L2 Loss funkcija** ili Squared Error Loss funkcija (kvadratna razlika prave vrednosti i predikcije, za svaki podatak dataset-a).

A picture containing font, text, white, line

Description automatically generated

1. **[DL09\_Generave\_Models]** Графички представити и описати која је специфичност варијационих аутоенкодера (енгл. *Variational Autoencoders – VAE*) у односу на обичне аутоенкодере. Која им је основна намена и шта су предности и недостаци њиховог коришћења?

Varijacioni autoenkoder (VAE) je neuronska mreža koja ima istu arhitekturu kao autoenkoder, ali razlika između autoenkodera i varijacionog autoenkodera leži u načinu modelovanja latentnog prostora. Varijacioni autoenkoder takođe ima enkoder i dekoder, ali koristi probabilistički pristup latentnom prostoru – modeluje se kao vektor srednje vrednosti i standardne devijacije slučajnih promenljivih, čime se omogućava generativni proces. Autoenkoder ne može generisati nove podatke, dok VAE omogućava generisanje novih podataka. Dekoderski deo varijacionog autoenkodera naziva se generativni model.

A picture containing text, mammal, cat

Description automatically generated

Svaki fičer opisan je svojom srednjom vrednošću i standardnom devijacijom. Ali ovde svaki fičer znamo kako da menjamo i u kom opsegu i znamo šta znači svaki od njih. Od običnog AE dobijamo encoding vektor koji ima N dimenzija i ne razmemo ga. Sada, mi možemo sami da podesimo vrednosti fičera i oni  sad imaju smisla.

A picture containing text, screenshot, design

Description automatically generated

Osnovna namena varijacionih autoenkodera jeste generisanje novih podataka (varijacije originalnih podataka) na osnovu ulaznog skupa originalnih podataka. Korite se za generisanje novih slika, zvukova, tekstova i drugih vrsta podataka.

**Prednosti**: Simultano uče enkodiranje podataka, rekonstrukciju i generisanje podataka. Latentni skup fičera je razumljiv i može se podešavati.

**Mane**: Rezultati su obično zamućene slike i daleko su gori od rezultata GAN mreža.

1. **[DL09\_Generave\_Models]** Графички представити и описати главне елементе архитектуре генеративних противничких мрежа (енгл. *Generative Adversarial Networks – GAN*). На који начин се обучавају ове мреже?

A picture containing text, diagram, screenshot, design

Description automatically generated

Generativne protivničke mreže (GAN) sastoje se iz 2 neuronske mreže:

* **Generator** – prima random šum na ulazu i generiše izlaz.
* **Deskriminator** – na osnovu dataset-a i izlaza generatora pokušava da provali da li je podatak koji je ušao u diskriminator generisan podatak (fake) ili je podatak iz dataset-a (real). Dakle, diskriminator vrši klasifikaciju podataka u 2 klase: real i fake.

Ove 2 mreže se takmiče jedna protiv druge i uče istovremeno iz međusobnih grešaka. Zato su **protivničke**. Generator pokušava da prevari diskriminator tj. pokušava da izgeneriše izlaz koji će moći da prođe kao originalni podatak i tako prevari diskriminator. Diskriminator sa druge strane pokušava da postane pametniji i da ne dozvoli generatoru da ga prevari.

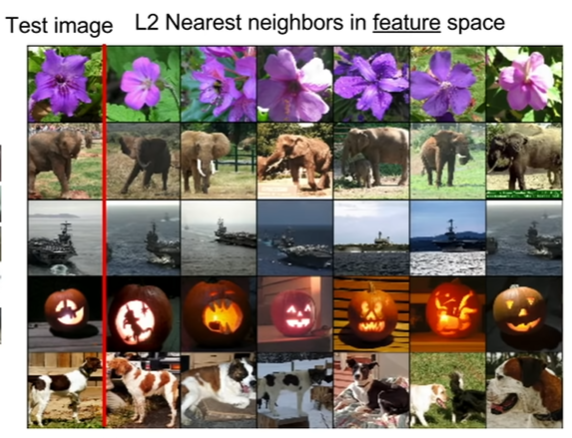
Ako diskriminator pogreši  i napravi lošu odluku, minimizuje se greška. Ako je pogrešio to znači da je generator uradio dobru stvar. Ako diskriminator pogodi, onda se generatoru maksimizuje greška i traži bolje generisanje. Kada se završi trening diskriminator više nije potreban, ostaje samo istreniran generator koji koristimo za dalje generisanje novih podataka. Diskriminator samo pomaže generatoru da uči distribuciju podataka iz dataset-a.

1. **[DL10\_Visualizing\_and\_Understanding] За које слојеве у конволуционим неуронским мрежама је могуће директно визуализовати тежине? Објаснити на који начин се то постиже?**

Први, односно улазни слој конволуционих неуронских мрежа се састоји од филтера. Ови филтери прелазе преко слика и врши се множење филтера са матрицом пиксела(конволуција). На основу филтера и како они изгледају заправо се може претпоставити шта ти филтери траже на слици преко које пролазе. Ови филтери су преузети са претренираних модела Torch-a. Уочава се да они траже орјентисане ивице, различите углове, супротне боје. Код прилично свих CNN мрежа су у првом улазном слоју ови филтери за које можемо рећи да траже ивице, углове и супротне боје.

Са друге стране, напреднији слојеви нису тако интерпретабилни и не може се лако визуелизовати као што је то случај са првим, улазним слојем. Тежине нису интуитивне као код првог слоја, а улаз у те слојеве није пиксел улазне слике као код првог слоја већ излаз из претходног конволуционог слоја, па је јако тешко директно визуелизовати тежине и разумети шта траже. Због тога, користе се друге технике како би се закључило шта CNN тражи у напреднијим слојевима.

1. **[DL10\_Visualizing\_and\_Understanding] На који начин је могуће визуализовати „простор“ вектора обележја (енгл. *feature vectors*) који се добијају на излазу конволуционог дела неуронске мреже и на основу којих се врши класификација слика?**

Узмимо за пример AlexNet мрежу која на излазу, испред класификатора, репрезентује улазну слику као 4096-димензионални вектор карактеристика на основу кога се врши класификација. Један од начина разумевања и визуализације CNN јесте разумети шта се дешава у последњем слоју. Идеја је да се кроз ту мрежу пропусти што више слика и да се прикупе вредности 4096-дим вектора карактеристика који се добије на излазу. Након тога, све забележене вредности вектора можемо веома лако визуализовати применом Nearest Neighbors метода.

Резултат овог метода приказан је на слици. Као што се види, слике су јако добро подељене, у непосредној близини јесу еквивалентни објекти иако се налазе са различитих страна слике. То је због тога што мрежа сама одређује карактеристичне тачке на основу којих касније врши поклапање. Супротно овој ситуацији, да се овај метод применио у простору пиксела, у непосредној близини би се нашле слике које поседују сличне боје и контуре иако се на њима не налазе исти објекти.

Осим овог метода, може се применити техника редукције димензионалности као што је PCA или t-SNE. Код PCA се једноставно тај вектор од 4096 димензија може редуковати на 2 и тако веома лакше визуалозовати. T-SNE је још моћнији у овом случају. Идеја код овог приступа јесте редуковати улазну слику (рецимо 28х28) на две димензије користећи t-SNE. Тако ће се формирати кластери који јасно одвајају класе.

1. **[DL10\_Visualizing\_and\_Understanding]** На који начин је могуће визуелизовати који делови слика максимално активирају поједине мапе обележја у конволуционој неуронској мрежи?

Поступак гласи овако:

Одабере се одређени слој у мрежи и одређени канал. На пример, слој 5 мреже AlexNet има 128 канала 13х13. Бирамо на пример канал 17. Пропустимо много слика кроз мрежу и пратимо вредности изабраног канала. Након тога се визуализују делови слике који одговарају максималној активацији. Сортирају се слике по активацији. У овом примеру, за слој 5, се види да се максимално активирају кругови и кружни облици на слици, као што су очи или њушка код животиња. Поређења ради, да је изабран неки ранији канал максималну активацију би изазвао неки примитивнији облик сличан линији, док би у даљим слојевима активацију изазвао неки сложенији облик као што је лице

1. **[DL10\_Visualizing\_and\_Understanding]** На који начин је могуће визуелизовати функцију једног неурона (позиције у мапи обележја) коришћењем градијентног успона (енгл.

*gradient ascent*)? Описати кораке по којима се спроводи овај поступак.

A picture containing text, font, screenshot, line

Description automatically generated

Градијентни успон генерише синтетичку слику која максимално активира неурон

1. Иницијализација слике на нуле
2. Понављати
   1. Проследити слику и израчунати вредности
   2. Backprop. ради добијања градијената према вредностима пиксела
   3. Ажурирати слику за нијансу

Постоје начини да се побољшају визуализације. Један од примера јесте представљен алгоритам регуларизације који чини да генерисане слике делују доста природније. Без регулазиратора би слика свакако максимално активирала неурон али не би личила ни на шта и не би имала смисла човеку, тако да је регуларизација додата због визуализације.

1. **[DL10\_Visualizing\_and\_Understanding]** На који начин је могуће преварити конволуциони неуронску мрежу да погрешно класификује слику (енгл. adversarial examples)? Описати кораке по којима се спроводи овај поступак.
2. Почните од произвољне слике
3. Изаберите произвољну класу
4. Модификујте слику како би се максимизовала класа
5. Понављати ове кораке док се мрежа не превари

На крају, мрежа ће се преварити али никаквих значајних модификација на слици неће бити, односно уколико се крене од слике слона и тежи на коали, слон неће постати црно бео о добити уши коале, напротив, различити биће само шумови

1. **[DL10\_Visualizing\_and\_Understanding] Шта представља и на који начин ради *DeepDream* алгоритам за креирање „халуцинација“ на основу слика?**

DeepDream алгоритам је техника која користи конволутивне неуронске мреже (CNN) за генерисање визуелних ефеката познатих као "халуцинације". Овај алгоритам је развијен од стране Google DeepMind тима.

DeepDream користи већ научене CNN моделе, као што је VGG16, који су обучени на великим скуповима слика. Идеја алгоритма је да се оптимизује улазна слика како би се максимизирале активације одређених неурона у мрежи.

Како ради DeepDream алгоритам?

* 1. *Одабир мреже*: Прво се бира дубока неуронска мрежа, на пример VGG, која је предвиђена за класификацију слика. Ова мрежа је претренирана на великом скупу слика.
  2. *Одабир слојева и активација*: Затим се бирају слојеви унутар мреже који су ближи улазу и садрже информације о облицима и текстурама на слици. Одабир слојева зависи од жељених ефеката и стила које желите да постигнете. Примера ради, уколико се одаберу почетни слојеви, биће приказани једноставнији облици, у случају каснијих слојева, биће приказане сложени објекти.
  3. *Пропагација унапред*: Улазна слика се прослеђује кроз мрежу и рачунају се активације неурона у одабраним слојевима.
  4. *Оптимизација*: Након што се израчунају активације, користи се метод оптимизације градијентима како би се израчунао градијент губитка у односу на улазну слику. Овај градијент представља информацију о томе како треба ажурирати пикселе улазне слике како би се појачала активација неурона у одабраним слојевима.
  5. *Итеративна оптимизација*: Процес обрнутог пропагирања и ажурирања пиксела улазне слике се понавља кроз више итерација. Свака итерација додаје мале измене на слику у смеру градијента губитка. То доводи до стварања изразитих облика и текстура на слици, што резултује халуцинацијама или психоделичним ефектима.
  6. *Резултат*: Када се заврши итеративна оптимизација, добија се крајња слика која садржи психоделичне облике и текстуре које су биле активиране у одабраним слојевима мреже.

1. **[DL10\_Visualizing\_and\_Understanding] На који начин функционише алгоритам за трансфер стила са једне слике на другу (енгл. *neural style transfer*)? На који начин се представља стил, а на који садржај слике?**

Алгоритам за трансфер стила, познат и као neural style transfer, је техника која омогућава комбиновање стила једне слике са садржајем друге слике. Ова техника се заснива на дубоким конволутивним неронским мрежама (CNN) и оптимизационим методама. Ево основног објашњења како функционише алгоритам за трансфер стила:

Приликом трансфера стила, постоје две слике које учествују: слика садржаја и слика стила. Слика садржаја представља слику на коју ће се пренети стил, док слика стила представља слику чији стил желимо да пренесемо.

Користе се предобучени модели CNN-а, као што су VGG или InceptionNet, који су обучени на великим скуповима слика.

Прелазак кроз мрежу. Улазна слика садржаја и слика стила се прослеђују кроз мрежу, и активације одабраних слојева се користе за израчунавање губитка (енг. loss) који мери колико се садржај и стил две слике разликују.

Губитак. Губитак укључује две компоненте: садржајни губитак и стилски губитак. Садржајни губитак оцењује колико је садржај слике садржаја запажен у генерисаној слици. Стилски губитак оцењује колико је стил слике стила пренесен на генерисану слику.

Оптимизација. Циљ је минимизовати губитак, при чему се користе оптимизациони алгоритми, као што је градијентни спуст, како би се ажурирали пиксели генерисане слике. Овај поступак се понавља више пута како би се постигао жељени резултат.

Добијање генерисане слике. Када алгоритам за трансфер стила заврши, добијамо генерисану слику која садржи садржај слике садржаја и стил слике стила.

Стил слике представља се користећи информације о текстурама, облицима и шареностима на слици. Ова информација се извлачи из активација различитих слојева CNN-а, који представљају различите нивое апстракције.

Садржај слике представља се користећи информације о глобалној структури и облицима на слици. Ова информација се такође извлачи из активација одговарајућих слојева CNN-а, који имају информације о низу сцена и објеката на слици.

Комбинујући активације слојева који представљају стил слике и слојева који представљају садржај слике, алгоритам трансфера стила омогућава генерисање нове слике која садржи стил слике стила и садржај слике садржаја.

1. **[DL11\_Reinforcement\_Learning]** Графички представити и описати основне елементе који се користе код појачаног учења (енгл. *reinforcement learning*). На примеру инверзног клатна побројати конкретне елементе по категоријама.

Agent - Komponenta koja donosi odluke – bira akcije kako bi uticao na svoje okruženje. Obično je komponenta kompleksnih sistema, radije nego ceo sistem.

Okruženje - Sve van agenta, odnosno sve nad čim agent nema potpunu kontrolu. Najčešće ostale komponente kompletskog sistema.

Akcija - Predefinisano ponašanje agenta. U svakom od vremenskih koraka, agent bira jednu ili više akcija kako bi interagovao sa okruženjem.

Stanje - Instanca skupa promenljivih kojima se predstavlja okruženje. Agent na osnovu trenutnog stanja okruženja donosi odluku koja će akcija biti izabrana.

Nagrada - Pozitivni ili negativni broj koji okruženje šalje agentu. Cilj agenta je da maksimizira ukupnu nagradu u dužem vremenskom periodu.

A diagram of a agent and environment

Description automatically generated with low confidence

1. **[DL11\_Reinforcement\_Learning]** Графички илустровати и описати архитектуру Q-мреже која је искоришћена за обучавање агената да играју класичне Atari игре. На који начин се обучаваја Q-мрежа (енгл. *Deep Q-Learning*)?

Arhitektura Q-mreže - Ulaz je stek od 84x84x84 poslednjih 4 frejma. Osnovni konvolucioni slojevi – 8x8 i 4x4. Potpuno povezani sloj 256 nakon čega sledi potpuno povezani sloj čiji broj izlaza zavisi od broja mogućih akcija.

Forward Pass: Iterativno se pokušava da vrednosna funkcija Q bude što bliže ciljanoj vrednosti yi – rešenju optimalne Belmanove jednačine

Backward Pass: Koristi se SGD za ažuriranje Q vrednosti.

A picture containing screenshot, diagram, design

Description automatically generated

1. Које су кључне карактеристике трансформер архитектуре за обраду природног језика?
   1. Атенција (Attention): Трансформер модел се заснива на механизму атенције који омогућава моделу да фокусира своју пажњу на битне делове улазних података. Атенција омогућава моделу да учи зависности и интеракције између речи у тексту, што га чини изузетно флексибилним за обраду NLP задатака.
   2. Словни уграђеници (Word Embeddings): У трансформер архитектури, речи се прво представљају као вектори у неком вишедимензионалном простору. Ови вектори, познати као словни уграђеници (word embeddings), се уче на великим скуповима текстова и могу засновати дистрибутивна својства речи, што олакшава моделу да препозна их семантичке и синтаксичке односе.
   3. Позиционални енкодинг (Positional Encoding): Пошто трансформер модел не користи рекурентне или конволуционе слојеве, потребно је додатно кодирати информацију о редоследу речи у улазном тексту. Позиционални енкодинг се додаје на сваки уграђеник и омогућава моделу да научи контекстуалне зависности на основу редоследа речи.
   4. Енкодер компонента служи како би се од улазне секвенце речи у облику великог текста генерисала нумеричка контекстуална репрезентација која успоставља везе између контекстуално сличних речи у тексту на основу контекста који свака реч у себи садржи, као и на основу позиције речи у реченици. Контекстуална репрезентација сваке речи се упоређује са осталим конт. Репрезентацијама осталих речи и на тај начин се утврђује сличност и условљеност које важи између појединих речи у реченици.
   5. Декодер компонента служи како би се на основу контекстуалне репрезентације великог текста, тј. Контекстуалних зависности између појединих речи утврдила и генерисала нова реч на основу новог улаза у мрежу при чему ће ново-креирана реч имати највећу контекстуалну условљеност са новом речју на улазу. Декодер мора да садржи контекстуалне репрезентације великог текста како би могао на квалитетан начин да генерише нови текст. Приликом генерисања новог текста, свака ново-креирана реч се рекурзивно доводи на улаз декодера како би се генерисала секвенца речи нове реченице.
2. Графички представити и описати принцип рада *stable diffusion* алгоритма за генерисање слика на основу текстуалног описа.

Difuzioni modeli se sastoje iz dva dela: difuzija unapred i difuzija unazad ○ Difuzija unapred predstavlja proces dodavanja šuma slici u određenom broju koraka ○ Difuzija unazad predstavlja proces gde se od nasumičnog šuma generiše slika ○ Nakon svakog koraka se dobija slika koja sadrži malo manje šuma i predstavlja sliku koja se prosleđuje sledećem koraku. Za generisanje slika na osnovu teksta, slikama je potrebno dodeliti opis ◎ Tokom generisanja, na kraju svakog koraka se generišu dva šuma (jedan na osnovu prosleđenog teksta, drugi standarno) ◎ Razlika ova dva šuma se oduzima od ulazne slike i dobija se izlazna slika, koja predstavlja ulaz za sledeći korak. Ovi modeli koriste U-Net arhitekturu ◎ Blokovima se prosleđuju informacije koje predstavljaju korak (positional encoding) i uneti tekst (text embedding) ◎ Latentni difuzioni modeli nadograđuju arhitekturu korišćenjem dekodera i enkodera koji smanjuju dimenzionalnost ulaza

A screenshot of a computer game

Description automatically generated with medium confidence