УНИВЕРЗИТЕТ У БЕОГРАДУ
ЕЛЕКТРОТЕХНИЧКИ ФАКУЛТЕТ

Анализа проблема апстрактног резоновања у неуронским мрежама на примеру Рејвенових матрица

Дипломски рад

Ментор
доц др. Марко Мишић

Студент
Алекса Рачић, 2019/0728

Београд, август 2023

Садржај

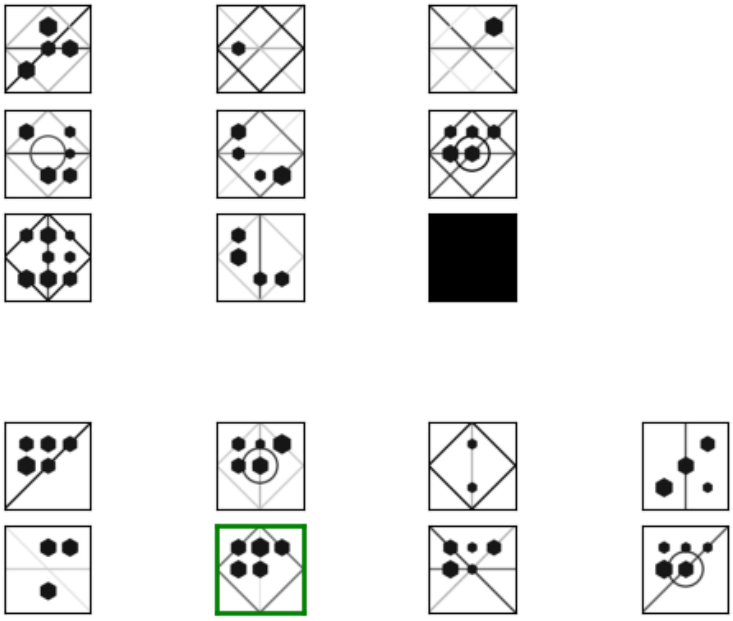
Увод

Главни циљ у домену вештачке интелигенције и машинског учења јесте стварање машине која ће имати когнитивне способности које су исте или боље од човека. Ово подразумева да машина мора бити способна да учи, закључује, решава проблеме и да се адаптира на нове ситуације, односно да поседује своју интелигенцију. Једна од фудаменталних карактеристика људске интелигенције је апстрактно закључивање - способност разазнавања шаблона, извођења закључака и решавања проблема у доменима који превазилазе просто меморисање података.

У том контексту, високо цењена Рејвенова серија тестова, коју је представио Џон Рејвен у својем раду из 1938. године [1], представља главни инструмент у процени апстрактног закључивања. Ови тестови се заснивају на употреби апстрактних визуелних стимула у облику матрица, помоћу којих се испитује способност посматрача да идентификује скривене образце и закључи о правилности или тенденцијама. Успешно решавање ових матрица захтева способност апстрактног и логичког закључивања, што га чини изазовним мерилом за анализу и процену когнитивних способности како људи, тако и алгоритама у области вештачке интелигенције.

Најчешћи облик Рејвеновоф теста се састоји од 3x3 матрице која садржи 8 познатих композиција и један непознати образац у доњем десном углу. Посматрач треба да одабере један од 8 понуђених одговора који најбоље одговара непознатом обрасцу. Постоји више различитих варијанти Рејвенових матрица, али је њихова основна идеја иста. На слици 1.1 је приказан пример једне од варијанти Рејвенових матрица која ће се користити за анализу у овом раду.

Поставља се питање: Да ли тестови интелигенције могу дати увид у интелигенцију машине? На крају крајева, машина се учи на конкретном домену задатака, док је људска интелигенција општег карактера. Али, чак и у случају људи, вежбом се могу пронаћи хеуристике које помажу у решавању таквих текстова као што је доказано у раду где су тренингом успели да повећају резултате испитаника на тестовима интелигенције [2].



Слика 1.1: Пример Рејвенове матрице.

Упркос томе, у овом раду се сматра да решавање Рејвенових матрица неуронским мрежама може да пружи увид у њихову интелигенцију и моћ апстрактног резоновања.

У сврху анализе изабране су три различите архитектуре неуронских мрежа, које су обучаване на примерима Рејвенових тестова како би добили увид у њихову способност апстрактног закључивања. Поред тога, у раду је представљен и софтверски оквир који је коришћен за имплементацију и тренирање неуронских мрежа, као и за тестирање и анализу резултата.

О проблему

Главни изазов представља сама детекција облика и линија на слици. Потребно је пројектовати архитектуру неуронске мреже која је у стању да разуме облике са слике као и њихое односе. Такође је потрено да науронска мрежа донесе одлуку који од понуђених одговора најбоље одговара непознатом обрасцу.

Изабрану архитектуру је потребно моделовати операцијама над скупом података како би рачунар могао да је извршава. Потребно је применити алгоритме учења неуронским мрежа пазећи на њихвоу ефикасност у извршавању.

На крају је потребан софтверски оквир који ће омогућити тренирање и тестирање неуронских мрежа, као и анализу резултата. Оквир треба да буде једноставан за коришћење и да омогући лаку модификацију архитектуре неуронске мреже. Такође, треба да садржи механизме за обучавање неуралне мреже, односно нашег модела. Његово тестирање на различитим скупом података и резултате визуелно прикаже како би били разумљиви човеку.

2.1 Формулација проблема

Проблем којим се овај рад бави је обиман, али се може дефинисати следећим скупом задатака:

- Пројектовање адекватне архитектуре неуронске мреже за решавање Рејвенових матрица,
- Моделовање неуронске мреже помоћу софтверског оквира,
- Тренирање неуронске мреже на скупу података,
- Тестирање неуронске мреже на скупу података,
- Анализа резултата и унапређење модела.

2.2 Машинско учење

Машинско учење представља процес учења модела из података са циљем постављања параметара тог модела на оптималне вредности. Циљ је да се модел обучи да даје најбоље резултате на скупу података који му су доступни. Ово се постиже минимизацијом функције губитка (енг. *loss function*). Функција губитка је кључни аспект у машинском учењу и она оцењује колико добро модел класификује или предвиђа резултате на основу улазних података. Што је вредност функције губитка мања, то је модел бољи у свом задатку. Процес учења модела своди се на постављање параметара модела тако да се вредност функције губитка минимизује, односно да модел тачније предвиђа жељене резултате. Ова оптимизација параметара модела је срж машинског учења и она омогућава моделима да постану способни за различите задатке и анализе на основу доступних података. У овом раду ћемо користити надгледано учење где за сваки улаз имамо јасно дефинисане излазе које очекујемо од нашег модела.

Стога да би се успешно обучила машина потребни су: модел, алгоритам за промену параметара модела (оптимизатор), функција губитка и скуп података.

2.3 Скуп података

У циљу што бољег обучавања неуронске мреже потребно је да се користи што већи скуп података. Даље, потребно је да подаци буду раздвојени у скуп за тренинг, валидацију и тестирање како би се евалуација модела радила на невиђеним подацима. У овом раду је коришћен скуп урађених Рејвенових тестова који је представљен у раду [3]. Метода којом су креирани тестови се састоји из следећих корака.

Прво се дефинише скуп типова релација, типова објеката и типова атрибута који ће се користити у генерисању тестова. Релације представљају однос у којим се облици или линије налазе у једном реду или колони. Облик се односи на то да ли се релација примењује на линије у позадини или облике који се налазе на композицијама, а тип атрибута се односи на то шта се мења у прогресији. У референтном раду су коришћени следећи скупови:

- **Тип релације (**P**)**: прогресија (енг. *progression*), ексклузивно ИЛИ (енг. *XOR*), ИЛИ (енг. *OR*), И (енг. *AND*), козистентна унија, (енг. *consistent union*),
- **Тип објекта (**O**)**: облик (енг. *shape*), линија (енг. *line*),
- **Тип атрибута (**A**)**: величина (енг. *size*), тип (енг. *type*), боја (енг. *colour*), позиција (енг. *position*), број (енг. *number*).

Затим се прави скуп тројки $S = \{[p, o, a] : p \in P, i \in O, a \in A\}$. У референтном раду

је величина скупа ограничена на мексимално четири тројке. Сваки скуп дефинише један тест пример.

После тога се насумично бирају остали атрибути (A). Током овог процеса је праћено да новододељени насумични атрибути не чине нови скуп тројки. У случају да није могуће изабрати такав атрибут сваки објекат у матрици узима исту вредност датог атрибута.

Слике које нису тачни одговори се генеришу насумично.

На основу тројки и атрибута се креирају слике Рејвенових матрица као и потенцијални одговори.

На примеру 1.1 су присутне тројке: ['или', 'облик', 'позиција'], ['или', 'линија', 'тип']. Прва тројка диригује да у једној колони позиције облика у последњем реду морају бити логичко или позиција у прва два реда. Док се друга тројка односи на то да у једном реду у последњој колони морају да се нађу сви облици линија као у прве две колоне. На основу наведеног је тачан одговор композиција под редним бројем 6.

Аутори референтног рада су представили осам различитих скупова података код којих се разликује заступљеност тројки као и вредности који атрибути могу имати и њихова заступљеност у скупу за тренирање и тестирање.

Скуп података је доступан јавности и у овом раду је коришћен готов скуп података [4]. Изабран је неутрални (енг. *neutral*) скуп података где су све тројке једнако заступљене у скупу за тренирање, валидацију и тестирање.

Скуп садржи 1.42 милиона Рејвенових тестова различите тежине подељених у три подскупа. Тренинг скуп садржи 1.2 милина, скуп за тестирање 400 хиљада, а скуп за валидацију 20 хиљада примера. Подаци су у *.npz* формату. Свака датотека садржи следеће информације:

- **Слика (енг. *image*)**: 160x160x16 матрица са вредностима од 0 до 255 која представља 16 црнобелих слика. Првих 8 слика представљају контекст Рејвенове матрице, а последњих осам потенцијалне одговоре,
- **Тројке (енг. *meta_target*)**: Скуп тројки $\{[p, o, a] : p \in P, i \in O, a \in A\}$ који су присутни у примеру,
- **Тачан одговор (енг. *target*)**: Реди број тачног одговора.

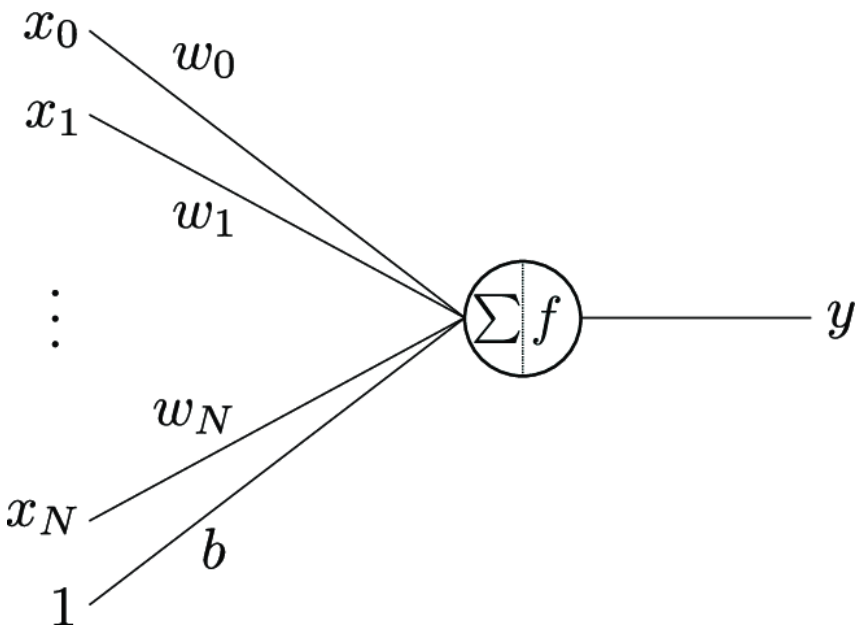
2.4 Неуронске мреже

У контексту неуронских мрежа, неурон представља основну јединицу обраде информација. Он симулира рад људског неурона у мозгу. Графички изглед једног неурона је приказана на слици 2.1. Сваки неурон прима улазне сигнале $x_0, x_1 ... x_N$, обрађује их, и издаје излазни сигнал y.

Тежине (енг. *weights*) $w_0, w_1 ... w_N$ представљају параметре који се користе за модификацију улазних података прослеђених неурону. Сваки улаз у неурон је помножен са одговарајућом тежином. Тежине утичу на значајност улазних података и одређују њихову улогу у формирању излаза неурона. Процес учења у neuronskim мрежама, познат као обука, састоји се из ажурирања и промене тежина како би мрежа најбоље моделовала жељени задатак.

Збир улаза помноженим са тежинама сигнала се прослеђује активационој функцији f. Коришћењем активационих функција, неуронске мреже су у могућности да моделирају нетривијалне односе између улазних и излазних података. Овај излазни сигнал затим служи као улаз за следећи слој неурона у мрежи.

Пристрасност (енг. *bias*) је додатни параметар који се користи за прилагођавање излаза неурона. Тежина којом пристрасност утиче на актиациону функцију је одређена тежином b.



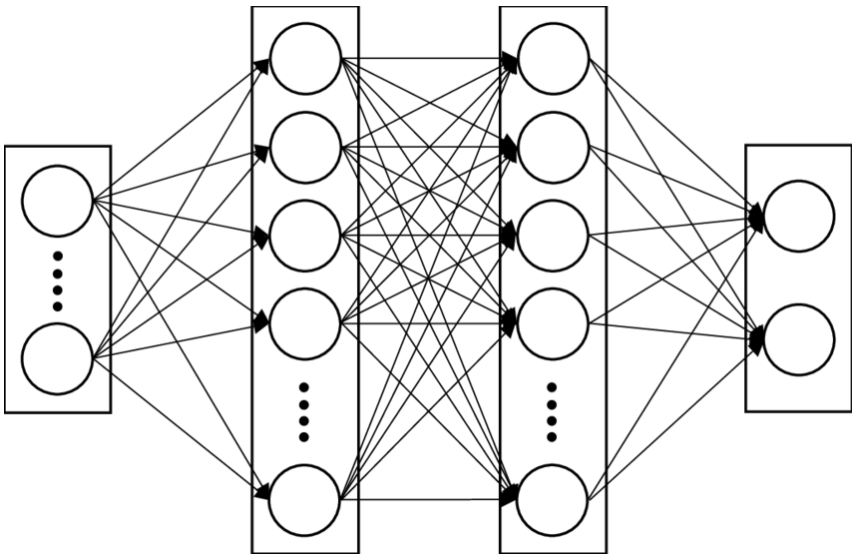
Слика 2.1: Графички приказ једног неурона. [5]

Математички израз како тежине, улази и пристрасност утичу на излазни сигнал је дат једначином 2.1.

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right) \tag{2.1}$$

2.4.1 Потпуно повезане неуронске мреже

Потпуно повезане неуронске мреже (ППНМ) (енг. Fully Connected Neural Networks) су скуп повезаних неурона описаних у 2.4. Састоји се од више слојева, први слој се назива улазни слој, а последњи слој се назива излазни слој. Сви остали слојеви се називају скривени слојеви (енг. *hidden layer*). На слици 2.2 је графички приказана неуронска мрежа која има 2 скривена слоја. Сваки слој може да има произвољан број неурона, а неуронска мрежа може да има произвољан број скривених слојева. Мењајући архитектуру, односно број неурона у једном слоју и број слојева се мења моћ мреже и комплексност задатка који мрежа може да решава.



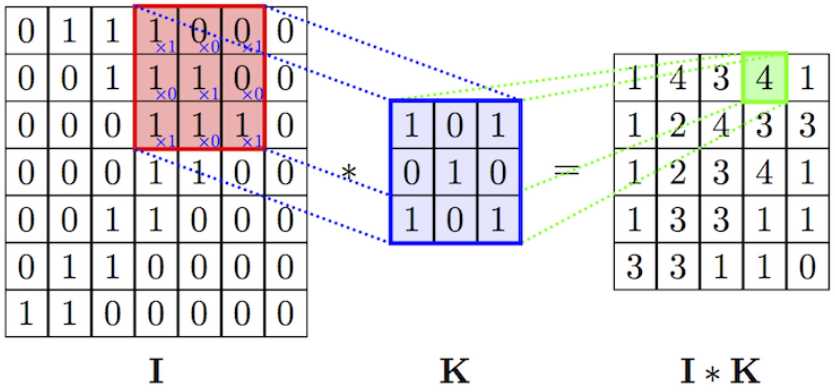
Слика 2.2: Графички приказ потпуно повезане неуралне мреже. [6]

ППНМ прима само децималне бројеве као улаз и битно је да приликом тренирања и предикције да исти вредности атрибута улазних података улазе у потпуно повезану мрежу на исти улаз. Овакав тип мреже се најбоље показао на проблемима класификације [7].

2.4.2 Конволуционе неуронске мреже

Конволуциона неуронска мрежа (КНМ) (енгл. Convolutional Neural Network - CNN) представља подврсту дубоких неуронских мрежа која је развијена за обраду и анализу

података, као што су слике и видео записи. Главна одлика ове врсте неуронске мреже јесу конволуциони слојеви који примењују велик број филтара на улазну слику. На графичком приказу 2.3 операције конволуције I представља улаз, а K филтар, који се помера кроз улаз и множи са суседним вредностима улазне слике. Током процеса тренирања филтри мењају своје вредности и понашају се као тежине. Могуће је на исти слику примењивати већи број филтера и то се означава као број канала. Поред тога конволуционе слојеве дефинише померај који говори за који број пиксела се филтар помера приликом конволуције.



Слика 2.3: Графички приказ конволуције. [8]

Математички приказ 2.2 конволуције где M и H представљају висину и ширину филтера.

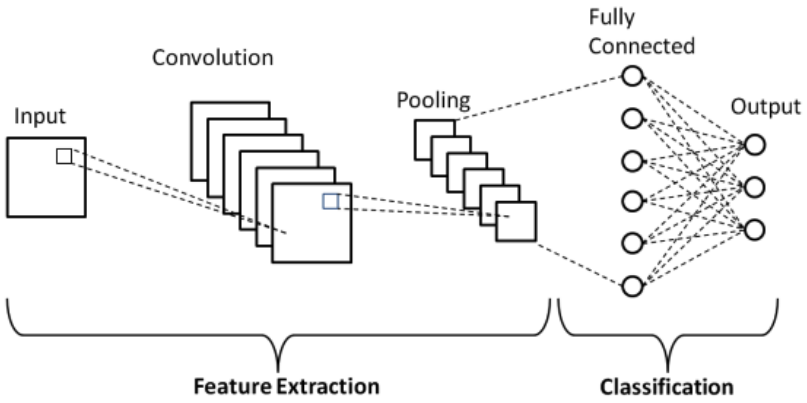
$$(I * K)[i,j] = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} I[m,n] \cdot I[i-m, j-n] \tag{2.2}$$

Конволуциони слојеви се не могу сами тренирати и обично су праћени слојевима неуралне мреже који раде класификацију. Стога се архитектура Конволуционе неуралне мреже 2.4 се састоји од низа конволуционих слојева праћеним неким другим типом неуралне мреже, обично је то потпуно повезана мрежа из 2.4.2 .

Класификациони слој служи да се декодује апстрактни запис слике из латентног вектора и квантификује излаз конволуционих слојева како би функција губитка могла оценити грешку мреже приликом тренирања.

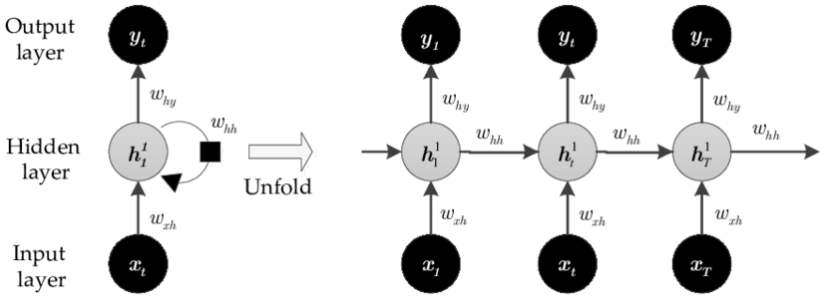
2.4.3 Рекурентне неуронске мреже

Рекурентне неуронске мреже (РНМ) (енг. Recurrent Neural Networks - RNN) представљају тип неуронских мрежа које су дизајниране за обраду и анализу серијских, узајамно зависних података, као што су текст, временске серије и генетички подаци. На слици 2.5 је приказан изглед једног неурона у рекурентној неуронској мрежи. На левој страни слике је приказан графички изглед неурона h_1^1. Излаз неурона се враћа на улаз са одре-



Слика 2.4: Графички приказ архитектуре конволуционе неуронске мреже. [8]

ђеном тежином. Ово значи да следећи излаз неурона зависи од претходног излаза. Ако развојемо неурон кроз време (слика 2.5 десно), приметимо да ће на излаз y_T утицати улази x_1, x_t, x_T.



Слика 2.5: Графички приказ једног неурона РНМ. [9]

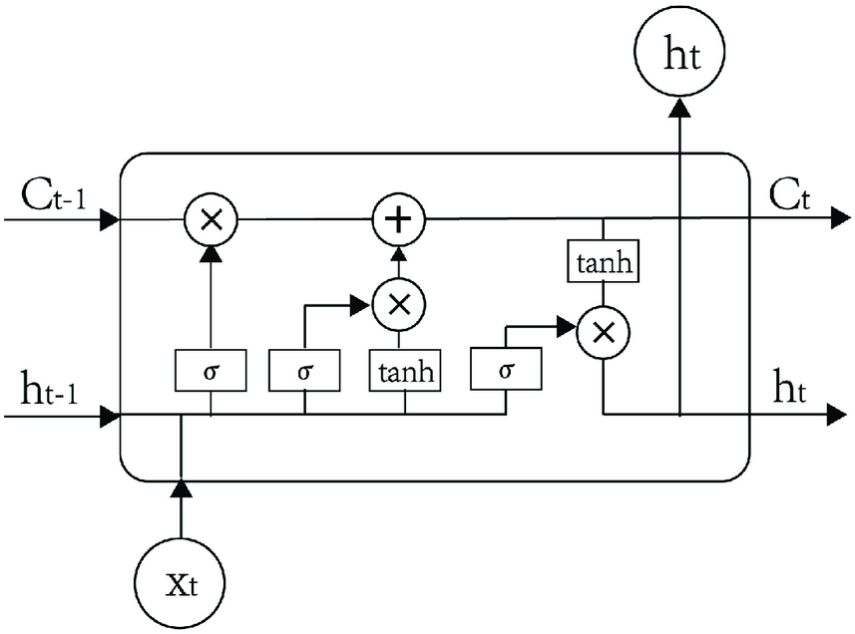
Неуронске мреже са краткорочном и дугорочном меморијом

Неуронске мреже са краткорочном и дугорочном меморијом (КДМ) (енг. Long Short Term Memory - LSTM) је специфичан тип рекурентне невралне мреже (РНМ). Она је дизајнирана да реши проблем нестанка градијента [1] (енг. vanishing gradient) и експлодирајућих градијената (енг. exploding gradient)[2] који се често јавља код стандардних РНМ-ова. Ови проблеми доводе до нестабилности мреже и прекорачења рачунарског ограничења за величине градијената.

На слици 2.6 је приказан изглед једног КДМ блока. Блок има три улаза. Улаз за сигнал дугорочне меморије (C_{t-1}), краткорочне меморије(h_{t-1}) и улаз новог сигнала (X_t).

[1]Проблем нестанка градијента је случај када, током обучавања мреже, се градијенти значајно смањи при сваком обрадном слоју.

[2]Проблем експлодирајућих градијента је случај када, током обучавања мреже, се градијенти значајно повећају при сваком обрадном слоју.



Слика 2.6: Графички приказ једног КДМ блока. [10]

Такође има два излаза који представњају нову информацију краткорочне и дугорочне меморије. Блокови се серијски повезују како би се радила предикција серија. КДМ блок садржи 3 капије, пратећи са лева на десно са слике 2.6:

- **Капија за заборављање (енг. Forget Gate)**: Ова капија одређује који део дугорочне меморије ће да се заборави.
- **Капија за ажурирање (енг. Update Gate)**: Ова капија одређује колико ће дугорочну меморију да ажурира новим информацијама.
- **Капија за излаз (енг. Output Gate)**: Ова капија одређује који део интерне меморије треба да буде излаз.

Блок садржи више активационих функција које регулишу сваку капију. Активациона функција хиперболичког тангенса ($tanh$) и сигмоидна функција (σ) ће бити детаљније описане у секцији о активационим функцијама 2.5.

2.4.4 Релационе неуронске мреже

Релациона неуронска мрежа (РНН) (енг. Relational Neural Network - RNN) су дизајниране да обраде информације које се односе на релације између објеката у систему. За неки сет улаза могу да одреде у каквој су улази релацији. То се постиже тако што два различита улаза пропустимо кроз посебне неуронске мреже, излазе спојимо, а затим опет пропустимо кроз следећу неуралну мрежу.

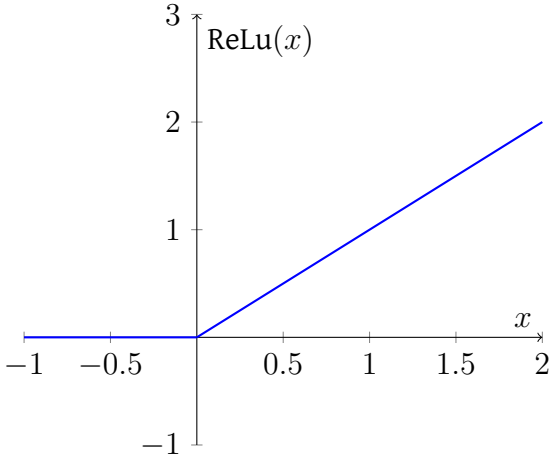
2.5 Активационе Функције

Активационе функције су математичке функције које се користе у неуронским мрежама да би се одређивала активација (излаз) неурона на основу његових улаза. Активационе функције доприносе нелинеарности у моделима, што је битно за то да мрежа може да апроксимира комплексне функције.

2.5.1 Функција активације исправљача

Функција активације исправљача (енг. ReLU - Rectified Linear Unit) активациона функција дефинисана као позитиван део свог аргумента. Све вредности које су мање од нуле постају нула, а веће се не мењају. Функција у математичком облику је дата изразом 2.3, а њен графички изглед је приказан на слици 2.7.

$$f(x) = \max(0, x) \tag{2.3}$$



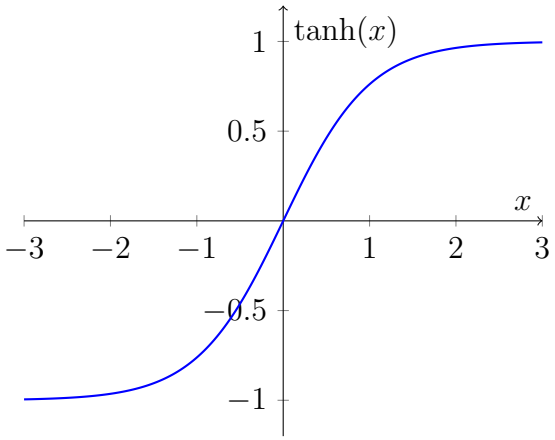
Слика 2.7: Графички приказ функције активације исправљача

Ова функција има проблем са мртвим неуронима, што су неурони који се никада не активирају за одређене улазне вредности. Ово може довести до тога да ти неурони не допринесу тренингу и остану неактивни.

2.5.2 Хиперболички тангенс

Хиперболички тангенс, обично означен као tanh, је активациона функција која све излазне вредности скупља у опсег $[-1, 1]$. Функција у математичком облику је дата изразом 2.4, а њен графички изглед је приказан на слици 2.8.

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.4}$$

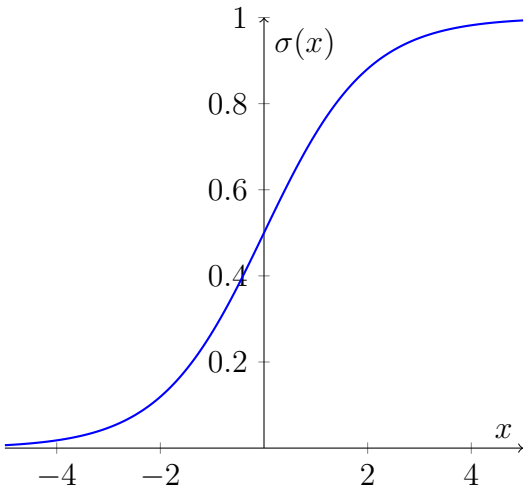


Слика 2.8: Графички приказ функције активације хиперболичког тангенса

2.5.3 Сигмоидна функција

сигмоидна функција активације јематематичка функција која има карактеристичну криву у облику латиничног слова „S” или сигмоидну криву. Означава се грчким словом сигма σ. По облику је слична хиперболичном тангенсу, али се разликује у томе што скупља вредности у опсег $[0, 1]$ и центрирана је око вредности 0.5. Функција у математичком облику је дата изразом 2.5, а њен графички изглед је приказан на слици 2.9.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.5}$$



Слика 2.9: Графички приказ функције активације сигмоидне функције

2.6 Функције грешке

Функција грешке (енг. loss function или error function) оцењује колико су предвиђени резултати (излази) различити од стварних резултата (циљне вредности) и представља

вредност коју алгоритам оптимизације покушава да минимизује током тренинга модела.

2.6.1 Унакрсна ентропија

Унакрсна ентропија (енг. Cross-Entropy Loss) је функција грешке која се често користи у задацима класификације. Ова функција се користи за мерење разлике између стварних и предвиђених вероватноћа класификације.

Нека имамо C класа у проблему класификације, где свака класа има своју стварну вероватноћу p_i и предвиђену вероватноћу q_i (где i означава индекс класе).

Унакрсна ентропија између стварних и предвиђених вероватноћа се израчунава помоћу израза 2.6.

$$H(p, q) = -\sum_{i=1}^{C} p_i \log(q_i) \tag{2.6}$$

2.7 Оптимизатори модела

Оптимизатори представљају алгоритме и методе који се користе за ажурирање параметара модела са циљем минимизовања функције грешке. Први корак представња израчунавање градијента сваког неурона. Градијент се рачуна по следећем алгоритму:

1. Улаз се пропусти кроз неуралну мрежу (енг. *Forward Pass*),
2. Израчуна се грешка модела у односу на очекивани излаз,
3. За сваки неурон се рачуна градијент функције грешке у односу на излаз тог слоја применом ланчаног правила извода (енг. *Backpropagation*).

2.7.1 Стохастички спуст (*Stochastic gradient descent - SGD*)

Алгоритам стохастичког спуста низ градијенте (енг. *Stochastic gradient descent - SGD*) је алгоритам који на основу израчунатих градијената мења вредности тежина у смеру смањења функције грешке. Понашање алгоритма је условњено параметром стопе учења (енг. *leraning rate*) који одређује за који проценат градијента желимо да променимо тежине модела. Алгоритма је приказан у псеудо коду 1.

Стохастички спуст низ градијенте може бити нестабилан током оптимизације. Градијенти се израчунавају за сваки мини-батч, ажурирања тежина су стохастичка и често

Псеудо код 1: Главна петља са SGD оптимизатором

```
for epoch ← 1 to N do
    for batch ← 1 to num_batches do
        gradient = compute_gradient(data, model)
        model_parameters = model_parameters − learning_rate · gradient
    end for
end for
```

се манифестују као шум у конвергенцији. Ово може резултовати осцилацијама или споријим конвергенцијама.

Такође је подложан заглављивању у локалним минимумима функције грешке. Ако је стопа учења превелика, алгоритам може скакати преко минимума, док ће са премалом стопом учења конвергенција бити спора или уопште не конвергирати.

2.7.2 Адам (*Adaptive Moment Estimation - Adam*)

Аутори рада [11] су представили модификацију на стохастички спуст - АДАМ (енг. *Adaptive Moment Estimation - ADAM*). Адам адаптивно регулише стопу учења за сваки параметар модела. Ово значи да се стопа учења аутоматски прилагођава за сваки параметар на основу историје градијената. Он користи механизам моментума како би убрзао и стабилизовао процес оптимизације. Моментум омогућава алгоритму да пређе платое и избегне локалне минимуме. Алгоритам, поред параметра стопе учења, уводи и следеће параметре:

- $\beta 1$ - у којој мери први моментум утиче на стопу учења,
- $\beta 2$ - У којој мери други моментум утиче на стопу учења,
- ϵ - Мала вредност која спречава дељење са нулом.

Алгоритам за Адам је приказан у псеудокоду 2 .

Псеудо код 2: Главна петља са Adam оптимизатором

for $epoch \leftarrow 1$ **to** N **do**
  for $batch \leftarrow 1$ **to** $num\_batches$ **do**
    $gradient = compute\_gradient(data, model)$
    $m \leftarrow \beta 1 \cdot m + (1 - \beta 1) \cdot gradient$
    $v \leftarrow \beta 2 \cdot v + (1 - \beta 2) \cdot (gradient \cdot gradient)$
    $m_{\text{corrected}} \leftarrow \frac{m}{1-\beta 1^t}$
    $v_{\text{corrected}} \leftarrow \frac{v}{1-\beta 2^t}$
    $model\_parameters = model\_parameters - \frac{learning\_rate \cdot m_{\text{corrected}}}{\sqrt{v_{\text{corrected}}} + \epsilon}$
  end for
end for

2.8 Технике регуларизације и убрзавања конвергенције

Технике регуларизације служе како би умањили ефекат преобучавања[3]. Са друге стране подаци који се користе за обучавање могу да имају аутлајере који негативно утичу на конвергенцију тренинга.

2.8.1 Изостављање неурона (*Dropout*)

Дропоут (енг. *Dropout*) слој се користи како би у процесу тренирања прескочили ажурирање тежина одређеног процента неурона у слоју где користимо ову технику. Он делује као регуларизација јер спречава да се било који појединачни неурон или група неурона превише ослања на одређене карактеристике у скупу за тренирање. Ово резултује у смањењу преобучавања и доприноси бољој генерализацији.

2.8.2 Нормализација гомиле (*Batch Normalization*)

Нормализација гомиле (енг. *Batch Normalization*) је техника која се користи да би се побољшала стабилност тренирања и убрзао процес конвергенције у дубоким неуронским мрежама. Током тренирања, ова техника нормализује активације сваког слоја тако да имају стандардну средњу вредност и варијансу у оквиру сваке гомиле. На овај начин се уањује утицај аутлајера и доприности стабилности учења.

[3]Појава када се модел превише прилагоди скупу података за тренирање

2.9 Додатне функције и слојеви

2.9.1 Софтмакс (*softmax*)

Софтмакс функција се често користи у неуралним мрежама за решавање задатака класификације, као и за генерисање расподеле вероватноћа излаза из мреже. Она трансформише илаз тако да све вредности излаза леже у интервалу између 0 и 1 и да се сума свих вредности излаза једнака 1 Формално, софтмакс функција је дефинисана у изразу 2.7 за вектор излаза Z :

$$\text{softmax}(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}} \tag{2.7}$$

Где:

- z је вектор излаза модела за сваку од N класа.
- e представља експоненцијалну функцију (елемент по елемент).
- i је индекс класе за коју рачунамо вероватноћу.

2.9.2 Функција највећег аргумента (*argmax*)

Фунција највећег аргумента враћа позицију највећег елемента из вектора вредности.

Коришћене технологије

За потребе овог пројекта потребни су софтверски алати који подржавају:

- Моделовање неуралних мрежа, њихово тренирање и тестирање и
- Цртање графика и визуелизација резултата и слика

Додатно је пожељно да се користи један софтверски алат који испуњава горе наведене услове ради лакшег трансфера резултата тестирања на модул за цртање графика и визуелизацију. Алат који је изабран у овом раду јесте *Python3* [12].

3.1 Python3

Python3 је интерпретерски програмски језик који подржава широки спектар библиотека. Његова одлика је да се написани код не компајлира него се интерпретира линију по линију што омогућава лакше тестирање исправности рада и могућност извршавања кода део по део. Ово га чини доста спорим у извршавању, међутим већина библиотека је оптимизована и позива се компајлиране делове кода како би се убрзало извршавање.

3.2 Torch

Torch [13] је биоблиотека намењена за креирање модела и њихово тестирање. Доступна је за *Python3* под називом *PyTorch*. Оптимизована је да ради на великом спектру хардвеских акцелератора што ће значајно да умањи време обучавања неуронских мрежа.

Додатно, садржи већ готове модуле које ће убрзати развој софтверског решења:

- Модул за потпуно повезану неуронску мрежу
- Модул за конволуције
- Модул за краткорочно-дугорочне меморијске блокове
- Имплементиране функције губитка
- Модул за ефикасно учитавање података

- Имплементирану логику ажурирања тежина у неуронској мрежи

Поред овога, потребно је имплементирати модул за релацину неуронску мрежу и спојити дате модуле како би добили жељену архитектуру неуронске мреже.

На крају, ова библиотека је уско интегрисана са библиотеком *torchvision* која пружа готове функције манипулисања са сликама као и библиотеком *tqdm* која олакшава праћење напретка тренирања неуралних мрежа.

3.3 MatPlotLib

MatPlotLib је бибиотека за *Python3* која пружа једноставан интерфејс за ртање графика и слика. У овом раду ће косрири за приказ Рејвенових тестова и плотовање прецизности мреже током тренинга неуронске мреже.

Опис решења

Решење проблема формулисаног у секцији 2 укључује израду софтвера и избор погодних модела који ће да решвају Рејвенове тестове. Такође је потрбно поставити одређени праг изнад којег се може рећи да модел поседује одређени ниво апстрактног резоновања. Једноставан модел који насумично бира одговоре има тачност од 12.5 посто. То је јер имамо 8 понуђених одговора. Сматраћемо да је успех ако мрежа пређе овај постављени праг. Поставља се питање да ли ће модел научити шаблон по којима су бирани тачни одговори или ће на основу контекста стварно изабрати најбољи одговор. Због тога је потребно да се у свакој епохи тренирања мења редослед тестова. Такође, увођењем валидационог и скупа за тест можемо тврдити да мрежа није научила шаблоне "на памет".

У насавку је описан начин избора модела који ће да решавају тестове, као и пратећи софтвер који омогућава ефикасно тренирање и тестирање модела.

4.1 Архитектуре неуронских мрежа

Као база за архитектуру неуронске мреже је узета конволуциона неуронска мрежа описана у одељку 2.5.2 јер је погодна за рад са сликама. У могућности је да енкодује одлике слике у латентни простор на основу којег се даље ради класификација резултата. Оно што представља велику препреку јесте чињеница да ми не радимо класификацију или детекцију над једном сликом, него треба да одредимо да ли неке слике прате шаблоне.

У раду [14] су аутори показали да конволуционе неуралне мреже имају моћ да препознају ротације, боје, величине и шаблоне у облицима. На основу тога ће се у овом раду користити иста архитектура конволуционе мреже за детектовање таквих фичера са улазних слика.

Прва, наивна, архитектура неуралне мреже која ће се анализирати у овом раду јесте конволуциона мрежа са потпуно повезаним слојем на излазу 4.1.1. Цео контекст Рејвеновог теста и могући одговори представњају појединачне канале улазне слике, а прет-

поставка је да ће неуралнa мрежа бити довољно комплексна да препозна релације међу улазним шаблонима.

Шаблони у Рејвеновим тестовима прате одређени образац, а решење представља наставак серије шаблона. У 2.4.3 је описано како неуралнa мрежа са КДМ блоковима има моћ праћења серије. Због тога је потребно размотрити неуралне мреже са КДМ блоковима. Међутим, у овом случају је проблем што имамо серију од 16 слика: 8 које представљају контекст, а 8 које представљају потенцијална решења, потребно је развити механизам како сигнализирати неуралној мрежи да о којем се шаблону из контекста ради, као и да ли одређена слика представља одговор. У раду [3] су то урадили додавањем одређене ознаке на латентни вектор.

На крају, овај рад ће да обухвати релационе неуралне мреже у анализу. Као и у претходним случајевима конволуциони слојеви неуралне мреже ће да енкодују слике у латентни простор, затим ће се помоћу рекурентне неуралне мреже одредити образац који се најбоље уклапа у представљени контекст. Овај приступ се показао као најбољи у раду [3].

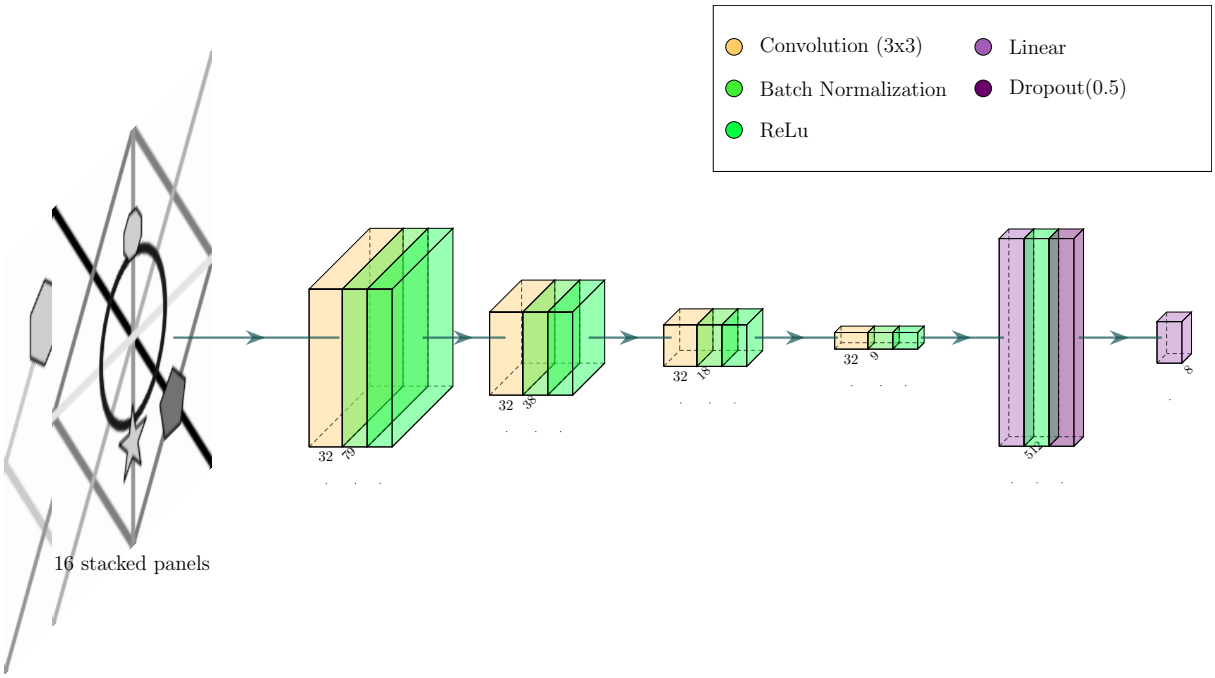
Оптимизација хипер параметара је још један битан аспект решења овог проблема. Потребно је одредити оптималан број филтара у КНМ, број и величину слојева у ППНМ. Ово се ради провером перформанси скупа архитектура са различитим хипер параметрима, међутим због хардверског и временског ограничења овај рад се ослонио на резултате других радова и хеуристику у одабиру хипер параметара.

4.1.1 Конволуциона мрежа са потпуно повезаним слојевима

Архитектура конволуционе неуралне мреже са потпуно повезиним слојевима је инспирисана радом [3]. Због хардверских ограничења број филтара је смањен са 64 на 32. Такође је умањена величина потпуно повезаног слоја са 1500 неурона на 512. Излаз ове мреже је вектор са 8 елемената од којих сваки представља вероватноћу уклапања понуђеног одговора. Табеларни приказ архитектуре је приказан на табели 4.1 , а графички приказ на слици 4.1.

Назив хипер-параметра	Вредност
КНН - број филтара	[32, 32, 32, 32]
КНН - померај филтра	2
КНН - величина филтра	3x3
ППНМ	[512, 8]
Рроценат изостављања	0.5

Табела 4.1: Табеларни приказ хипер-параметара мреже



Слика 4.1: Графички приказ конволуционе мреже са потпуно повезаним слојевима

4.1.2 Конволуциона мрежа са ДКМ блоковима

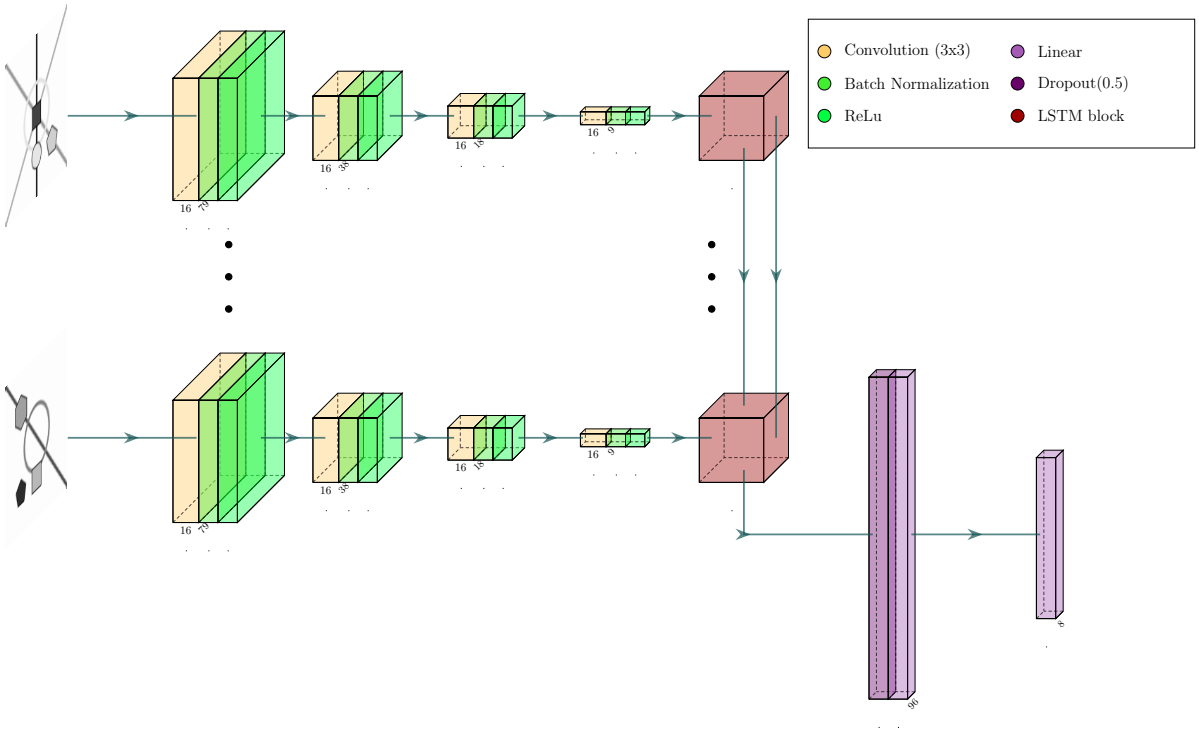
Архитектура конволуционе неуралне мреже са ДКМ блоковима је преузета из рада [3]. Број филтара је 8 из разлога што се у овом случају конволуција примењује на сваку слику посебно и самим тим је потребана мања величин латентног вектора. Остали параметри КНН су непромењени у односу на претходну архитектуру. Величина скривеног слоја, односно величина сигнала за краткорочну и сигнала за дугорочну меморију је је 96. Табеларни приказ архитектуре је приказан на табели 4.2 , а графички приказ на слици 4.2.

Назив хипер-параметра	Вредност
КНН - број филтара	[8, 8, 8, 8]
КНН - померај филтра	2
КНН - величина филтра	3x3
КДМ - скривени слој	96
Рроценат изостављања	0.5

Табела 4.2: Табеларни приказ хипер-параметара мреже

4.1.3 Конволуциона мрежа са РНН

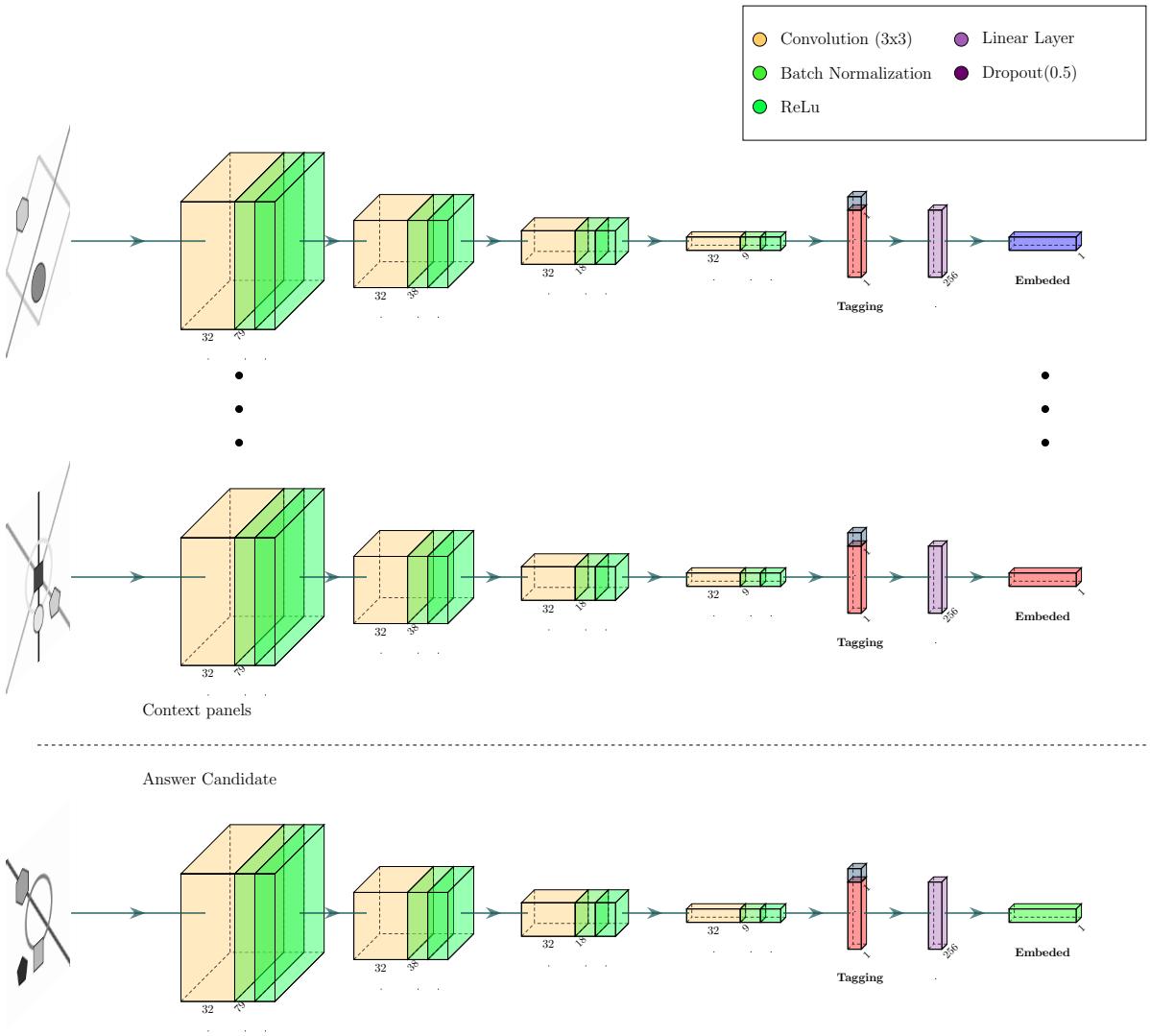
Главни допринос рада [3] јесте увођење нове архитектуре за решавање Рејвенових матрица. Они су представили нову архитектуру рекурентних неуралних мрежа за решавање Рејвенових тестова(Wild Relation Neural Network - WREN) по презимену једног од аутора (Wild). За потребе овог рада је прузета њихова архитектура, са изменама у слоју



Слика 4.2: Графички приказ конволуционе мреже КДМ блоковима

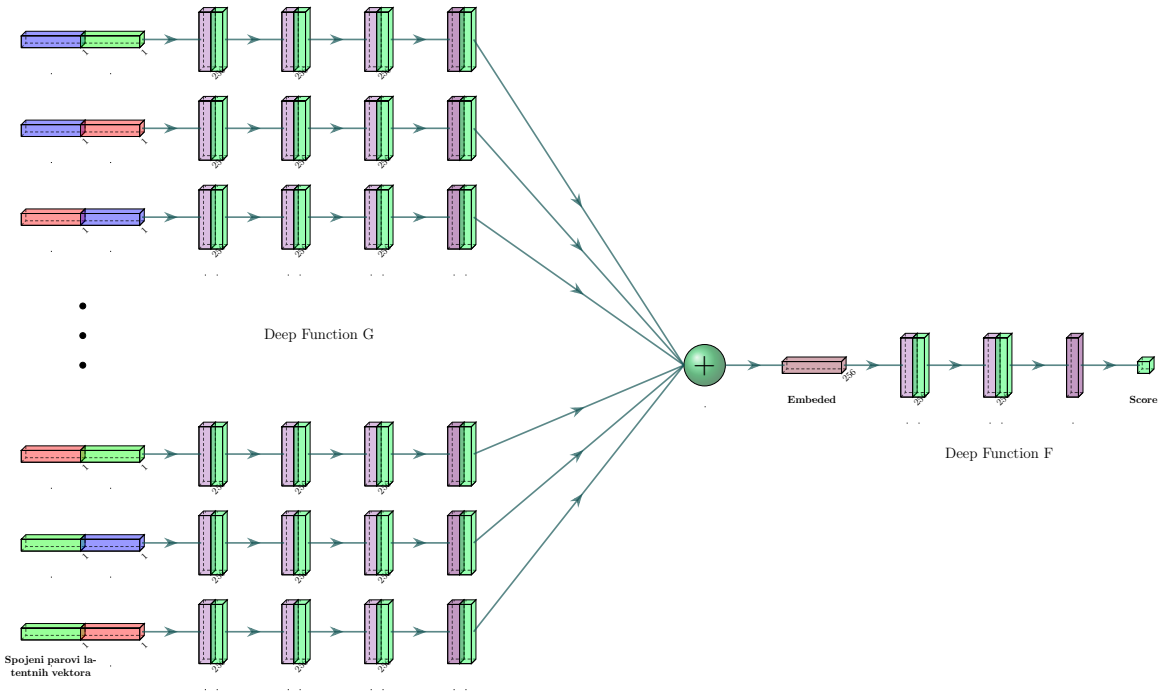
Ф. У референтом раду су користили додатне информације у виду мета матрица како би побољшали перформансе модела. Пошто се у овом раду бавимо само простом класификацијом, уместо 13 излаза, наша архитектура има само 1 који представља резултат колико се потенцијални одговор уклапа у контекст.

Први део архитектуре чини конволуциони слој који се примењује на све слике из теста појединачно 4.3, а затим се на латентни вектор додаје ознака која је слика у питању. За потребе означавања се додаје *one hot* вектор на латентни вектор сваке од слика. На крају се означени вектор пропушта кроз потпуно повезани слој са једним слојем како би векторска представа слике била одређене величине.



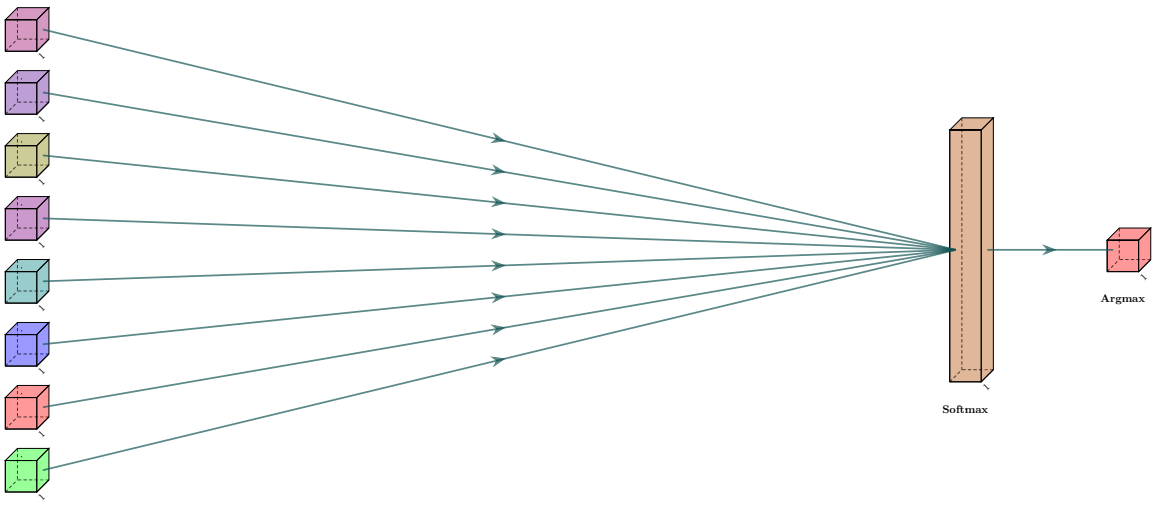
Слика 4.3: Графички приказ првог дела WREN архитектуре

Након што је припремљена векторска репрезетација, подаци се пропуштају кроз РНН - спајају се све могуће варијације вектора контексних панела са једним вектором из потенцијалних одговора. Ово се ради појединачно за сваки потенцијални одговор. Спојени вектори пропуштају кроз потпуно повезану неуралну мрежу Г 4.4. Након тога се сви резултати за један потенцијални одговор сабирају. Збир се пропушта кроз Потпуно повезани слој Ф. Резултат овог дела архитектуре је једна вредност за сваки потенцијални одговор која представља његов резултат уклапања.



Слика 4.4: Графички приказ другог дела WREN архитектуре

На крају, се резултати спајају у један вектор и примењује софмакс функција. Резултат је вектор који садржи вероватноће да је понуђени одговор прави. Након тога се одређује индекс одговора са највећом вероватноћом што представља коначан излаз мреже - argmax.



Слика 4.5: Графички приказ трећег дела WREN архитектуре

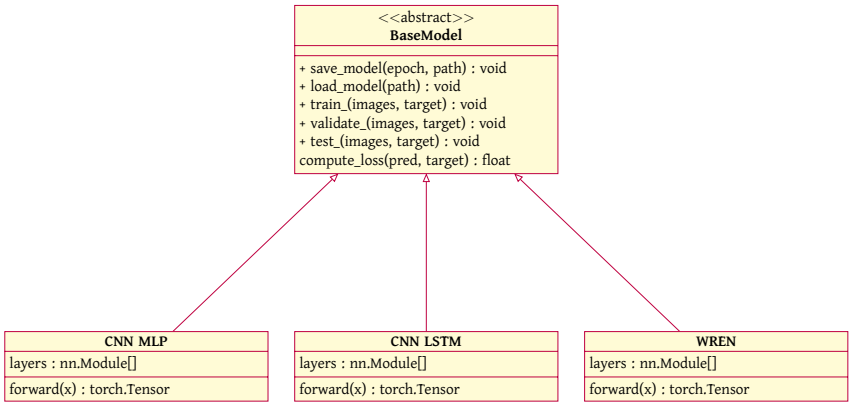
Табеларни приказ хипер-параметара архитектуре је приказан у табели 4.3

Назив хипер-параметра	Вредност
КНМ - број филтара	[32, 32, 32, 32]
КНМ - померај филтра	2
КНМ - величина филтра	3x3
Слој за енкодовање	256
ППНМ - Г	[256, 256, 256, 256]
ППНМ - Ф енкодовање	[256, 256, 1]
Рроценат изостављања	0.5

Табела 4.3: Табеларни приказ хипер-параметара мреже

4.2 Архитектура софтвера

Због чињенице да све три описане архитектуре имају исти улаз и исти излаз, могуће је издвојити логику за тренинг и рачунање губитка у апстрактну класу *BaseModel* која поседује апстрактну функцију *forward*. Остале класе које дефинишу различите архитектуре наслешују све функционалности *BaseModel* касе, графички приказ је приложен на слици 4.6.



Слика 4.6: Графички приказ функције активације исправљача

Основни (базни) модел

Основни базни модел имплементира основну логику тренирања модела. Прво се градијенти поставе на 0, затим се тренинг подаци провуку кроз мрежу након чега се израчуна функција грешке. На основу функције грешке се ажурирају тежине, а функција враћа колика је вредност функције губитка као и колико је примера тачно урађено у тренутној гомили. Алгоритам који је имплементиран за тренинг је приказан у псеудо коду 3.

Алгоритми за валидацију и тестирање не поседују кораке за ажурирање тежине, него само за предикцију и број тачно урађених тестова у итерацији.

Псеудо код 3: Алгоритам једне итерације тренинга

procedure TRAIN_($input, target$)
    $reset\_gradients()$
    $output \leftarrow forward(input)$
    $loss \leftarrow compute\_loss(output, target)$
    $loss.backward()$
    $update\_weights()$
    $pred \leftarrow argmax(output)$
    $correct \leftarrow count_matching\_elements(pred, target)$
    return $loss, correctsize(target)$
end procedure

Модел конволуционе мреже са потпуно повезаним слојевима

За имплементацију модела конволуционе мреже са потпуно повезаним слојевима су коришћене готове класе библиотеке *Torch*: *Conv2d* и *Linear*. Оне су иницијализоване са параметрима описаним у табели 4.1.

Додатно после сваке конволуције је примењена нормализација хрпе у конволуционом слоју, ППНМ је имплементирано изостављање у претпоследњем слоју.

Модел конволуционе мреже са ДКМ ћелијама

Поред модула описаних у претходној секцији, у сврху моделирања конволуционе мреже са ДКМ ћелијама је искоришћена класа *LSTM* из истог модула. Сви параметри су презети из табеле 4.2.

Модел конволуционе мреже са релационим слојевима

Модул за РНН је имплементиран помоћу интерфејса за креирање слојева *torch.nn.Module*. Комбиновање латентих вектора је урађено проширивањем димензија оргиналног тензора. Прво се проширила претпоследња димензија чиме је добијен тензор којем се у претпоследњој димензији сваки латентни вектор понавља узастопно N пута, затим је проширивање урађено по трећој димензији са краја чиме смо добили тензор којем се у последњем слоју наизменично понављају вектори N пута. Када спојимо ова два тензора резултат је тензор са свим могућим варијацијама скупа латентних вектора.

Приликом ове операције градијенти се клонирају, а приликом проласка уназад се гледа просек градијената копија при рачунању следећег градијента у ланцу.

4.3 Функција губитка, оптимизатор и учитавање података

За функцију губитка је изабрана унакрсна ентропија из одењка 2.6.1. Она је имплементирана у библиотеци под именом *cross_entropy()*.

Адам је изабран као оптимизатор модела са парамтерима приказаним у табели 4.4:

Параметар	Вредност
$learning\_rate$	0.0003
$\beta 1$	0.9
$\beta 2$	0.999
ϵ	1^{-8}

Табела 4.4: Табеларни приказ апраметара Адам оптимизатора

Резултати и дискусија

У овом поглављу прво ће бити описан хардвер на коме су резултати добијени, затим софтверски стек (енг. *software stack*) који је коришћен при обучавању и тестирању модела како би се отклониле евентуалне системске зависности. Потом, биће представљени и резултати овог рада у виду више решења проблема, свако са својим каратеристикама. Финално, биће дискутоване разике између решења и њихова могућа побољшања.

5.1 Тест платформа

За потребе овога рада је коришћен рачунарски сервер са хардверским акцелератором са ознаком *NVIDIA GeForce RTX 3060* и меморијским капацитетом од 12GB. Берзије драјвера и *CUDA toolkit*-а је приложен у табели 5.1.

Назив	Верзија
Driver	530.30.02
CUDA	12.1

Табела 5.1: Верзије драјвера и *toolkit-a*

При изради софтверског решења су искоришћене верзије програма и бибилиотека који се могу видети у табели 5.2:

Назив програма или библиотеке	Верзија
Python3	3.10.12
torch	2.0.1
torchvision	0.15.2
tqdm	4.66.1
matplotlib	3.7.2

Табела 5.2: Верзије софтера и библиотека у ком је израђен софтверски део решења

5.2 Методологија тестирања

Сви модели су обучавани на истом тренинг и валидационом скупу. Параметри оптимизатора су приказани у табели 4.4, са гомилама величине 32 (енг. *batch_size*). Због величине скупа података за тренирање валидација је вршена на сваких 10% обрађених података. У свим случајевима обучавања је имплементиран механизам раног заустављања (енг. *early_stopping*). Критеријум за рано заустављање је да се после треће епохе не повећа тачност на три узастопна валидациона сета на крају епохе.

Максимална вредност епохе је 16.

5.3 Резултати и дискусија

Закључак

Овај рад је увео проблем...
Једна занимљива идеја за будући рад би могла бити...d

Литература

[1] J. C. Raven, "Raven standard progressive matrices," *Journal of Cognition and Development*, 1938.

[2] J. Te Nijenhuis, O. F. Voskuijl, and N. B. Schijve, "Practice and coaching on iq tests: Quite a lot of g," *International Journal of Selection and Assessment*, vol. 9, no. 4, pp. 302–308, 2001.

[3] D. G. T. Barrett, F. Hill, A. Santoro, A. S. Morcos, and T. Lillicrap, "Measuring abstract reasoning in neural networks," July 2018. arXiv:1807.04225 [cs, stat].

[4] "Procedurally Generated Matrices (PGM) data," July 2023. original-date: 2018-06-07T12:40:21Z.

[5] Y. Ioannou, *Structural Priors in Deep Neural Networks*. PhD thesis, Sept. 2017.

[6] I. Anwar and N. Ul Islam, "Learned Features are Better for Ethnicity Classification," *Cybernetics and Information Technologies*, vol. 17, Sept. 2017.

[7] "Artificial neural network," Sept. 2023. Page Version ID: 1175471829.

[8] I. S. Mohamed, *Detection and Tracking of Pallets using a Laser Rangefinder and Machine Learning Techniques*. PhD thesis, Sept. 2017.

[9] M. Husein and I.-Y. Chung, "Day-Ahead Solar Irradiance Forecasting for Microgrids Using a Long Short-Term Memory Recurrent Neural Network: A Deep Learning Approach," *Energies*, vol. 12, p. 1856, May 2019.

[10] J. Qiu, B. Wang, and C. Zhou, "Forecasting stock prices with long-short term memory neural network based on attention mechanism," *PLOS ONE*, vol. 15, p. e0227222, Jan. 2020.

[11] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," Jan. 2017. arXiv:1412.6980 [cs].

[12] G. Van Rossum and F. L. Drake Jr, *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.

[13] R. Collobert, K. Kavukcuoglu, and C. Farabet, "Torch7: A matlab-like environment for machine learning," in *BigLearn, NIPS Workshop*, 2011.

[14] D. Hoshen and M. Werman, “IQ of Neural Networks,” Sept. 2017. arXiv:1710.01692 [cs].