SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Diplomski studij

RJEŠAVANJE LOGIČKE ZAGONETKE SUDOKU KORIŠTENJEM KONVOLUCIJSKIH NEURONSKIH MREŽA

Projektni zadatak iz kolegija Raspoznavanje uzoraka i strojno učenje

Filip Znaor

Osijek, 2021.

SADRŽAJ

1. UVOD

Sudoku je jedna od najpoznatijih logičkih zagonetki svih vremena. Unatoč malom broju vrlo jednostavnih pravila te veličini ploče od samo 81 polja, broj mogućih konfiguracija sudokua sastoji se od čak 22 znamenke. Kako svaki sudoku, zahvaljujući pravilima, ima striktno definirana prostorna svojstva, odnosno značajke, a konvolucijske neuronske mreže vrlo dobro pronalaze takva svojstva, postavlja se pitanje mogu li konvolucijske neuronske mreže točno i učinkovito rješavati sudoku.

1.1. Opis projektnog zadatka

Cilj ovog projektnog zadatka jest testirati uspješnost konvolucijskih neuronskih mreža u rješavanju sudokua, pritom analizirajući utjecaj parametara kao što su broj konvolucijskih slojeva i broj neurona (filtera) u svakom konvolucijskom sloju na rezultate mreže. Također, cilj je napraviti grafičko sučelje koje korisniku omogućuje jasan uvid u rješavanje sudokua od strane neuronske mreže.

2. PREGLED PODRUČJA TEME

Sudokui su računalno rješivi i bez korištenja postupaka strojnog učenja, odnosno korištenjem „običnog“ programiranja. Najjednostavnija takva metoda je tzv. *backtracking* koja naivno isprobava sve moguće kombinacije brojeva sve dok ne nađe ispravnu. Iako metoda uvijek daje točan rezultat, u određenim slučajevima može biti poprilično spora. Neke naprednije i efikasnije metode uključuju tzv. *naked twin* metodu, stohastičku pretragu i optimizaciju, Crookeov algoritam i dr.

Konvolucijske neuronske mreže nisu uobičajeno korištene za rješavanje sudokua zbog postojanja već dovoljno brzih i točnih deterministčkih algoritama. Međutim, mogućnost korištenja konvolucijskih neuronskih mreža za rješavanje sudokua razmatrana je na internetu, a primjeri mogu biti pronađeni na [1] i [2].

3. SUDOKU

Sudoku je logička zagonetka bazirana na kombinatorici. Prvi put se pojavio krajem 19. stoljeća, a današnji oblik poprima 1979. godine nakon što je objavljen u časopisu Dell Magazines. Cilj sudokua je popuniti 9x9 rešetku znamenkama od 1 do 9 tako da svaki redak, stupac i svaka od devet manjih 3x3 rešetki sadrži svaku znamenku od 1 do 9 točno jednom.

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

Slika 3.1. *Primjer riješenog sudokua*

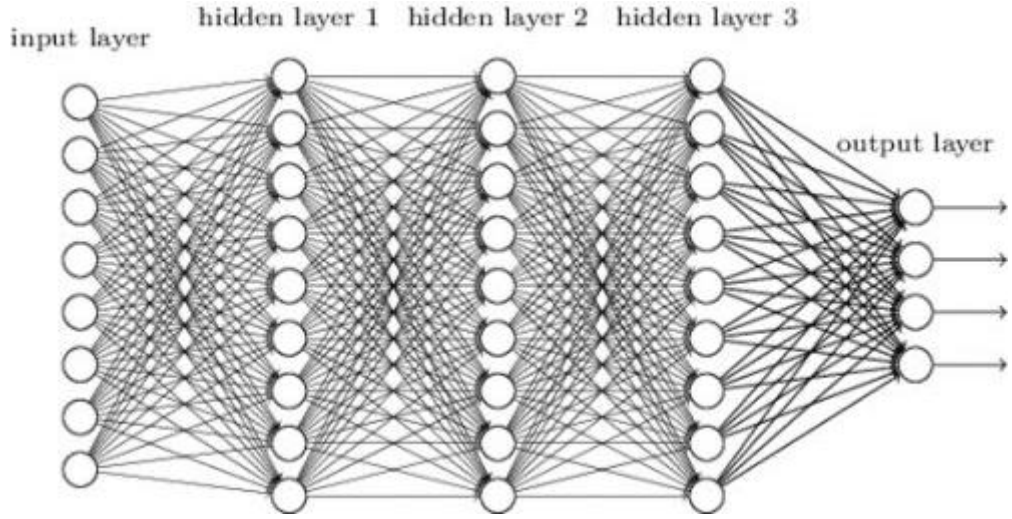
Klasični 9x9 sudoku može biti riješen na 6,670,903,752,021,072,936,960 načina. Težina sudokua ovisi o više faktora kao što su broj danih popunjenih polja (potrebno je minimalno 17 da bi sudoku imao jedinstveno rješenje) te položaj danih popunjenih polja.

Postoje brojne tehnike rješavanja sudokua. Jednostavnije tehnike oslanjaju se na proces eliminacije, odnosno pronalaženje onih polja za koja uvrštavanje samo jedne znamenke zadovoljava pravila sudokua. Naprednije metode oslanjaju se na dublju analizu svih polja i traženje pojava kao što su tzv. goli parovi (parovi ćelija u kojima su moguće samo iste dvije znamenke), skriveni parovi (isto što i goli parovi, no s dodatnim mogućim znamenkama u ćelijama), tzv. *X-Wing* (pronalaženje 4 ćelije s istom mogućom znamenkom koje tvore kvadrat) i sl.

Postoje brojne varijante sudokua koje uvode različite dimenzije rešetki, oblike manjih rešetki, dodatna ograničenja na nizove brojeva u retcima/stupcima itd., no u projektnom zadatku fokus će biti isključivo na klasični sudoku.

4. KONVOLUCIJSKE NEURONSKE MREŽE

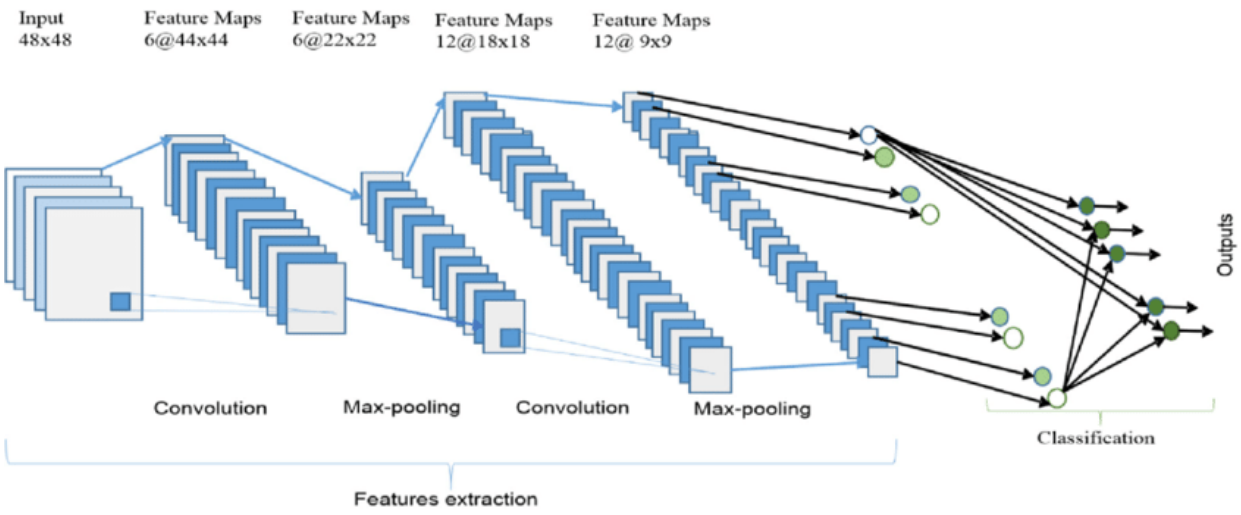
Neuronska mreža je računalni sustav čija je građa i način rada inspirirana ljudskim mozgom te čiji je zadatak rješavanje složenih problema koji nisu rješivi tradicionalnim programiranjem. Svaka neuronska mreža se sastoji od jednog ulaznog i jednog izlaznog sloja te jednog ili više skrivenih slojeva koji se nalaze između ulaznog i izlaznog sloja. Mreže s velikim brojem skrivenih slojeva nazivaju se duboke neuronske mreže, a njihova izgradnja, treniranje i validacija skupno se naziva dubokim učenjem.



Slika 3.1. *Primjer duboke neuronske mreže [3]*

Konvolucijske neuronske mreže su podskup dubokih neuronskih mreža koje se primarno uporabljuju za probleme analiziranja slika i videa, no pronalaze uporabu i u drugim područjima kao što su obrada prirodnih jezika, razvoj sustava za davanje preporuka i sl. Ono što razlikuje konvolucijske neuronske mreže od običnih (umjetnih) neuronskih mreža je uporaba tzv. konvolucijskih slojeva, odnosno matematičke operacije konvolucije umjesto isključivo matričnog množenja. Konvolucija mreži omogućuje otkrivanje raznih uzoraka, pravilnosti i pojedinosti u slikama koje mreža koristi za donošenje konačne odluke (npr. je li na slici pas ili mačka).

Kao što je vidljivo na slici 3.2., konvolucijska neuronska mreža se sastoji od više različitih vrsti slojeva. Svaki sloj ima svoju zadaću te se pravilnim raspoređivanjem slojeva i odabirom njihovih parametara mogu postići optimalni rezultati. U nastavku su opisani češće korišteni tipovi slojeva u konvolucijskim neuronskim mrežama na primjeru problema rješavanja sudokua.



Slika 3.2. *Primjer arhitekture konvolucijske neuronske mreže [4]*

Ulazni sloj (eng. *input layer*) sastoji se od određenog broja neurona gdje svaki neuron predstavlja jednu ćeliju sudokua. Kako je broj neurona u ulaznom sloju stalan, potrebno je osigurati da svi sudokui koji se predaju mreži budu istih dimenzija.

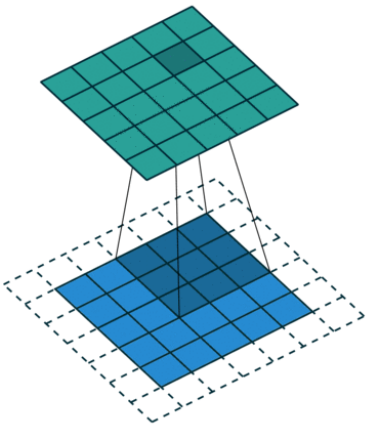
Konvolucijski sloj je najbitniji sloj konvolucijske neuronske mreže. Sastoji se od unaprijed odabranog broja neurona. Svaki neuron nad izlazom iz prethodnog sloja obavlja operaciju konvolucije.

$$s[i,j] = (I * K)[i,j] = \sum_m \sum_n I[m,n]K[i-m, j-n]$$

Slika 3.3. *Matematička formula konvolucije*

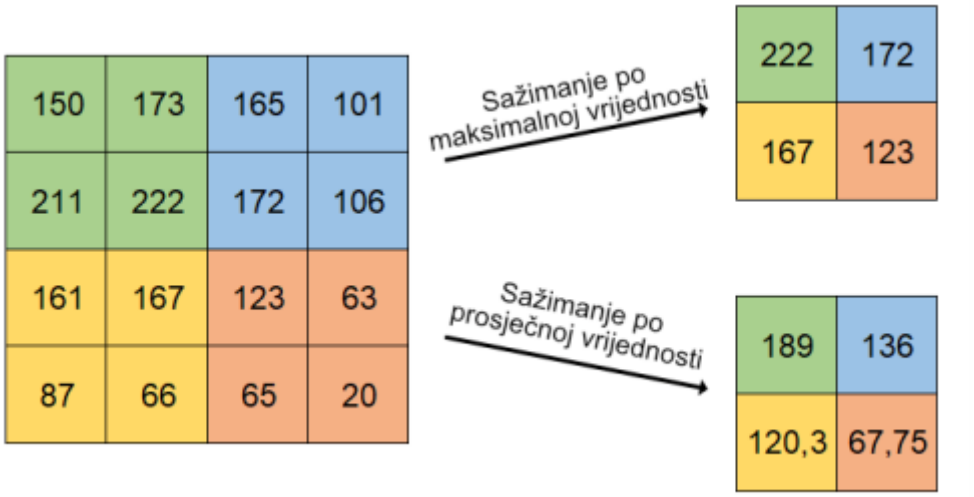
U osnovi, uzimamo filter (eng. *kernel*) čija je dimenzija manja od dimenzija sudokua te njime prolazimo kroz originalni sudoku te množimo vrijednosti elemenata na istim indeksima. Kao rezultat se dobiva tzv. *feature map,* odnosno matrica smanjene dimenzije koja na svakoj poziciji sadrži podatke o originalnom elementu na toj poziciji, ali i njegovim susjedima. Korištenje ove operacije omogućuje pronalaženje uzoraka i informacija u slikama, ali i u sudokuima. Prvi konvolucijski sloj neće biti sposoban primijetiti složene uzorke, no naslagivanjem većeg broja konvolucijskih slojeva mreža postaje sposobna detektirati složene uzorke (npr. prepoznati ljudsko lice na slici). Za konvolucijski sloj je također potrebno definirati veličinu filtera (npr. 3*3 ili 5*5) i veličinu koraka (za koliko se filter pomiče nakon svakog izračuna) što izravno utječe na dimenzije izlaza iz konvolucijskog sloja. Također, potrebno je definirati kako će se postupati s vrijednostima

koje se nalaze izvan raspona matrice nad kojom se vrši konvolucija. Najčešće se za te vrijednosti stavlja 0 ili vrijednost najbližeg ruba matrice. Osim toga, definira se je li potrebno dodati *padding* kako bi dimenzija izlaza ostala jednaka dimenziji ulaza u konvolucijski sloj. Vrijednosti elemenata matrica koje predstavljaju filtere se mijenjaju za vrijeme treniranja mreže kako bi se postigli što bolji rezultati. Naposljetku se nad izlazom iz konvolucijskog sloja upotrjebljuje aktivacijska funkcija (npr. ReLU) kako bi se u mrežu uvela nelinearnost, odnosno omogućilo pronalaženje nelinearnih uzoraka. Primjerice, ako kao ulaz imamo sliku dimenzije 32*32*3 te koristimo 6 filtara dimenzija 5*5*3, tad će izlaz iz konvolucijskog sloja imati dimenziju 28*28*6 (po jedna 28*28 matrica za svaki filter).



Slika 3.4. *Grafički prikaz konvolucije [5]*

Ako je potrebno smanjiti složenost mreže, moguće je koristiti tzv. slojeve sažimanja (eng. *pooling layers*). Sloj sažimanja smanjuje veličinu svakog *feature mapa* u sloju kako bi se smanjio broj parametara koji se prosljeđuje daljnjem dijelu mreže. Za sloj sažimanja se mora unaprijed zadati na koju će dimenziju smanjiti sve mape. Primjerice, ako mapu veličine 4*4 želimo smanjiti na 2*2, tada će se grupe od 4 elementa u originalnoj mapi morati pretvoriti u samo jedan element. Kao vrijednost tog elementa najčešće se uzima najveća vrijednost (*max pooling*) ili srednja vrijednost (*average pooling*).



Slika 3.5. *Prikaz rada sloja sažimanja [6]*

Nakon što se ulazni sudoku provede kroz veći broj konvolucijskih slojeva i slojeva sažimanja kao rezultat su dobivene mape značajki koje u sebi sadrže informacije o raznim uzorcima i informacijama u sudokuu te na temelju njih svaku ćeliju sudokua treba popuniti znamenkom između 1 i 9. Kako bi to bilo moguće, potrebno je prvo sve mape značajki pretvoriti u jednodimenzionalni vektor što se postiže tzv. *flatten layerom*. Naposljetku se dodaje tzv. *fully connected* sloj, odnosno sloj u kojem su svi izlazni neuroni spojeni sa svim ulaznim neuronima. Broj izlaznih neurona odgovara broju klasa te je svaka klasa definirana kao kombinacija značajki pronađenih u sudokuu koristeći konvoluciju. Nakon što se izračunaju vrijednosti izlaznih neurona nad njima se provodi *softmax* aktivacijska funkcija kako bi vrijednosti predstavljale vjerojatnost da ćelija sadrži određenu znamenku.

Kao i sve ostale tipove neuronskih mreža, konvolucijsku neuronsku mrežu potrebno je istrenirati na trening skupu podataka kako bi se odredila optimalna kombinacija podesivih parametara (koeficijenti filtera u konvolucijskom sloju, težine veza u *fully connected* sloju) te je zatim istreniranu mrežu potrebno testirati na novim podatcima kako bi se utvrdila njezina sposobnost da točno riješi i nove sudokue.

5. OSTVARENO PROGRAMSKO RJEŠENJE

Programsko rješenje ostvareno je u programskom jeziku Python, a za izgradnju te treniranje i testiranje konvolucijskih neuronskih mreža korištena je biblioteka Keras.

5.1. Skup podataka

Za treniranje konvolucijskih neuronskih mreža korišten je skup podataka dostupan na [7]. Skup se sastoji od milijun računalno generiranih sudokua te njihovih rješenja. Kao što je vidljivo na slici 5.1., svaki sudoku predstavljen je 81-znamenskastim brojem u kojem znamenke 0 predstavljaju neispunjena polja. Za treniranje je uzeto 900000 sudokua, dok je preostalih 100000 ostavljeno za validaciju i testiranje.

```
quizzes,solutions
004300209005009001070060043006002087190007400050083000600000105003508690042910300,864371259325849761971265843436192587198657432257483916689734125713528694542916378
040100050107003960520008000000000017000906800803050620090060543600080700250097100,346179258187523964529648371965832417472916835813754629798261543631485792254397186
600120384008459072000006005000264030070080006940003000310000050089700000502000190,695127384138459672724836915851264739273981546946573821317692458489715263562348197
497200000100400005000016098620300040300900000001072600002005870000600004530097061,497258316186439725253716498629381547375964182841572639962145873718623954534897261
005910308009403060027500100030000201000820007006007004000080000640150700890000420,465912378189473562327568149738645291954821637216397854573284916642159783891736425
100005007380900000600000480820001075040760020069002001005039004000020100000046352,194685237382974516657213489823491675541768923769352841215839764436527198978146352
009065430007000800600108020003090002501403960804000100030509007056080000070240090,289765431317924856645138729763891542521473968894652173432519687956387214178246395
```

Slika 5.1. *Korišteni skup podataka*

5.2. Građa mreže

U svrhu pronalaženja idealne mreže za rješavanje sudokua izgrađeno je 7 mreža koje se međusobno razlikuju po broju konvolucijskih slojeva i veličini konvolucijskih slojeva (broju filtera).

```
model = Sequential()

model.add(Conv2D(50, kernel_size=(3,3), activation='relu', padding='same', input_shape=(9,9,1)))
model.add(Conv2D(50, kernel_size=(3,3), activation='relu', padding='same'))
model.add(Conv2D(50, kernel_size=(3,3), activation='relu', padding='same'))

model.add(Flatten())
model.add(Dense(729))
model.add(Reshape((81, 9)))
model.add(Activation('softmax'))
```

Slika 5.2. *Primjer korištene konvolucijske neuronske mreže*

Na slici 5.2. prikazana je mreža s 3 konvolucijska sloja pri čemu svaki sadrži 50 neurona. Za veličinu filtera uzet je 3x3 jer se veličinom poklapa s manjim mrežama dimenzija 3x3 u sudokuu (isprobana je i veličina filtera 5x5, no nije davala dobre rezultate). Za aktivacijsku funkciju korištena je ReLU funkcija (*Rectified Linear Unit*) kako bi se u mrežu unijela nelinearnost. Također, zbog vrlo male dimenzionalnosti ulaznog sloja bilo je potrebno dodati *padding* u sve konvolucijske slojeve kako bi se kroz sve slojeve očuvala dimenzija 9x9. Naposljetku su dodani *flatten* i *fully connected* slojevi koji za svaku ćeliju sudokua provode klasifikaciju. Kako je u pitanju zapravo 81 odvojenih klasifikacijskih problema, izlazni sloj morao je biti pretvoren u dimenziju 81x9 kako bi se za svaku ćeliju pojedinačno primijenila *softmax* aktivacijska funkcija radi dobivanja vjerojatnosti svake znamenke za pojedinu ćeliju.

5.3. Treniranje mreže

Na slici 5.3. vidljive su postavke koje su bile korištene pri treniranju svih izgrađenih mreža.

```
adam = keras.optimizers.Adam(lr=.001)
model.compile(loss='sparse_categorical_crossentropy', optimizer=adam, metrics=['accuracy'])

callback1 = keras.callbacks.EarlyStopping(monitor='val_loss', patience=10)
callback2 = keras.callbacks.ReduceLROnPlateau(monitor="val_loss",factor=0.1,patience=5,verbose=1)

model.fit(X_train, y_train, batch_size=320, epochs=100, validation_data=(X_test,y_test), callbacks=[callback1,callback2])

model.save("model")
```

Slika 5.3. *Kôd za treniranje konvolucijske neuronske mreže*

Kao optimizacijski algoritam korišten je algoritam *Adam*, a kao kriterijska funkcija korištena je *sparse categorical crossentropy*, funkcija pogodna za slučajeve kad su izlazne veličine cijeli brojevi. Trajanje treniranja postavljeno je na 100 epoha, ali je za slučaj ranijeg postizanja lokalnog minimuma kriterijske funkcije dodana *callback* funkcija za ranije zaustavljanje procesa treniranja. Također, dodana je i *callback* funkcija za smanjenje stope učenja u slučaju da kroz duže vrijeme ne dolazi do promjene kriterijske funkcije. Kao validacijski skup podataka korišteno je 99000 sudokua iz originalnog skupa podataka.

5.4. Testiranje mreže

Testiranje mreže za rješavanje sudokua razlikuje se od standardnog postupka testiranja konvolucijske neuronske mreže. Naime, testiranje istovremenim predviđanjem svih praznih polja nije idealno za problem rješavanja sudokua jer od mreže očekuje da odmah besprijekorno odredi točnu znamenku za svako prazno polje. Ovaj postupak ne daje visoku stopu točnosti (pri testiranju na bolje performirajućim mrežama nije uspio imati stopu točnosti iznad 60%).

Bolji princip testiranja mreže bazira se na tome da mreža odjednom predviđa sadržaj isključivo jedne ćelije. To se ostvaruje predavanjem trenutnog stanja sudokua mreži te traženjem znamenke i polja koje mreža predviđa s najvišom vjerojatnošću (ne brojeći već popunjena polja). Ovim postupkom mreža uvijek popunjava samo onu ćeliju za koju je najsigurnija te pritom dobiva dodatne informacije o konfiguraciji čitave ploče koja joj pomaže u budućim izvođenjima.

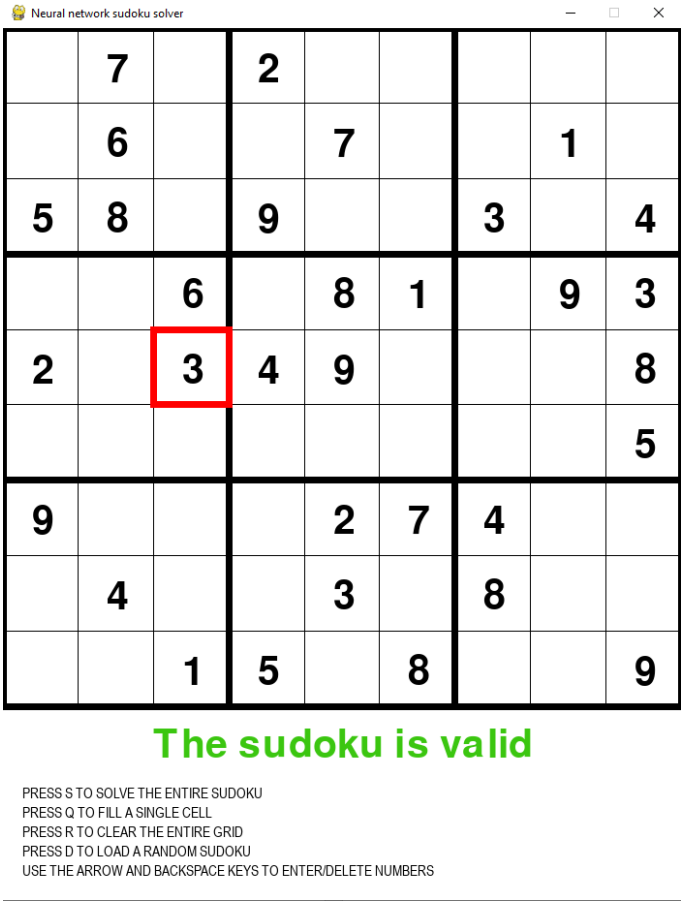
```
# repeat while there are zeros in the sudoku
while True:
    if not 0 in sudoku:
        break

    # get results predicted by the model and find the highest probability for a field that isn't already filled
    result = model.predict(sudoku)
    result = result.reshape(81,9)
    args = result.argmax(axis=1)
    sudoku = np.ravel(sudoku)
    result = result.max(axis=1)
    result = np.where(sudoku==0, result, 0)
    sudoku[np.argmax(result)] = args[np.argmax(result)]+1
    sudoku = sudoku.reshape(1,9,9,1)

# check whether the solution is correct
if(np.array_equiv(sudoku,(y_test[i]+1).reshape(9,9,1))):
    correct += 1
```

Slika 5.4. *Kôd za testiranje rada mreže*

Navedeni postupak osjetno usporava brzinu rada mreže, no rezultira znatno boljim rezultatima.

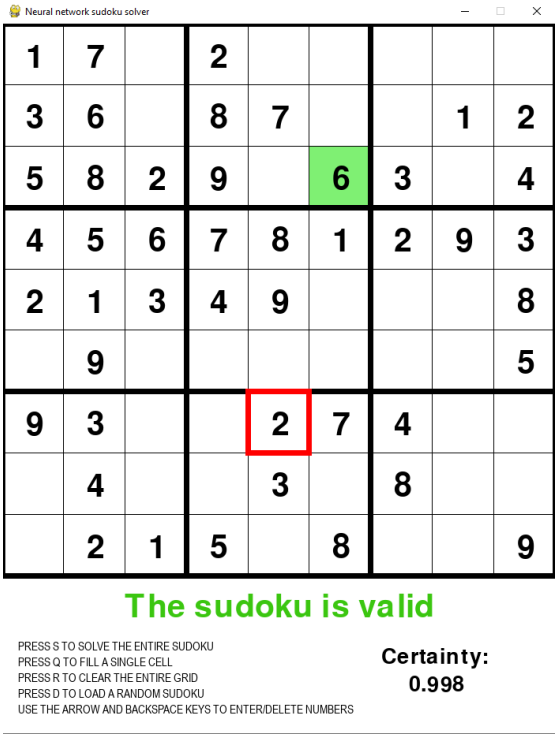
5.5. Korisnička aplikacija

Kako bi korisnik mogao lakše vizualizirati mrežin proces rješavanja sudokua te kako bi mu se olakšao samostalan unos sudokua, napravljen je program koji mu to omogućuje. Program je napravljen u Pygameu, skupu Python modula namijenjenom za razvoj jednostavnih računalnih igara.

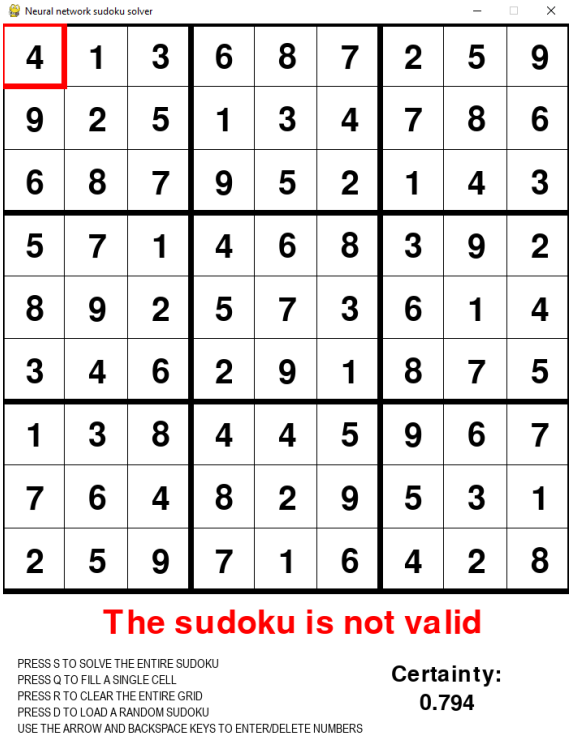


Slika 5.5. *Korisničko sučelje aplikacije*

Kao što je vidljivo na slici 5.5., korisniku je prikazana sudoku rešetka te mu je omogućeno dodavanje i brisanje znamenki sa simultanom provjerom ispravnosti sudokua pri svakoj izmjeni. Korisnik može sam unositi znamenke ili, pritiskom na tipku 'D' učitati jedan od 1000 sudokua korištenih u testnom skupu. Korisniku je također omogućeno promatrati rješavanje sudokua koristeći mrežu korak po korak ili odmah prikazati konačno rješenje. Pri svakoj novo dodanoj znamenki, to polje se oboja u zelenu boju te se ispiše vjerojatnost s kojom je mreža bila sigurna da ta znamenka treba biti upisana u to polje. Za rješavanje sudokua koristi se ista logika kao i za testiranje (popunjavanje jedno po jedno polje).



Slika 5.6. *Izgled aplikacije nakon dodavanja nove znamenke*



Slika 5.7. *Primjer netočno riješenog sudokua*

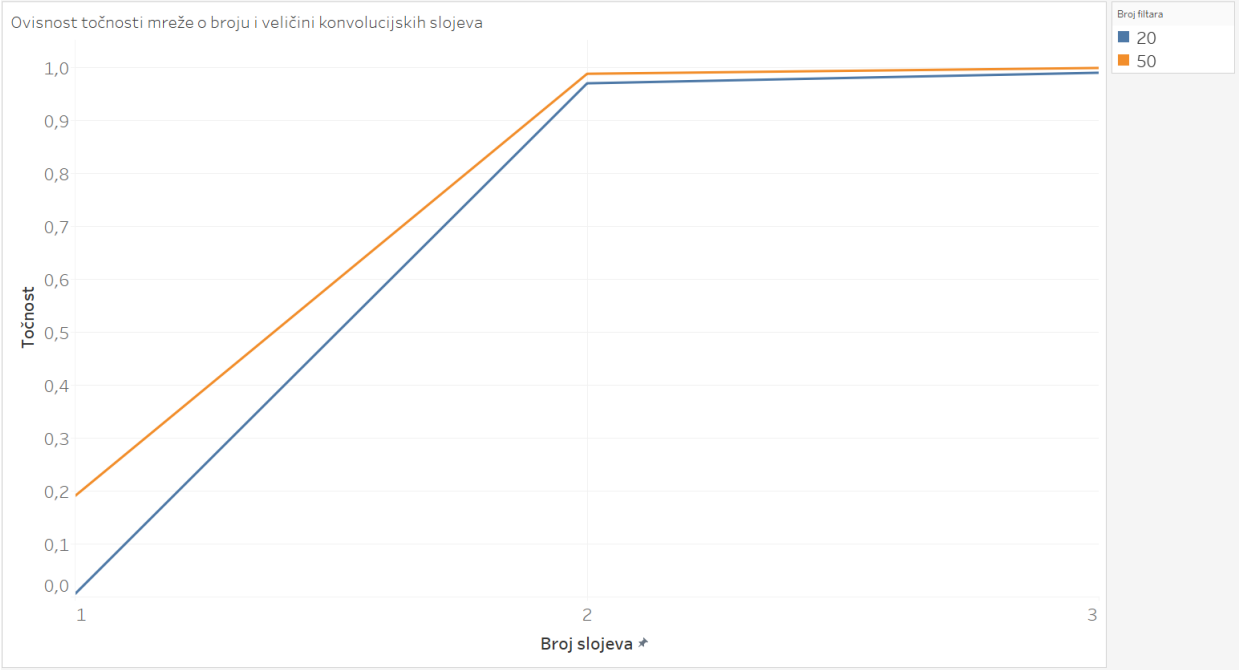
6. OSTVARENI REZULTATI

Izgrađene mreže testirane su na 1000 sudokua iz originalnog skupa podataka. Testiranje je provođeno postupkom opisanim u prethodnom poglavlju. Kako je kod sudokua bitno isključivo konačno rješenje, kao mjera uspješnosti uzeta je konačna točnost sudokua, a ne broj ispravno popunjenih polja. U tablici 6.1. su vidljive korištene mreže te njihova točnost u rješavanju sudokua. Parametri mreža koji nisu navedeni u tablici zajednički su svima te su vidljivi na slikama 5.2. i 5.3. (3x3 veličina filtra, ReLU aktivacijska funkcija, *padding* i dr.)

Tablica 6.1. *Postignuti rezultati na testnom skupu*

Broj konvolucijskih slojeva	Broj filtara u konvolucijskom sloju	Broj parametara	Točnost
1	20	1181989	0.6%
1	50	2853679	19.1%
2	20	1185529	97.0%
2	50	2976229	98.8%
3	20	1189149	99.0%
3	50	2998779	99.9%
3	100	6086829	99.9%

Iz rezultata je jasno vidljivo da se točnost povećava dodavanjem konvolucijskih slojeva te povećavanjem broja filtara u konvolucijskim slojevima. Također je primjetljivo kako mreži trebaju minimalno 2 konvolucijska sloja kako bi davala zadovoljavajuće rezultate, dok dodavanje dodatnih konvolucijskih slojeva dodatno poboljšava točnost. U ovom slučaju nije bilo potrebno imati više od 3 konvolucijskih slojeva jer je dosegnuta točnost od gotovo 100%. Kao model korišten u korisničkoj aplikaciji uzeta je mreža s 3 konvolucijska sloja i 50 filtara po sloju jer daje jednako dobre rezultate kao i mreža sa 100 filtara uz korištenje dvostruko manje hiperparametara.



Slika 6.1. *Ovisnost točnosti mreže o broju i veličini konvolucijskih slojeva*

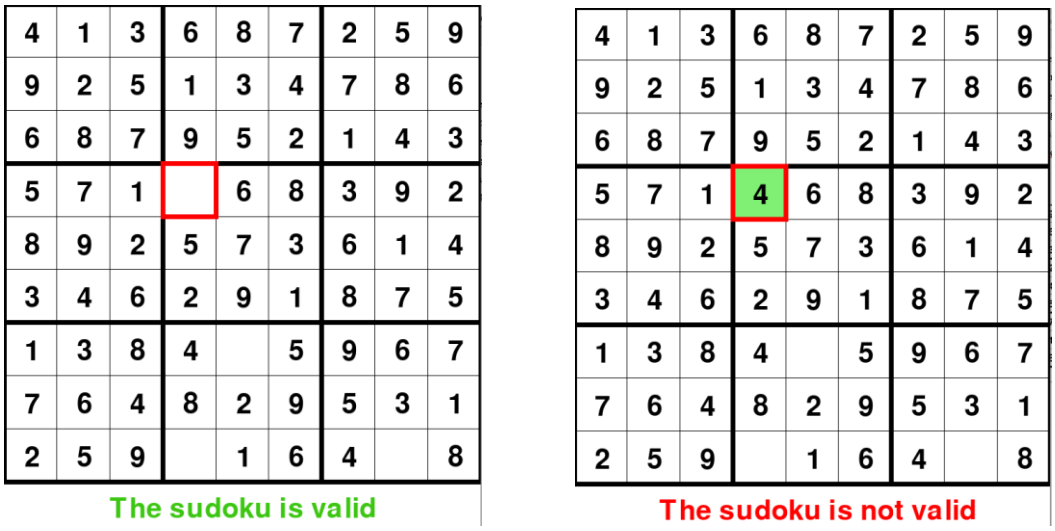
Korištenjem korisničke aplikacije moguće je analizirati kako mreža rješava sudoku. Kako mreža uvijek popunjava isključivo ono polje za koje je najsigurnije u točno rješenje, većina poteza je u skladu s onima koje bi i čovjek odigrao, tj. potezi u kojima nije odmah vidljivo da je ispunjeno polje uistinu točno vrlo su rijetki. Na slici 6.2. dan je primjer jednog poteza. Mreža promatrajući sva polja zaključuje kako je jedini mogući položaj znamenke devet u gornjoj lijevoj manjoj rešetki u prvom polju drugog stupca na temelju ostalih devetki u mreži te popunjava to polje. Čovjek bi pri rješavanju lako došao do istoga zaključka.

		7	4			9	6	
		1	6	3	8	2	5	
			9		2	4		
2	5			1	9			
4						3		8
6			7		3			
	9	2	1					5
5	8	6		2			9	
					6	8	7	

		7	4			9	6	
9		1	6	3	8	2	5	
			9		2	4		
2	5			1	9			
4						3		8
6			7		3			
	9	2	1					5
5	8	6		2			9	
					6	8	7	

Slika 6.2. *Primjer odigravanja jednog poteza*

Također je zanimljivo i analiziranje netočno riješenih sudokua. Naime, pri svakom testiranom sudokuu koji je završio netočnim rješenjem, mreža je napravila prvi ilegalan potez (ponavljanje znamenke u stupcu/retku/3x3 rešetki) tek u zadnjih 5 poteza. Ovo pokazuje kako mreža neće pri početku rješavanja napraviti vrlo očitu grešku poput npr. upisivanja dvaju jedinica u isti stupac, već će, vođena nedostatkom informacija o čitavoj rešetki u određeno polje unijeti krivu znamenku te, nesvjesna pogreške, nastojati poštovati pravila sudokua što je duže moguće. Primjer toga vidljiv je na slici 6.3. Iako je mreža već ranije napravila pogrešku unosom krive znamenke u ćeliju, nastavila je popunjavati polja poštujući pravila sudokua sve dok nije došla do točke gdje nije bilo moguće izbjeći ilegalan potez. Ovo dodatno potvrđuje kako je izuzetno bitno popunjavati sudoku polje po polje kako bi mreža dobivala što više dodatnih informacija o čitavoj rešetki što je brže moguće, time minimizirajući vjerojatnost pogrešnog poteza.



Slika 6.3. *Primjer netočnog poteza*

Unatoč izuzetno dobrim performansama mreže na testnom skupu, primijećeni su određeni nedostatci. Naime, svi sudokui u korištenom skupu podataka sadrže između 45 i 50 praznih polja. Treniranje mreže na takvom skupu omogućuje joj da uspješno rješava sudokue sa sličnim brojem praznih polja (i one sa manje praznih polja). Međutim, mreža nije sposobna riješiti većinu sudokua s više od 50 praznih polja. Ovo je testirano na 10 nasumično generiranih sudokua na [8] srednje i visoke razine složenosti, od kojih je mreža uspješno riješila tek 2. Nedostatak prikladnih skupova podataka na Internetu onemogućio je daljnje treniranje mreže za uspješno rješavanje složenijih sudokua. Isprobana je metoda u kojoj se mreža trenira na sudokuima u kojima je nasumično obrisano između 50 i 60 polja, no metoda nije dala dobre rezultate jer nasumično brisanje ne garantira da će se dobiti sudoku s jedinstvenim rješenjem što uvelike ograničava uspješnost

treniranja mreže. Potencijalno rješenje bilo bi razvoj i korištenje generatora sudokua za generiranje vlastitog skupa podataka koji sadrži i sudokue s više od 50 praznih polja.

7. ZAKLJUČAK

Sposobnost konvolucijskih neuronskih mreža da otkrivaju prostornu povezanost varijabli pokazala se učinkovitom u rješavanju sudokua. Uz dovoljno velik i raznolik skup podataka za treniranje, mreža je sposobna naučiti pravila rješavanja sudokua te ih slijediti kako bi došla do točnog rješenja. Kako je operacija konvolucije bitna za dobivanje podataka o okolini svake ćelije u sudokuu te za daljnje izvlačenje značajki, uspješnost mreže u rješavanju sudokua znatno se poboljšava dodavanjem konvolucijskih slojeva kao i povećavanjem broja neurona u svakom konvolucijskom sloju.

U procesu rješavanja sudokua bilo je bitno ograničiti mrežu da popunjava jedno po jedno polje umjesto sva odjednom kako bi se postigla bolja točnost. Ova metoda korištena je za testiranje rada mreža te za rad mreže u korisničkoj aplikaciji.

Napravljene mreže daju odlične rezultate na testnom skupu podataka. Međutim, ne performiraju najbolje na složenijim sudokuima (sudokui s više praznih polja) zbog nedostatka takvih sudokua u korištenom skupu podataka. Daljnja poboljšanja rada mreže moguća su nadogradnjom skupa podataka, odnosno dodavanjem složenijih sudokua u skup.

Iako je u projektnom zadatku pokazano da konvolucijske neuronske mreže uistinu mogu rješavati sudoku s visokom razinom točnosti, postojanje brzih i učinkovitih deterministčkih algoritama za rješavanje sudokua umanjuje potrebu za razvojem neuronskih mreža za isti zadatak.

LITERATURA

[1] C. Akin-David, R. Mantey, Solving Sudoku with Neural Networks, dostupno na: https://cs230.stanford.edu/files_winter_2018/projects/6939771.pdf [13.6.2021.]

[2] S. Verma, Solving Sudoku with Convolution Neural Network, dostupno na: https://towardsdatascience.com/solving-sudoku-with-convolution-neural-network-keras-655ba4be3b11 [13.6.2021.]

[3] A. Kyrykovych, Deep Neural Networks, dostupno na: https://www.kdnuggets.com/2020/02/deep-neural-networks.html [13.6.2021.]

[4] https://www.researchgate.net/figure/The-overall-architecture-of-the-Convolutional-Neural-Network-CNN-includes-an-input_fig4_331540139 [13.6.2021.]

[5] S. Saha, A Comprehensive Guide to Convolutional Neural Networks – the ELI5 way, dostupno na: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 [13.6.2021.]

[6] R. Grbić, Uvod u duboko učenje. Konvolucijske neuronske mreže., kolegij Raspoznavanje uzoraka i strojno učenje, FERIT

[7] https://www.kaggle.com/bryanpark/sudoku [14.6.2021.]

[8] https://www.sudoku-solutions.com/ [14.6.2021.]