Chess Position Scanner

Projekat iz predmeta Duboko Učenje

Vuk Grujić
Br indeksa: 1693

SADRŽAJ

1. Opis projekta

U okviru ovog projekta bavimo se problemom prepoznavanja digitalnih slika pozicija u šahu i generisanjem pozicije u FEN formatu, koji je objavljen u obliku Kaggle problema. Projekat je odrađen u obliku istraživačkog projekta, i samim tim obuhvata isprobavanje više različitih rešenja za ovaj problem kako bi se našlo rešenje sa najboljom preciznošću.

1.1 Opis dataseta

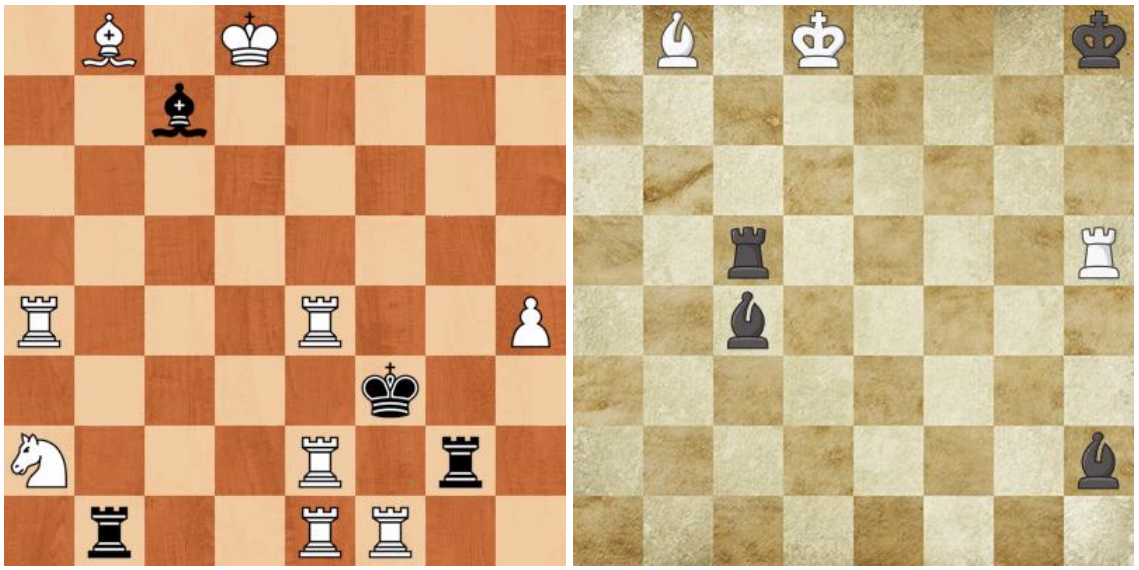
Dataset se sastoji od 100.000 slika nasumično generisanih šahovskih pozicija sa po 5-15 figura. Od ovih figura, 2 su uvek kraljevi, po jedan za svakog igrača, dok ostale figure figure prate distribuciju:

- 30% za Pijun (Pawn)
- 20% za Lovca (Bishop)
- 20% za Konja (Knight)
- 20% za Topa (Rook)
- 10% za Kraljicu (Queen)

Slike su generisane korišćenjem 28 stilova za šahovsku tablu kao i 32 stilova za figure, što nam ukupno daje 896 kombinacija.

Dataset je zvanično koristi 80/20 podelu za train i test, tako da je i tokom izrade projekta praćena ova podela. Međutim train set je dalje podeljen 80/20 podelom na train i validation setove.

Same slike su veličine 400x400 pixela, u JPEG formatu, i sadrže odgovarajuću FEN notaciju u imenu slike. Kako je korišćen JPEG format koji predstavlja vid lossy kompresije, moramo nagovestiti da će slike sadržati određenu količinu buke, tj *noise*.



Slika: Primeri slika koji se mogu naći u datasetu.

1.2 FEN notacija

FEN (*Forsyth–Edwards Notation*) je standardna notacija koja opisuje određenu poziciju u šahu. Primera radi, početna pozicija u šahu se može ispisati u FEN notaciji kao:

```
rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq - 0 1
```

Gde svaki segment, podeljen sa '/' predstavlja jedan red na tabli, od osmog reda pa do prvog. Unutar ovih segmenta koriste se karakteri:

- P za Pauna (Pawn)
- B za Lovca (Bishop)
- N za Konja (kNight)
- R za Topa (Rook)
- K za Kralja (King)
- Q za Kraljicu (Queen)

Pritom se kapitalizacija koristi kako bi se označila boja figure, gde ne kapitalizovani karakteri opisuju crne figure, dok kapitalizovani karakteri opisuju bele figure.
Takođe, ukoliko imamo n praznih polja za redom, to označavamo brojkom. (Kao što vidimo u primeru, gde su 4 reda u sredini prazni).

Nakon toga sledi karakter koji označava koji igrač trenutno igra. w (White) za belog igrača, b (Black) za crnog igrača.

Dalje imamo segment sa karakterima K i Q, koji označavaju da li odgovarajući igrač jošuvek može da izvrši rokadu (Castling) sa kraljeve ili kraljičine strane. Duplirano je za oba igrača prateći već spomenuta pravila kapitalizacije.

Zatim sledi segment koji beleži poziciju, u standardnoj šahovskoj notaciji (eg. e3), na kojoj je moguće vršiti En passant. Ukoliko nije moguće onda se koristi znak '-'.

Najzad imamo dva polja, jedan za broj polu poteza (koji računa potez svakog igrača zasebno) od poslednjeg uzimanja ili napredovanja pešaka koji se koristi za pravilo 50 poteza. Kao i broj punih poteza (koji počinje od 1 i inkrementira se posle svakog poteza crnog igrača).

Na osnovu postavke problema na Kaggle stranici, tokom ovog projekta bavimo se samo prvim segmentom ove notacije koji opisuje trenutnu poziciju. Međutim kako se radi o notaciji koja je pre svega napravljena za kompaktnost prenošenja informacija, tokom analize koristi se proširena verzija ove notacije. Ova proširena verzija notacije izbacuje brojke i sva prazna polja opisuje praznim (whitespace) karakterima (' '). Takođe se polje predstavlja matricom karaktera umesto jednom dugom linijom. Primera radi, početna pozicija bi se onda predstavila kao:

```
'rnbqkbnr'
'pppppppp'
'        '
'        '
'        '
'        '
'PPPPPPPP'
'RNBQKBNR'
```

1.3 Korišćene tehnologije

Za izradu ovog projekta korišćen je *Python* programski jezik zajedno sa *Jupyter* okruženjem. Ceo projekat organizovan je u skup *Jupyter Notebook*-a na osnovu oblasti.

Što se tiče biblioteka, korišćene su:

- *Tensorflow* kao primarna biblioteka za implementaciju veštačkih neuronskih mreža kao i njihovog treniranja. Specifično verzija 2.15 kako su postojali problemi sa grafičkim drajverima na najnovijoj verziji.
- *Pillow* za operacije predprocesiranja slika. Korišćen je Pillow umesto OpenCV biblioteke koja je navedena u dokumentu za prijavu projekta iz dva razloga. Pre svega jer je Pillow već instaliran, a i takođe jer na kraju nisu zahtevane naprednije transformacije koje OpenCV obezbeđuje.
- *Tensorflow model optimization* pod-biblioteka kako bi se izvršilo orezivanje dobijenih veštačkih neuronskih mreža.
- *Numpy* za implementaciju pojedinih vektorskih operacija nad slikama koje se izvršavaju na CPU umesto na GPU.

1.4 Arhitektura projekta

Kako se radi o istraživačkom projektu, gde je fokus stavljen na isprobavanje više različitih rešenja kako bi se našao najbolji, arhitektura koda je minimalna. Jedini deljeni kod jeste kod za učitavanje skupova podataka, kao i kod za evaluaciju mreže nad test skupom.

Učitavanje skupova podataka vršeno je preko load_dataset funkcije, koja u pozadini koristi image_dataset_from_directory Tensorflow funkciju sa prepopručenim vrednostima uzetim iz dokumentacije Tensorflow biblioteke.

Kod za evaluaciju mreža se primarno koristi preko continuous_eval i continuous_eval_color funkcija. Ove funkcije ivršavaju testove i prikazuju greške čim se pojave i, ukoliko je uključen ret_on_fail flag, odmah prekidaju dalju evaluaciju. Ovo je urađeno kako bi se sačuvalo vreme na evaluaciji s'obzirom da vreme pune evaluacije traje 15-20min.

Koraci koji se vrše tokom evaluacije su:

- Učitava se slika cele ploče iz test skupa.
- Ploča se deli na 8x8 pod-slika polja.
- Sva polja se odjednom šalju klasifikatoru.
- Rezultati klasifikatora se konvertuju u dekompresovan FEN format spomenut u poglavlju 1.2.
- Ovaj format se zatim konvertuje u normalan FEN format, i upoređuje se sa očekivanim rezultatom.

Ovi koraci su svi, van predikcije mreže, implementirani na CPU, korišćenjem Pillow i Numpy biblioteka. Testirana je i implementacija preko Tensorflow custom slojeva koje bi trebala da vrši neke operacije na GPU, međutim u praksi je ova implementacija bila 20% sporije u odnosu na predhodnu, i samim tim je zanemarena.

Sam projekat podeljen je u 3 Jupyter Notebook-a:

- 01 - Preprocessing: Gde je izvršeno preprocesiranje podataka kao i kreiranje training skupa koji se koristi nadalje.
- 02 - Networks: Koja sadrži sve pristupe problemu, osim orezivanja.
- 03 - Pruning: Koja sadrži orezivanje, kako je Tensorflow Model Optimization biblioteka imala čudne interakcije sa Tensorflow 2.15 verzijom.

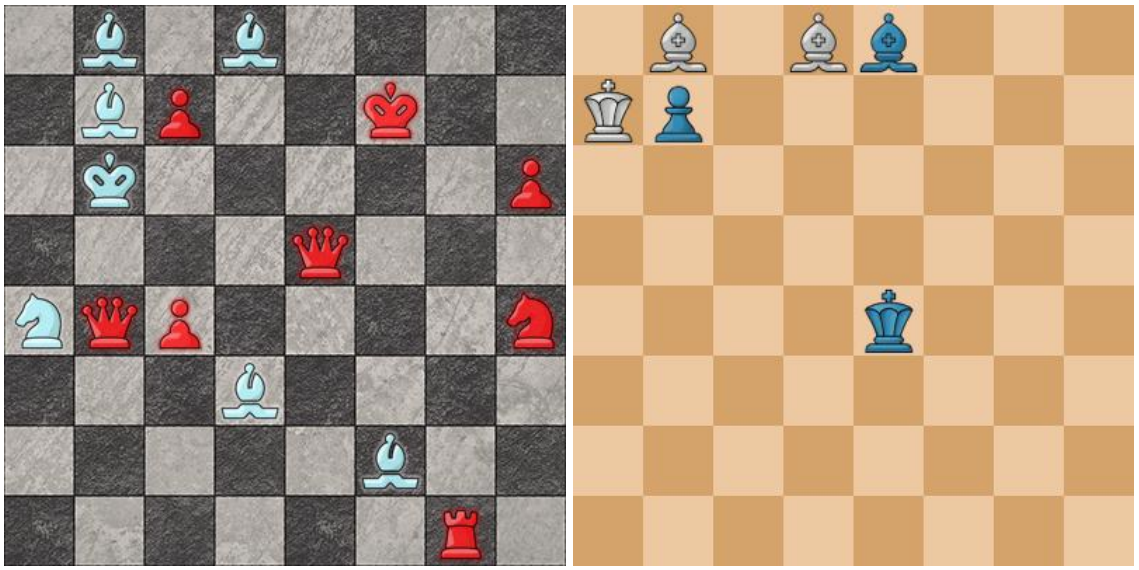
2. Priprema projekta

2.1 Pristup problemu

Na prvi pogled najočigledniji, naivni, pristup problemu bi predstavljao treniranje veštačke neuronske mreže nad celom šahovkom tablom. Kako imamo po 5 tipova figura,, u 2 boje, i na 64 mogućih pozicija, veoma je očigledno da bih naivni način postavke problema zahtevao previše podataka i bio previše komplikovan.

Takođe, u teoriji ne bi trebala da postoji nikakva zavisnost između stanja jednog polja i stanja bilo kog drugog polja. Međutim u praksi, kako po opisu dataseta svaka slika ima od 5 do 15 figura, postoji veoma velika šansa da bi ovo uticalo na performanse neuronske mreže.

Jedini izuzetak nezavisnosti između polja na tabli bi mogao da nastane prilikom odlučivanja boja figura. Ovo se primećuje u pojedinačnim stilovima figura, gde bi i ljudima bilo veoma teško odrediti boju figure bez da vide i ostale figure:



Slika: Stilovi figura sa neočiglednim bojama.

Međutim nije očigledno da li će nastati problem oko ovakvih situacija dok ne izvršimo testiranje. Samim tim, problem se zanemaruje za sada i uzima se pretpostavka da su polja na tabli međusobno nezavisna.

Samim tim, sa uvedenom pretpostavkom, pristup koji najviše ima smisla jeste treniranje neuronske mreže koja može da raspozna stanje na pojedinačnom polju. Ovaj pristup bi za predstavljanje stanja polja maksimalno zahtevao 13 klasa, 6 za svaku figuru, puta 2 za svaku boju, kao i jednu klasu za prazno polje.

2.2 Preprocesiranje podataka

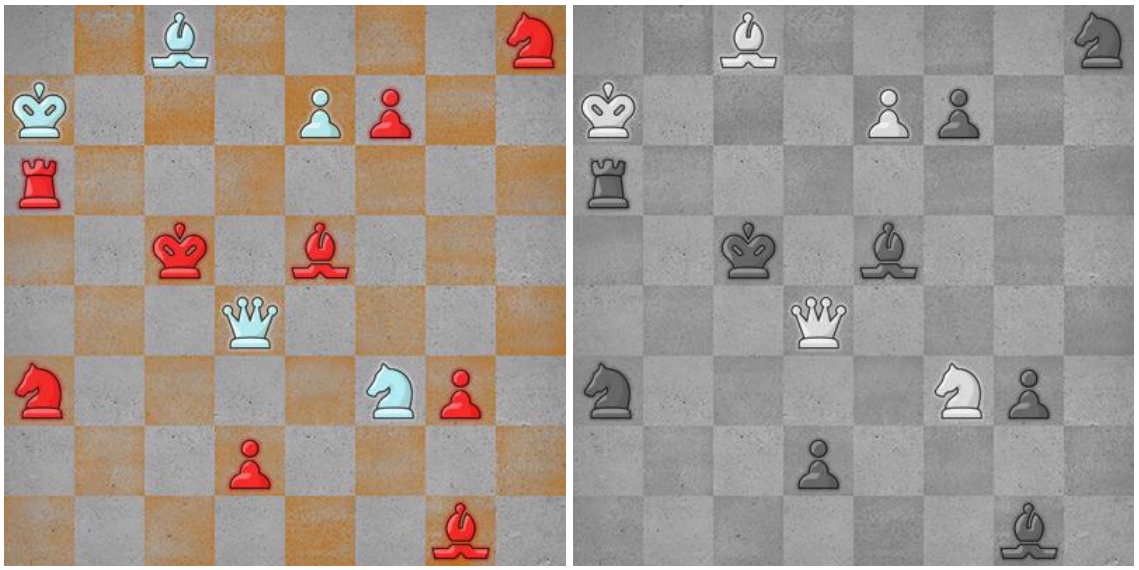
Kako vršimo prepoznavanje pojedinačnih polja umesto cele table, potrebno je izvršiti segmentaciju skupa podataka za treniranje. Preciznije, vrši se podela ulaznih slika šahovskih tabli, veličine 400x400 piksela, na 8x8 jednakih polja, veličine 50x50 piksela.

Za polja koja sadrže figure sačuvana su sva polja. Sa druge strane, za prazna polja sačuvana samo 10, po 5 za svaku boju polja. Ovo je urađeno zato što je primećeno da polja često nisu identična i sadrže patterne na nivou table koji bi možda ometali neuronskoj mreža ukoliko nisu bili više uračunati prilikom treniranja. Rezultat ove transformacije jeste dataset polja koji ukupno sadrži 1.5 miliona slika.

Kako je ovo bilo preveliko za mašinu koja je korišćena prilikom izrade projekta da bi se izvršila bilo kakva analiza u prikladnom vremenu, izvršen je nasumičan *sampling* slika. Zadržane su po deset hiljada slika za svaku klasu, gde klasa u ovom kontekstnu predstavlja svaku vrstu mogućeg stanja polja. Tačnije, obuhvata 12 klasa za svaku kombinaciju figure i boje kao i klasu za prazno polje. Ovo nam daje dataset od 130000 slika koji je korišćen tokom daljih analiza.

Dodati bonus ovog procesa jeste to što je dataset nakon ovoga u potpunosti balansiran, što je u praksi imalo veoma veliki uticaj na performanse.

Sve slike su takođe konvertovane u grayscale format tokom ovog procesa. Primarni razlog za ovu transformaciju predstavlja smanjenje ulazne veličine podataka na jednu trećinu. Međutim ovo takođe značajno olakšava određivanje boja figura u slučajevima kada bi to inače bilo značajno teže:



Slika: Stil figura sa ne-očiglednim bojama u:
originalu (levo) i transformisan u grayscale (desno)

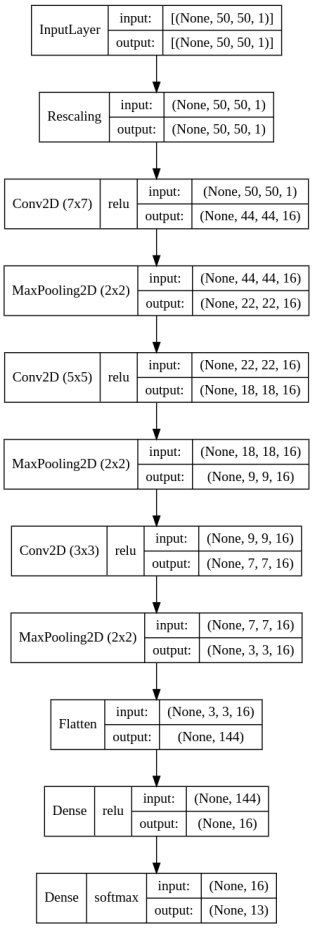
Ova konverzija je izvršena korišćenjem Pillow biblioteke, koja koristi ITU-R 601-2 luma transformaciju:

$$L = R * 299/1000 + G * 587/1000 + B * 114/1000$$

3. Testirana rešenja

3.1 Konvoluciona neuronska mreža

Pre svega, kako se radi o problemu klasifikacije slika, isprobana je standardna plitka konvoluciona neuronska mreža kao *baseline* model. Korišćeni su alternirajući 2D konvolucioni i max pooling slojevi sa gradualnim smanjenjem dimenzija konvolucionih filtera. Kao output sloj tokom treniranja je ostavljen sloj bez aktivacione funkcije, dok je tokom testiranja korišćena Softmax aktivaciona funkcija. Dijagram mreže se može videti na sledećoj slici:



Slika: Struktura korišćene konvolucione neuronske mreže.

Kako bi se pre sve isprobali najjednostavniji pristupi, klase nad kojima se vrši klasifikacija imale sve kombinacije mogućih figura i boja figura, kao i prazno polje.

Konvoluciona neuronska mreža je ubrzo nakon treniranja postigla preciznost od 100% i na skupu za treniranje kao i na validacionom skupu. Mreža je zatim testirana nad test skupom, gde je postigla istu preciznost.

Objašnjenje za ovo da jeste da mreža najverovatnije naučila sve moguće stilove još u training skupu. Kako je u opisu dataseta spomenuto da postoje 896 kombinacija stilova figura i

šahovske table, čak i sa značajno smanjenim skupom za treniranje pokrivamo gotovo sve moguće ulaze. Jedini šum bi najverovatnije dolazio od određenih stilova šahovske table koji variraju u zavisnosti od pozicije na tabli.

Kako bi sa trenutnom mrežom imali barem ekvivalentno ili bolje rešenje sa postojećih rešenja sa Kaggle, koji bi inače bili korišćeni za validaciju. Kao i zbog toga što nije moguće dalje poboljšati rezultate uz odobrenje profesora je fokus na dalje fokus stavljen na poboljšanje efikasnosti mreže.

3.2 Perceptron

Radi ispunjenja cilja pojednostavljanja mreže pre svega uklonjeni su konvolucioni kao i max pooling slojevi. Rezultat transformacije je standardna feed forward multi layer perceptron mreža, sa jednim skrivenim slojem. Kako je i ova mreža dobila identične rezultate, tj perfektnu preciznost i nad celim skupom podataka odlučeno je da se izvrši dalje uprošćenje neuronske mreže.

Sledeći vid uprošćenja je obuhvatao pokušaj da se ukloni čak i skriveni sloj neuronske mreže. Rezultujuća mreža predstavlja jednoslojnu perceptron mrežu.

Pokazalo se da čak i jednoslojni perceptron uspeva da postigne perfektnu preciznost nad ovim datasetom. Ova činjenica ukazuje na to da je problem kojim se bavimo u ovom projektu zapravo linearno separabilan. Samim tim, ovaj problem se najbolje može predstaviti, kao i optimizovati, kao problem linearne regresije i linearne algebre. Međutim kako ne bi previše odlutali van teme predmeta, ovi pristupi su zanemareni i fokus je umesto toga stavljen na vršenje optimizacija tehnikama koje bi bile aplikabilne i na drugim problemima iz dubokog učenja.

Jedna stvar koju bi treblao nagovestiti jeste da je brzina treniranja i kod jednoslojnog i višeslojnog perceptron neuronskih mreža bila značajno sporija od brzine treniranja kod konvolucionih neuronskih mreža. Konvolucione mreže su generalno dostizale perfektnu validacionu preciznost unutar prvih deset epoha, dok su mreže bez konvolucionih slojeva, sa istim vrednostima hiperparametara, zahtevale barem dvadeset do trideset. Samim tim, izbor između ove dve vrste mreža u većoj meri zavisi od toga da li je veći fokus stavljen na treniranje ili na izvršavanje mreže. Kod većih mreža se najčešće fokusiramo najpre na treniranje, pa zatim tek na vreme izvršavanja. Dok kod manjih mreža kao što je ova možemo lakše da stavim fokus na cenu izvršavanja. Ovaj problem je takođe rešen uz pomoć adaptivnog *learning rate* parametra, korišćenjem *ReduceLROnPlatou callback*-a.

3.3 Odvojeni perceptroni za klasifikaciju vrstu i boju figure

Trenutna veštačka neuronska mreža prolazi kroz svih 2500 piksela za sve težine svih 13 klasa. Rezultat ovoga jeste 2500 puta 13, tj 32500, operacija množenja. Kako ove 13 klase uključuju sve kombinacije vrsta figura i njihovih boja, najočigledniji način na koji možemo da pojednostavimo mrežu jeste da je odvojimo u dve zasebne mreže.

Zasebna mreža za klasifikaciju vrste figura bi samo morala da vrši klasifikaciju nad 6 vrsta figura, uključujući i prazno mesto. Dok sa druge strane, klasifikator boje figure može da se

predstavi kao binarni klasifikator koji bi zahtevao samo jedan izlaz. Samim tim, kombinacijom ove dve mreže smanjujemo broj množenja težina na 2500 puta 8, tj 20000, što predstavlja smanjenje od 40%.

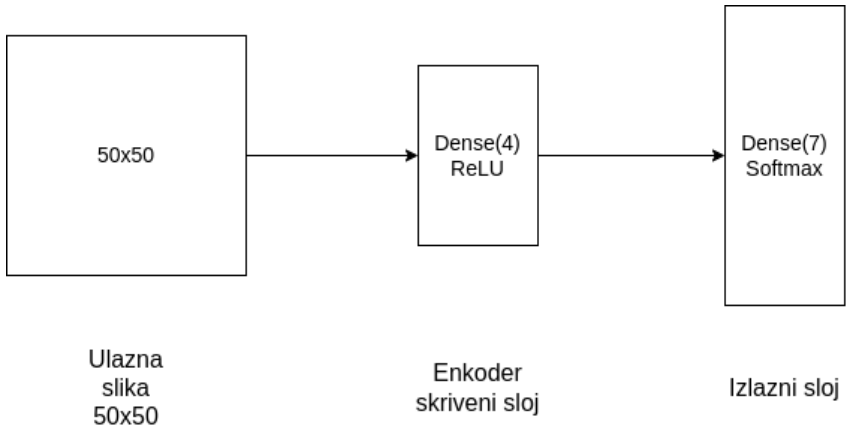
Pre svega rekreirani su skupovi podataka za ove dve mreže. Za klasifikator vrste figura su prvo spojene sve slike koje imaju istu figuru a različitu boju. Zatim je dodato još 10000 slika praznih polja kako bi skup podataka bio u potpunosti balansiran. Pre nego što je ovo balansiranje urađeno performanse mreža su bile značajno gore. Sa druge strane, za klasifikator su spojene sve slike koje imaju istu boju u dve klase, belu i crnu. Takođe su u potpunosti izbačene slike praznih polja, kako boja figure nije definisana u ovim slučajevima.

Kako se još uvek radi o jednoslojnim perceptronima, strukture ovih neuronskih mreža su gotovo identične prošloj, van broja klasa nad kojima vrše klasifikaciju. Jedina bitna promena jeste to što je aktivaciona funkcija izlaznog sloja kod klasifikatora boje figura promenjena u sigmoid funkciju, prateći preporučenu praksu za binarne klasifikatore.

Tokom treniranja obe neuronske mreže su postigle 100% preciznost nad validacionim i training skupom podataka. Zatim su isti rezultati dobijeni i na test skupu nakon treniranja, potvrđujući da nije došlo do pada performansi mreža tokom podele.

13.4 Skriveni sloj za enkodiranje

Najveći broj operacija se trenutno izvršava u klasifikatoru za vrstu figura. Preciznije zato što je neuron za svaku od 7 klasa zasebno povezan na svaki od 2500 ulaznih piksela. Jedan način da smanjimo ovaj broj jeste da uvedemo skriveni sloj širine n, gde je n manje od 7. Samim tim za svaki korak za koji je n manje od 7 smanjujemo veličinu mreže za 2500, dok je povećavamo za samo 7*n. Za ovaj skriveni sloj onda možemo da kažemo da kompresuje ili enkoduje originalnu sliku u format koji je značajno kraći, veoma nalik skrivenom sloju u Auto Encoder mrežama.



Slika: Struktura MLP mreže sa skrivenim slojem za enkodiranje.

Isprobani su skriveni slojevi veličina 2, 3, 4 i 5. Uspešni rezultati su dobijeni za n = 4 i 5, smanjujući minimalni broj parametra na 10039. Za skriveni sloj veličine 3 isprobane su razne konfiguracije neuronskih mreža sa dodatnim slojevima koji bi se ponašali kao dekoderi. Čak i sa ovim dodatnim slojevima mreža bi u globalu bila manja zbog smanjenja od 2500 parametra

na prvom skrivenom sloju. Međutim čak i kada bi ove mreže uspele da postignu validacionu preciznost od 100% tokom treniranja, na trening skupu bi uvek imali veliki broj grešaka.

Tokom treniranja ove grupe mreža primećeno je da u potpunosti nisu dobijale na performansama tokom treniranja dok learning rate hiper parametar nije smanjen za jedan do dva reda magnitude od uobičajnog. Međutim nakon toga bi trenirale gotovo istom brzinom kao i perceptroni.

Ova optimizacija nije iskorišćenja na neuronskoj mreži za klasifikaciju boje figura, kako nije moguće ubaciti sloj za enkodiranje sa veličinom manjom od 1.

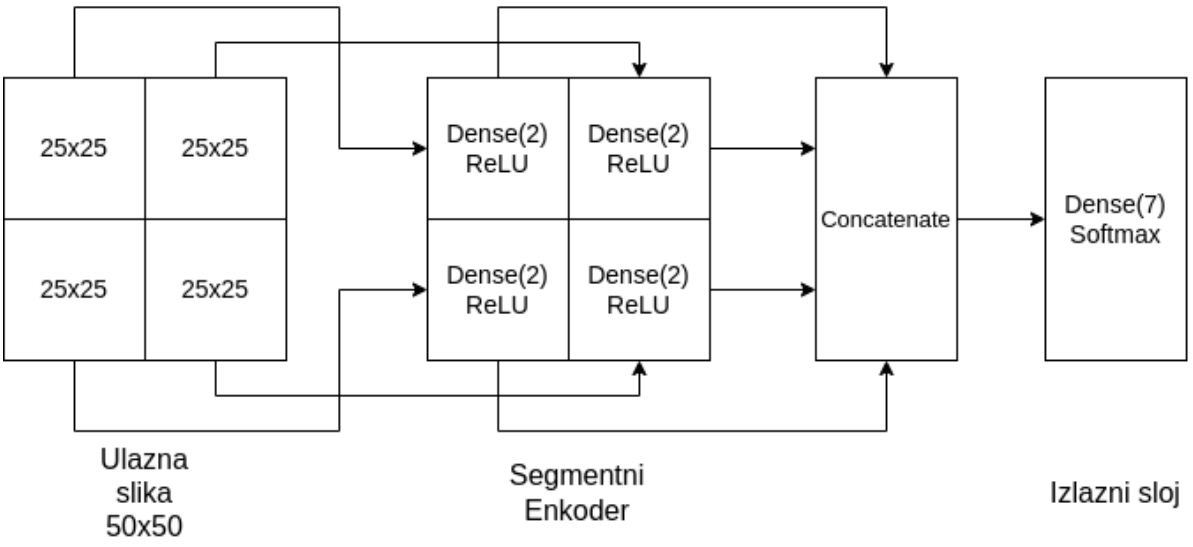
3.5 Segmentiran sloj za enkodiranje

Kako je limitirajući faktor za veličinu neuronske mreže ponovo to što su svi neuroni u prvom ne ulaznom sloju mreže povezani sa svih 2500 vrednosti piksela slike uvedena je još jedna optimizacija. Ulazna slika je segmentirana na MxM jednakih segmenata koji su zatim prosleđeni zasebno na MxM matricu dense slojeva veličine N. Rezultati predikcija ovih slojeva se zatim konkateniraju u listu veličine W i prosleđuju dalje.

Primera radi, ukoliko uzmemo vrednosti W=2 i H=1, imaćemo matricu 2x2 matricu dense slojeva veličine 1, gde će svaki biti povezan na samo jedan, zaseban, segment ulazne slike, gde bi u ovom slučaju ta veličina bila 25x25, tj 625. Tako da bi broj težina za ovaj skup slojeva ukupno bio 4x625, ili 2500. Ovo bi predstavljalo smanjenje od 4x u odnosu na trenutno najbolje rešenje. Dok bi sa druge strane imala isti broj izlaznih parametra enkodera (4) kao i najbolje rešenje.

U teoriji ovi paralelni slojevi bi mogli da se uže specijalizuju nego što je obično moguće, kako mogu da se samo fokusiraju na jedan deo slike. Međutim takođe se mora nagovestiti da je ova optimizacija jedino moguća zato što su slike relativno slične, i generalno centrirane. Ukoliko se zahteva otpornost na translaciju performanse ova tehnike verovatno ne bi bila najbolja.

Isprobane su različite vrednosti za W i H. Minimalna neuronska mreža koja je dobila perfektnu preciznost bila je mreža sa W=2 i H=2. Segmentiran enkoder sloj je direktno prosleđen output sloju bez ikakvih slojeva za dekodiranje. Samim tim, mžera je u celini sadržala 5071 parametra, što predstavlja smanjenje od približno dva puta u odnosu na mrežu bez segmentiranja.



Slika: Struktura mreže sa segmentnim enkoderom

Do sada su svi slojevi koji vrše enkodiranje segmenata bili homogenih veličina. Međutim kada su isprobane matrice slojeva ne homogenih veličina, nađena je nova najmanja matrica slojeva za enkodiranje koja ostvaruje perfektnu preciznost. Ova mreža vrši segmentiranje slike na 4x4 (W=4, H=4) segmenata sa sledećim brojevima neurona za slojeve koji vrše enkodiranje tih segmenata:

1	1	1	1
1	2	2	1
1	2	2	1
1	1	1	1

Slika: Matrica podele broja neurona u matrici za enkodiranje segmenta.

Izlaz iz enkoderskog sloja povezan je direktno na izlazni sloj neuronske mreže. Navedena mreža je u totalu imala svega 3047 parametra, i ukupno vršila 2999 množenja, što predstavlja dalje smanjenje od 40%.

Takođe je isprobana verzija sa filterom kako bi zadati segmenti mogli da se u potpunosti zanemare. Međutim izbacivanje bilo kojeg segmenta iz prethodno prikazane matrice bi navelo mrežu da pravi greške na test setu, čak iako bi dobijale validacionu preciznost od 100%. Samim tim, ova ideja je zanemarena za sada.

3.6 Horizontalna simetričnost figura

Dalje je primećeno kako većina figura poseduju horizontalnu simetričnost, ili tačnije u odnosu na y-osu. Izuzetak ovome bi bile nesimetrične figure kao što su konj i ponekad lovac, kao i neki stilovi koji imaju senčanje koje nije simtetrično:

Slika: Primeri nesimetričnih stilova.

Šta ide protiv ideje da možemo da se oslanjamo na simetričnost figura jeste to što predhodno spomenuti pokušaj ignorisanja bilo kog segmenata nije uspeo da dobije perfektnu preciznost. Međutim takođe je moguće i da su slojevi za segmente koji bi trebali da budu simetrični ustvari vršili različite uloge.

Kako ne bi prosto odbacili pola slike, izvršena je transformacija u tri koraka:

- Slika je prvo podeljena na dva dela, levi i desni.
- Desni deo je okrenut (flipped) po y-osi.
- Levi i desni deo su sumirani i prosleđeni dalje.

Ova transformacija implementirana je u vidu custom tensorflow sloja.

Ideja sa ovom transformacijom jeste da se prepolovi broj ulaznih piksela bez da se izgube podaci koji nisu simetrični. U principu, zamenjujemo pola od ukupnog broja operacija množenja sa ekvivalentnim brojem sabiranja, koje je u praksi značajno brže.
Takođe se mora spomenuti da se kod mreža koje koriste ovaj sloj rescaliranje mora prilagodi kako bi vrednosti bile u opsegu [0,1]. Ovo bi mogli da zaobiđemo korišćenjem srednje vrednosti umesto sume u transformaciji. Međutim to bi dodalo dodatne operacije množenja svakog puta kada koristimo ovaj sloj. Dok sa druge strane ukoliko koristimo sumu sa prilagođenim slojem za reskaliranje takođe smanjujemo i broj deljenja koje i sloj za reskaliranje mora da izvrši.

Ovaj sloj u nekoj meri predstavlja modifikovanu verziju Average Pooling 2D sloja sa dodatom flip operacijom. Samim tim moguće je zameniti operaciju sumiranja sa max ili min operacijom kako bi se umesto Average Pooling simulirao Max ili Min Pooling sloj. Max verzija bi možda imale bolje rezultate s'obzirom da segmentni enkoder koristi ReLU slojeve. Međutim zbog manjka vremena ove opcije su zanemarene za sada.

Pre svega je testirana transformacija sa jednoslojnim perceptronom za klasifikaciju figura kao baseline. Dobitkom potpune preciznosti sa tim modelom potvrđujemo da je problem jošuvek linearno separabilan i da nismo izgubili previše informacije.

Zatim je vršeno testiranje ove transformacije nad mrežama sa segmentiranim enkoderom. Veličine matrice W i H su fiksirani na W=4 i H=2, dok je fokus stavljen na nalaženje distribucije broja neurona po slojevima koja bi održala potupunu preciznost nad test skupom. Na kraju je nađena sledeća distribucija:

1	1
1	4
1	3
1	3

Slika: Matrica podele broja neurona u segmentnom enkoderu sa najmanjim nađenim brojem neurona.

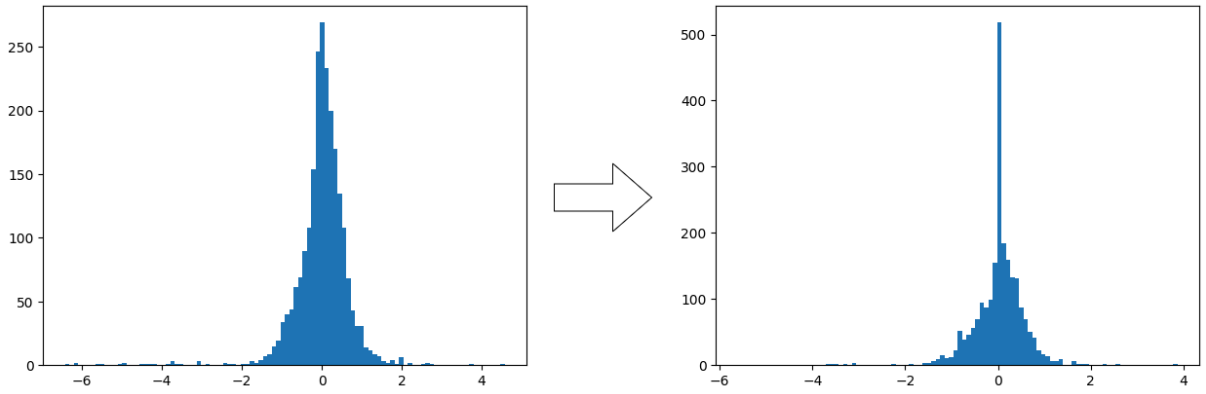
Segmentni enkoder je ponovo povezan direktno na izlazni sloj. Rezultujuća neuronska mreža sadrži ukupno 2287 parametra, što predstavlja dalje smanjenje od 25% u odnosu na prošlo rešenje.

3.7 L1 Regularizacija

L1, ili Lasso, regularizacija predstavlja jednu od najčešće korišćenih metoda regularzacije. Karakteristika koja je za ovaj projekat bitna jeste to što, za razliku od L2 regularizacije koja podstiče da se težine rasporedele što je više ravnomerno, L1 regularziacija podstiče težine neuronske mreže da budu što je više proređene. Ukoliko je dovoljan broj težina rezultujuće mreže jednak nuli možemo značajno ubrzati operaciju klasifikacije.

Korišćena je ista struktura kao i u trenutnoj najboljoj mreži, sa dodatom L1 regularizacijom, kako bi se istrenirala nova, proređena mreža. Međutim, glavni problem kod korišćenja ove tehnike u našem slučaju jeste to što ne želimo da efekat regularizacije negativno utiče na preciznost neuronske mreže. Samim tim njen faktor je postepeno smanjivan dok mreža nije ponovo dostigla potpunu preciznost, što je postignuto sa faktorom L1 regularizacije od 1e-6.

Ova vrednost deluje veoma nisko, međutim ipak je imala vidljiv efekat na distribuciju težina u neuronskoj mreži, gde primećujemo značajno veću količinu vrednosti oko nule:



Slika: Histogram distribucije magnituda težina, na neuronskoj mreži bez L1 regularizacije (levo), kao i sa L1 regularizacijom (desno).

Međutim broj težina koji je zapravo jednak nuli je zapravo 0, ali ovo ne predstavlja problem kako je model sa L1 regularizacijom odrađen samo kao priprema za kasnije orezivanje mreže.

3.8 Optimizacija klasifikatora boje figura

Sledeća stvar koja je optimizovana jeste neuronska mreža za klasifikaciju boju figura. Korišćene su slične tehnike kao i kod klasifikatora vrste figura tako da će samo biti pređene ukratko.

Pre svega, jedina nova optimizacija jeste dodavanje CenterCrop sloja, koji vrši centralno isecanje slike. Ovo može značajno da smanji količinu ulaznih podataka u mrežu. Minimalna veličina isečka koja je nađena koja održava potupunu preciznost jeste 26x22. Ovo samo po sebi smanjuje broj ulaznih podataka od 2500 na 572.

Dalje iskorišćena je transformacija spomenuta u poglavlju 3.6. Međutim, u ovom slučaju izvršena je dva puta, po x i po y osi. Ovim dalje smanjujemo dimenzije ulaznih podataka na 13x11,rezultujući u mrežu sa sve ukupno 144 parametra.

Nisu korišćeni nikakvi slojevi za enkodiranje, kako u ovom slučaju imamo samo jedan izlazni neuron. Iako bi segmentni enkoder smanjio količinu parametra, dobitci bi bili zanemarljivi, dok bi sa druge strane kompleksnost mreža bila značajno veća.

Najzad izvršeno je treniranje najbolje nađene strukture neuronske mreže mreže sa dodatnom L1 regresijom, ponovo sa faktorom od 1e-6.

3.9 Orezivanje mreža

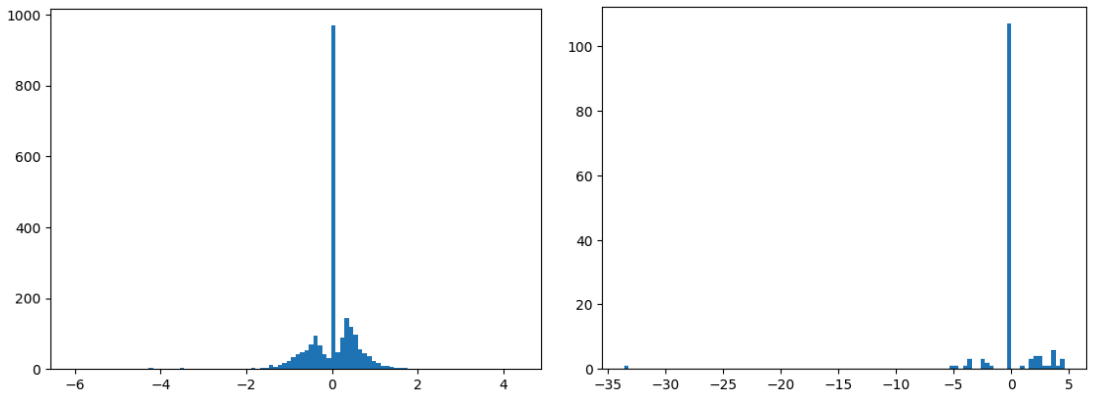
Orezivanje (*pruning*) veštačke neuronske mreže predstavlja proces u kome se određeni procenat parametra mreže sa najmanjim magnitudama, tj apsolutnim vrednostima, forsiraju na 0 i njihovo dalje treniranje se isključuje. Neuronska mreža se zatim ponovo trenira, ili *fine-tune*-uje kako bi mreža naučila da funkcioniše čak i dok je zadati procenat parametra jednak nuli. Ovaj proces se najčešće ne izvršava u jednom koraku, već se parametri postepeno isključuju na osnovu plana orezivanje (pruning schedule).

Orezivanje se u praksi koristi kako bi se smanjila memorija koju model zauzima, kako bi se poboljšale performanse mreže, i kao vid regularizacije.

Postoje više različitih vrsta orezivanja kao što su: orezivanje težina, orezivanje neurona, sparse orezivanje i struktuisano orezivanje. Specifično za ovaj projekat, od interesa je orezivanje težina, zato što samim tim smanjujemo i broj množenja koji mreža treba da izvrši.

Za ovu svrhu korišćena je implementacija u *Tensorflow model optimization* biblioteci. Specifično sa *Polynomial Decay* planom orezivanja, kako je generalno preporučen kao najbolji.

Pretraga za najveći procenat retkosti izvršena je iterativno, gde je smanjivana dok model ne dobije potpunu preciznost nakon orezivanja. Za klasifikator tipa figura ovaj procenat je bio 40%, dok je kod klasifikatora boje figura bio čak 75%. Postoje nekoliko objašnjenja za ovu razliku. Pre svega kako je klasifikator boje jošuvek perceptron sa dosta preprocesiranja podataka, značajno je jednostavniji u odnosu na klasifikator tipa figure. Samim tim kako postoje manje zavisnosti između težina lakše je ukloniti manje bitne težine bez da se ugroze performanse neuronske mreže. Drugo objašnjenje jeste da je kod klasifikatora boja bilo moguće jos više smanjenje broja neurona, dok je klasifikator tipa figure bio značajno bliži minimumu.



Slika: Distribucija težina u mrežama nakon orezivanja:
Klasifikator tipa figure (levo) i klasifikator boje figure (desno)

Sa ovim korakom dobijamo finalnu verziju ovih klasifikatora. Klasifikator tipa figure na kraju ima ukupno 2287 parametra, od kojih su 40% jednaki nuli, dajući nam otprilike aktivnih 1372 parametra. Dok sa druge strane, klasifikator boje figure ima ukupno 144 parametra, od kojih su oko 36 aktivni. Samim tim, broj operacija koje bi procesor trebao da izvrši je, veoma uprošćeno, blizak 1400 operacija po ploči. Ovo nam daje ukupno oko 90000 operacija da bi se procesirala cela šahovska tabla, što bi trebalo da bude veoma brzo čak i na sporijim procesorima.

4. Validacija

Kako je postignuta preciznost od 100% i na training kao i na test skupovima podataka, nije stavljen preveliki fokus na validaciju rešenja.

Pre svega, analizirana su rešenja drugih korisnika sa Kaggle sajtu. Fokus je stavljen na 3 rešenja, pre svega po vrstama pristupu problemu, a zatim po glasovima:

- Peter Kerman (meditech101)
- ashwinbhatt
- Jannik Labs

Rešenje korisnika Peter Kerman predstavlja kako je većina korisnika odradila ovaj zatak. Ova klasa rešenja je koristila konvolucione neuronske mreže, generalno komplikovaniju verziju od one koja je analizirana tokom ovog projekta. Takođe su vršili predikciju nad spojenim klasama za boju i figuru, tj nad 13 klasa. Generalno su svi iz ove klase rešenja dobili perfektnu preciznost, međutim ukoliko je korišćena prevelika regularizacija nastaju greške.

Petar Kerman-ovo rešenje specifično je specijalno po tome što koristi custom loss funkciju u vidu weighted categorial crossentropy. Težine koje su korišćene za ovu funkciju su samo date i njihov izvor nije objašnjen.

Rešenja korisnka ashwinbhatt je jedno od jedinih koje je koristilo MLP neuronsku mrežu za svoje rešenje. Doduše njegovo rešenje koristi značajno veću MLP mrežu sa 6 slojeva i značajno više neurona po sloju. Test preciznost ove mreže je bila 99.8%. Najverovatniji razlog zašto njegovo rešenje nije dobilo 100% je ili do toga što je izvršeno smanjenje slike na 1/2 na nivou ploče. Ovo je najverovatnije unelo dovoljno noise na nivou ploča da nije bilo moguće dobiti potpunu preciznost. Takođe je moguće da je i zbog toga što nije kreirao balansiran skup podataka ploča, kako je i ovo značajno uticalo na performanse mreže tokom ovog projekta.

Rešenje Jannik Labs je takođe koristili konvolucione mreže. Pre svega, za razliku od ostalih korisnika koristili su PCA kako bi ukolini noise sa slika. Takođe su jedni od jedinih nađenih rešenja koji su radili i predikciju nad celom šahovskom pločom. Međutim rezultujuća neuronska mreža je imala test preciznost od svega 87%.

5. Zaključak

Naprkos tome što je perfektna preciznost postignuta još sa prvom neuronskom mrežom, mnogo toga je naučeno tokom izrade ovog projekta. Naime, fokus na optimizaciju metrika van uobičajnog ciljanja za najveću preciznost. Kao i razmišlanje van okvira radi uprošćavanja mreže, što je bio slučaj sa slojevima za segmentno enkodiranje i flip transformacija.

Dosta toga je naučeno o unutrašnjem radu *Tensorflow* biblioteke tokom procesa implementacije *custom* slojeva kako bi se postigle metode opisane u projektu. Kao i o *Tensorflow model optimization* biblioteci, čak i ako je tokom projekta samo korišćen podmodul za orezivanje.

Mreža koja je na početku projekta imala ukupno 32000 parametra na kraju projekta smanjena je na klasifikator tipa figure koji ima 2287 parametra, od kojih su oko 1372 aktivni, tj nisu nule, kao i na klasifikator boje figure, koji ima 144 parametra, od kojih su oko 36 aktivni. Ukoliko gledamo samo aktivne parametre, ovo predstavlja smanjenje od otprilike 95.6%. Detaljan pregled se može videti na sledećoj tabeli:

	Klasifikator figure:	Klasifikator boje:	Ukupno:	Relativno Smanjenje (%):	Ukupno smanjenje (%):
Konvoluciona neuronska mreža (*):	-	-	12077	-	-
Perceptron:	-	-	32513	-	-
Odvojeni perceptroni:	17507	2501	20008	38.46	38.46
Skriveni sloj za enkodiranje:	10039	2501	12540	37.33	61.43
Segmentiran sloj za enkodiranje:	3047	2501	5548	55.76	82.94
Horizontalna simetričnost:	2287	2501	4788	13.70	85.27
Optimizacija klasfikatora boje:	2287	144	2431	49.23	92.52
Orezivanje mreža (aktivni):	1372	36	1408	42.08	95.67

(*) - Konvoluciona neuronska mreža sadrži manje parametra, međutim vrši više operacija.

Tabela: Pregled broja parametra po poglavlju.

Gotovo sigurno je moguće dalje uprostiti neuronsku mrežu. Specifično detaljnijom pretragom optimalne raspodele segmenata u segmentnom enkoderu, ili fokusiranjem na još veću dimenziju matrica segmentnog enkodera. Takođe i isprobavanjem aktivacionih funkcija van ReLU funkcije za slojeve u segmentnom enkoderu. Van toga, veoma moguće da je max operacija bolja kod flip transformacije umesto sume, posebno sa ReLU funkcijama u slojevima segmentnog enkodera.

Međutim sa dobijenim rezultatima trebalo bi biti lako implementirati verziju koja bi mogla da radi veoma brzo čak i na slabijim procesorima. Posebno kako je kod veoma pogodan za tehnike kao što su mutlithreading ili SIMD. Čime se računa da je postignut cilj optimizacije efikasnosti ove neuronske mreže.