SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 2295

OPTIMIRANJE KONVOLUCIJSKIH MODELA ITERATIVNIM PODREZIVANJEM

Stjepan Močilac

Zagreb, lipanj 2020.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 2295

OPTIMIRANJE KONVOLUCIJSKIH MODELA ITERATIVNIM PODREZIVANJEM

Stjepan Močilac

Zagreb, lipanj 2020.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 13. ožujka 2020.

DIPLOMSKI ZADATAK br. 2295

Pristupnik:	**Stjepan Močilac (0036487832)**
Studij:	Računarstvo
Profil:	Računarska znanost
Mentor:	prof. dr. sc. Siniša Šegvić

Zadatak: **Optimiranje konvolucijskih modela iterativnim podrezivanjem**

Opis zadatka:

Konvolucijski modeli su dominantna paradigma za vizualno raspoznavanje. Međutim, praktične primjene moraju se suočiti s ogromnom računskom složenosti odgovarajućih implementacija. Taj problem može se umanjiti podrezivanjem mapa značajki te finim ugađanjem novo dobivenih modela. Ovaj rad razmatra hipotezu dobitne kombinacije lutrije prema kojoj podrezani model najbolje uči ako se incijalizira jednako kao i početni model na temelju kojeg je provedeno podrezivanje. U okviru rada, potrebno je proučiti konvolucijske arhitekture za klasifikaciju slika. Oblikovati klasifikacijski model za raspoznavanje na skupu CIFAR-10 te uhodati iterativni postupak podrezivanja. Validirati hiperparametre, prikazati i ocijeniti ostvarene rezultate te provesti usporedbu s rezultatima iz literature. Predložiti pravce budućeg razvoja. Radu priložiti izvorni kod razvijenih postupaka uz potrebna objašnjenja i dokumentaciju. Citirati korištenu literaturu i navesti dobivenu pomoć.

Rok za predaju rada: 30. lipnja 2020.

SADRŽAJ

1. Uvod

Konvolucijske neuronske mreže (CNN) su temelj u rješavanju problema računalnog vida. Iako postižu dobre rezultate u području klasifikacije, detekcije i segmentacije, mana su im visoki računski zahtjevi zbog kojih se teško integriraju na ugradbene uređaje i uređaje manje potrošnje. Ovaj rad se bavi pronalažanjem rješenja tog problema postupkom podrezivanja. Zadatak podrezivanja konvolucijskih mreža je pronaći i ukloniti parametre (Slika 4.1) koji su najmanje relevantni za točnost i generalizaciju modela uz takvu inicijalizaciju parametara, arhitekturu i metodu učenja. Benefiti koje podrezivanje (potencijalno) donosi su redukcija kapaciteta modela, smanjenje računske složenosti te ubrzavanje izvođenja. Ponekad, moguće je postići i veću točnost te bolju generalizaciju modela nakon podrezivanja [3]. Većina programskih okvira ne zna profitirati na jednostavnom podrezivanju težina, pa je u tim slučajevima potrebna specijalna programska ili sklopovska podrška. Upravo radi toga, u ovom radu istražujemo strukturno podrezivanje, odnosno podrezivanje cijelih kanala. Uz podrezivanje, za kompresiju modela često se koristi i kvantizacija.

2. Duboki konvolucijski modeli

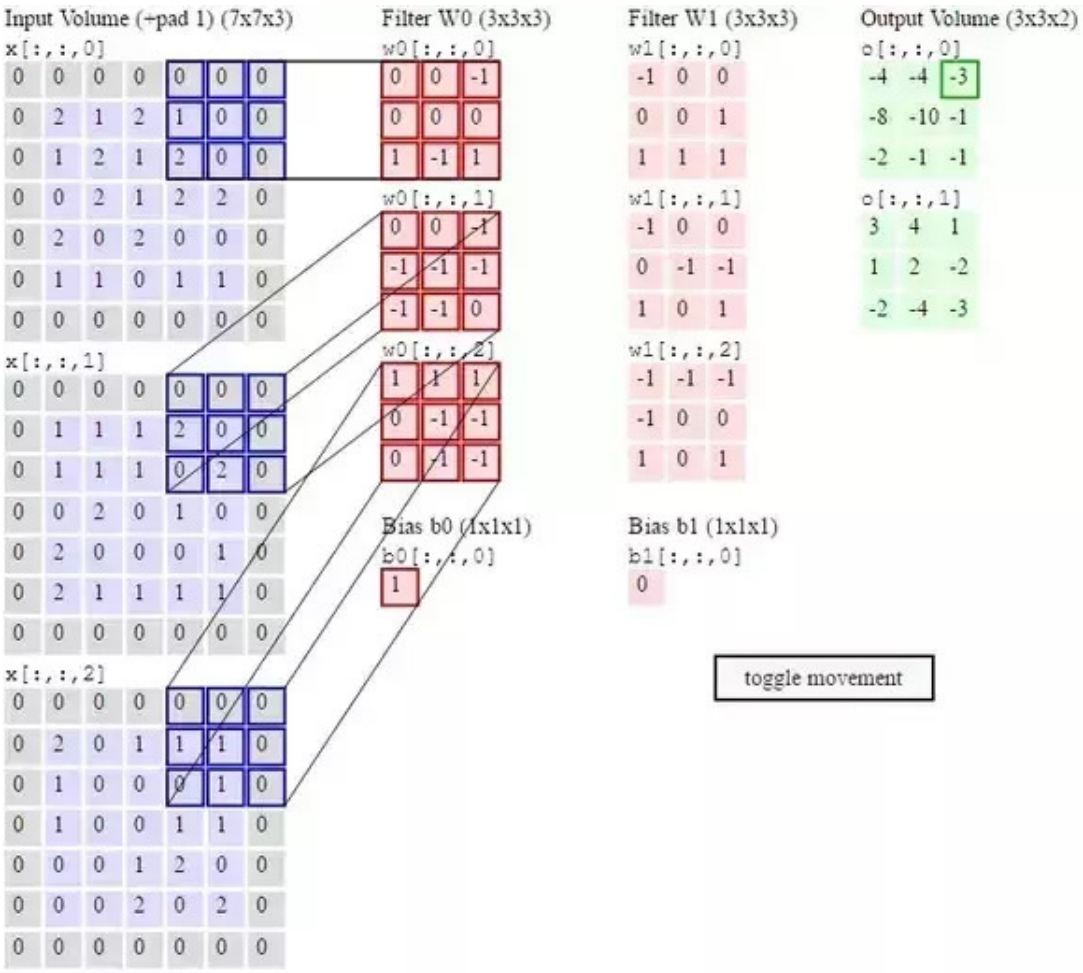
Konvolucijske neuronske mreže su duboke neuronske mreže specijalizirane za obradu slika, a temelje se na:

- rijetkoj povezanosti,
- dijeljenju parametara i
- ekvivarijantnosti s obzirom na translaciju

Rijetka povezanost je uvedena kako bi se smanjio broj težina modela, a ostvaruje se zaduživanjem svakog neurona samo za dio slike, umjesto za cijelu. Invarijantnost na translacije unutar slike je postignuta time da svaki neuron ima više izlaza gdje svaki odgovara odzivu na drugom položaju na slici. Na taj način se ujedno postiže i dijeljenje težina neurona duž cijele slike. Konvolucijske mreže se tipično grade od konvolucijskih slojeva, slojeva sažimanja te potpuno povezanih slojeva.

2.1. Konvolucijski sloj

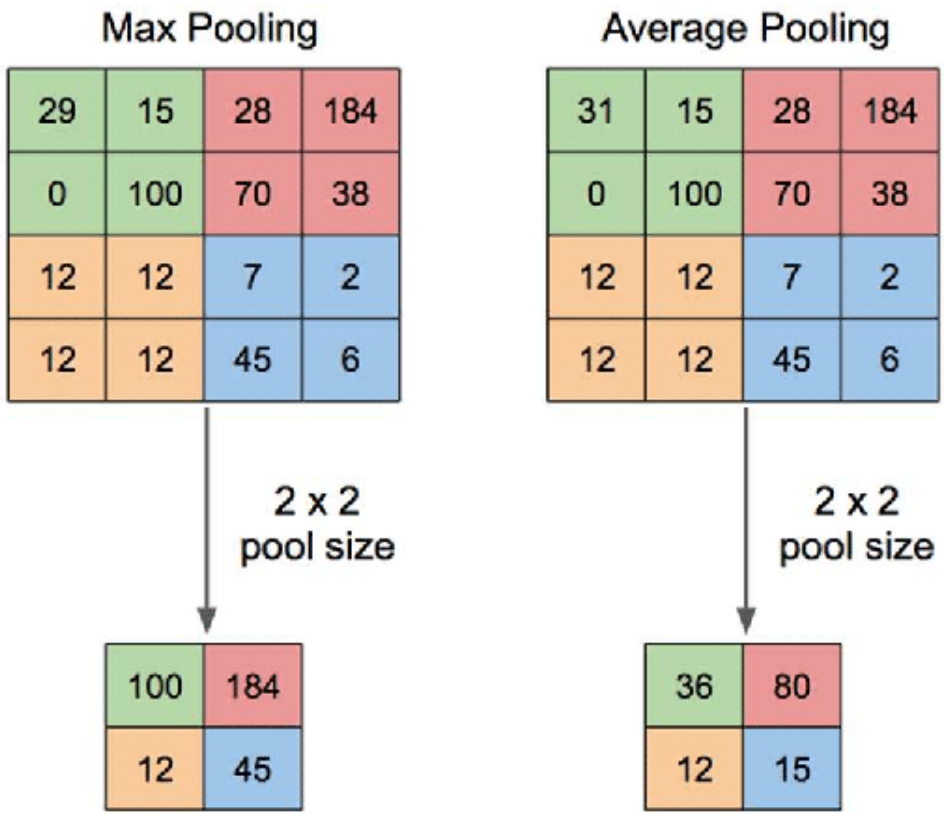
Da bi problem podrezivanja u ovakvim arhitekturama bio jasniji, krenut ćemo s generalnim opisom konvolucijskog sloja. Parametri konvolucijskih slojeva su tenzori težina koji se nazivaju filtri (konvolucijske jezgre) i vektori pomaka (engl. bias). U svakom konvolucijskom sloju postoji više filtra, a svaki od njih zadužen je za pronalazak neke druge značajke na slici. Kod obrade slike, konvolucijske jezgre predstavljamo tenzorima trećeg reda (npr. 3x3x3). Konvolucijska jezgra "se pomiče po slici" za korak K (Slika 2.1). Izlaz svakog od tih pomaka je suma svih elemenata matrice dobivene Hadamardovim umnoškom prozora slike sa konvolucijskim filtrom uvećanim za vektor pomaka. Ova operacija se naziva unakrsna korelacija, a njen rezultat je mapa značajki koja predstavlja odziv tog filtra duž cijele slike.



Slika 2.1: Matrice označene plavom bojom prikazuju 3 ulazna kanala dimenzija 7x7. Svaki filter kreira po jednu izlaznu mapu označenu zelenom bojom. Takva mapa se dobije sumiranjem rezultata po kanalima za svaki od filtera. Linijama je povezana vizualizacija načina na koji se obavlja množenje u konvolucijskom sloju.

2.2. Sloj sažimanja

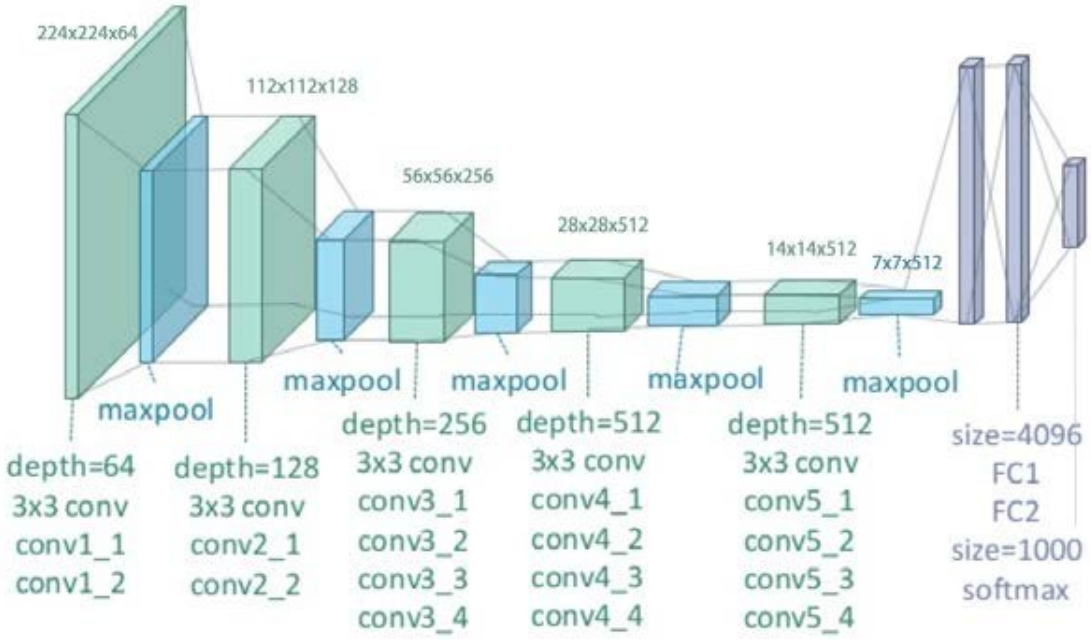
Konvolucijska mreža bolje radi kada se između konvolucijskih slojeva umeću slojevi sažimanja gdje se dijelovi mapa značajki grupiraju i transformiraju u jednu vrijednost. Time se postiže smanjivanje prostorne veličine mapa značajki i reduciranje broja parametara te očuvanje na invarijantnost prema malim translacijama piksela. U ovom radu podrazumijevamo da se koristi sažimanje maksimumom (eng. max-pooling, Slika 2.2).



Slika 2.2: U gornjem dijelu su ulazne mape. Vidimo da sažimanje maksimom uz korak 2 iz svakog potprozora uzima najveću vrijednost, odnosno iz zelenog prozora to je vrijednost 100. Desni stupac pokazuje nešto manje popularnu metodu gdje se iz svakog prozora uzima prosječna vrijednost. Primjerice plavi prozor ima sumu 60, odnosno srednju vrijednost 60/4=15.

2.3. Potpuno povezani sloj

Da bismo iz značajki dobivenih konvolucijskim slojevima i slojevima sažimanja dobili konkretnu predikciju, na kraj mreže se u pravilu stavlja jedan ili više potpuno povezanih slojeva čime se povećava receptivno polje izlaznih značajki. Primjer jedne takve mreže je VGG-19 (Slika 2.3).



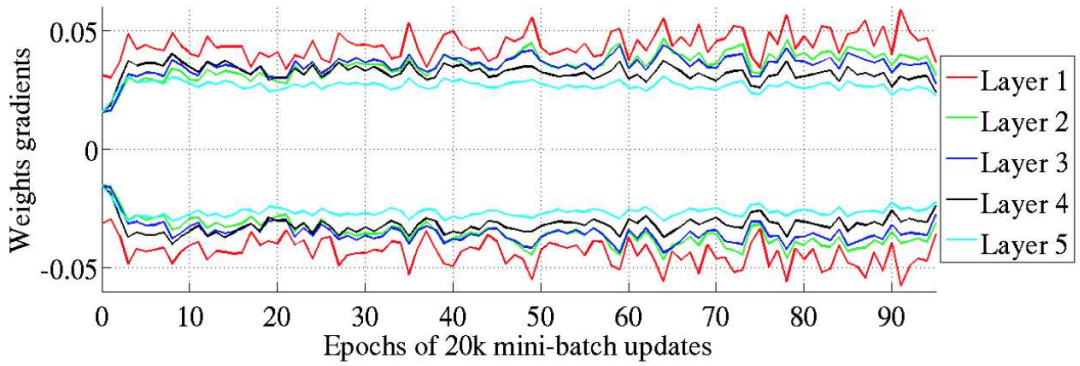
Slika 2.3: Arhitektura mreže VGG-19.

2.4. Metode inicijalizacije slojeva

Inicijalizacija parametara dubokih modela vrlo je bitna stavka pri njihovom treniranju. Kada govorimo o ovoj temi, postoje dva osnovna problema koja se u praksi javljaju pri lošoj inicijalizaciji:

- Nestajući gradijent - gradijent gubitka je premali te se slojevi bliže početku mreže ne ažuriraju (vrijednosti tenzora su nule)
- Eksplodirajući gradijent - gradijent gubitka je prevelik da bismo imali benefita od prolaska unatrag (vrijednosti tenzora su NaN vrijednosti)

Upravo radi tih problema dolazimo do motivacije za normalizaciju pri inicijalizaciji težina. Može se matematički pokazati da standardnom normalnom distribucijom težina W, njihov umnožak sa ulazom x, će u prosjeku imati standardnu devijaciju koja iznosi $\sqrt{n}$, gdje je n broj ulaznih veza. Pokazalo se da inicijalizacija standardnom normalnom distribucijom u rasponu [-1, 1] te skaliranjem sa $1/\sqrt{n}$ također dovodi do infintenzimalno maloga (skoro nestajućeg) gradijenta u danim primjerima. Upravo ovim dolazimo do Xavier inicijalizacije koju su predložili Glorot i Bengio [4] gdje težine sloja postavljamo na slučajne vrijednosti u rasponu $\pm\frac{\sqrt{6}}{\sqrt{n_i+n_{i+1}}}$, gdje su n_i broj ulaznih veza te n_{i+1} broj izlaznih veza tog sloja. Eksperimentalno je pokazano da takva inicijalizacija omogućuje gotovo jednaku varijancu kroz pet slojeva dubine (Slika 2.4).



Slika 2.4: Prikaz varijance težina kroz 5 slojeva mreže.

3. Kompresija mreže

Duboki modeli imaju jako velik broj parametara. Parametri današnjih neuronskih mreža zahtjevaju i po nekoliko stotina megabajta (kod ekstremnijih pričamo čak i o gigabajtima). Proces treniranja takvih modela u svrhu postizanja razumnih rezultata je vremenski vrlo zahtjevan. Problem smanjivanja računskog troška počinje biti sve više aktualan za aplikacije koje se izvode u stvarnom vremenu poput on-line učenja [12]. Dodatno tome, posljednjih godina bilježi se značajan tehnološki napredak u područjima koja zahtijevaju izvođenje dubokih modela na uređajima ograničenih resursa (memorije, CPU, energije). Postoji više tehnika koje se bave reduciranjem veličine dubokog modela. Generalno to su metode podrezivanja, kvantizacije i dijeljenja težina te SVD dekompozicije matrice težina.

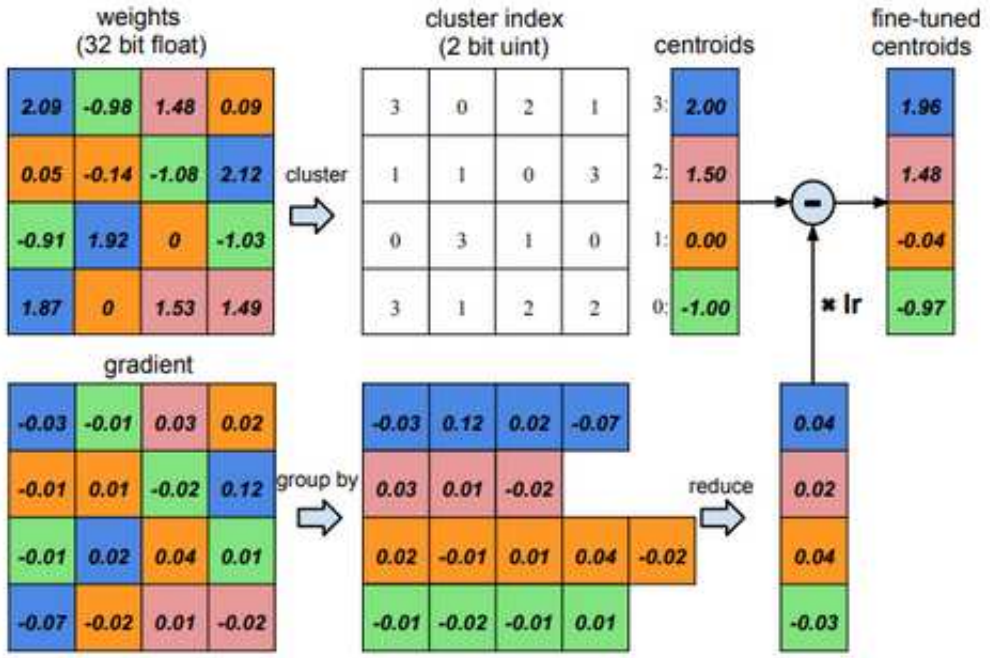
3.1. Podrezivanje

Podrezivanje, kao centralna tema rada, je proces maskiranja težina u svrhu smanjenja gustoće neuronske mreže. Zadatak podrezivanja u konvolucijskim mrežama je pronaći i ukloniti parametre koji, uz takvu inicijalizaciju parametara, arhitekturu i metodu učenja, su najmanje relevantni za točnost i generalizaciju modela (Slika 4.1). Ova metoda biti će detaljno obrađena u poglavlju 4.

3.2. Kvantizacija

Kvantizacija je tehnika redukcije broja bitova potrebnih za spremanje svake pojedine težine u neuronskoj mreži uz pomoć tehniku dijeljenja težina. Standardno, za pohranu težine, odnosno parametra neuronske mreže koristi se 32-bitni zapis decimalnog broja. Pri kvantizaciji primjenjujemo algoritam grupiranja (eng. clustering) (Slika 3.1). Ako bismo koristili 4 bita za zapis svake od težina, tada bismo imali $2^4 = 16$ centroida u primjerice k-means algoritmu. Svaki od tih 16 centroida zapravo predstavlja neku vrijednost te se ta mapa konstantno ažurira za vrijeme treniranja. Ažuriranje je slično klasičnoj metodi gradijentnog spusta. Računa se parcijalna derivacija svake od težina te se one sumiraju svakoj diskretnoj odgovarajućoj 3-bitnoj težini. Obzirom na gore navedeno, prepostavimo da želimo ažurirati

centroid koji ima vrijednost 4, te su parcijalne derivacije prema težinama tog centroida 0.1, 0.2 i 0.3. Tada ažuriramo vrijednost koju taj centroid predstavlja sa 4 na 4.2.



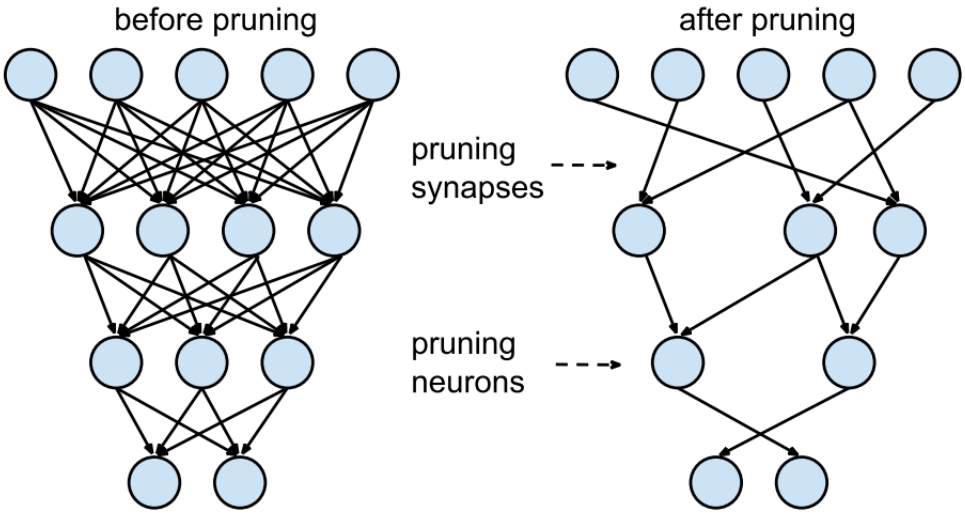
Slika 3.1: Primjer kvantizacije težine uz pomoć algoritma za grupiranje gdje konstantno podešavamo centroide svake od grupa uz pomoć gradijenta težina obzirom na to kojoj grupi ta težina pripada [17].

3.3. Kompresija SVD dekompozicijom

Jedna od efikasnih metoda kompresije potpuno povezanih slojeva dubokih modela je temeljena na primjeni SVD (engl. Single Value Decomposition) dekompozicije na 2D matrice tenzora težina. Dekompozicijom nastanu dvije matrice čiji su retci i stupci sortirani silazno s obzirom na singularnu vrijednost. Retci i stupci koji odgovaraju najmanjim singularnim vrijednostima su uklonjeni što rezultira dvjema matricama smanjenih dimenzija. Ovaj postupak smanjuje broj težina koji se pohranjuje dok rezultat sekvencijalnog množenja vektora takvim reduciranim matricama težina ostaje približan rezultatu množenja originalnom matricom težina. Ova metoda se uspješno primijenjuje u poboljšanju efikasnosti mreža poput Fast R-CNN gdje su uz iznimno nizak pad preciznosti, potpuno povezani slojevi kompresirani do čak 25% svoje originalne veličine [14].

4. Podrezivanje

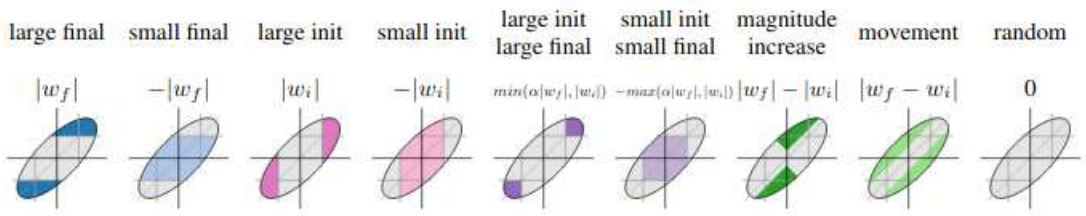
Zadatak podrezivanja u konvolucijskim mrežama je pronaći i ukloniti parametre koji su najmanje relevantni za točnost i generalizaciju modela uz takvu inicijalizaciju parametara, arhitekturu i metodu učenja (Slika 4.1). U sklopu ovog rada uvest ćemo termin *slobodno podrezivanje* koje označuje odabir nestrukturiranog skupa parametara i postavljanje istih na nulu. Takve parametre želimo ukloniti u cilju smanjivanja memorijskog zauzeća, latencije, broja operacija računala te brzine izvođenja. Uz bolje performanse, imamo za cilj održati dobru generalizaciju i točnost modela na neviđenom skupu ulaznih podataka. Obzirom da promatramo konvolucijske modele, ulazni podatci u okviru ovog rada bit će slike (matrice). Danas se većinom duboki modeli izvode na grafičkim jedinicama. Slobodnim podrezivanjem ne bismo imali puno benefita na takvim uređajima obzirom da i dalje množimo iste te matrice. Radni okviri (npr. CuDNN) za izvođenje na grafičkim jedinicama ne znaju to optimizirati. Upravo radi toga, u ovom radu će bit razmatrano i strukturirano podrezivanje kojim umjesto da uklanjamo samo parametre, radimo na uklanjanju ili smanjivanju dimenzija cijelih struktura u konvolucijskim slojevima. U potpuno povezanim slojevima, za optimizaciju na grafičkim karticama, ciljno podrezivanje u vidu optimizacije brzine izvođenja na grafičkim jedinicima je podrezivanje cijelih neurona (Slika 4.1).



Slika 4.1: Prikaz tzv. slobodnog podrezivanja [1]

4.1. Slobodno podrezivanje težina

U pravilu, podrezivanje težina izvodi se računanjem norme (u ovom slučaju apsolutne vrijednosti) težina. Težine se sortiraju te na taj način rangiraju. Najmanjih p% težina se uklanjaju iz mreže (ne sudjeluju u unaprijednom prolazu niti se ažuriraju pri prolasku unatrag, u svrhu istraživanja obično se koristi jednostavna binarna maska). Rangiranje se može obavljati lokalno ili globalno. Kod lokalnog rangiranja također podrezujemo određeni postotak težina, ali sada imamo rangiranje po, primjerice, svakom sloju neuronske mreže, pa ćemo podrezati lokalno najlošije. Kod globalnog uzimamo generalno najmanje težine po apsolutnoj vrijednosti, odnosno normi. Postoji više pristupa za rangiranje, te ih možemo nazvati skup heuristika za rangiranje relevantnosti težina u neuronskoj mreži. Tako u [3] autori koriste upravo gore navedeni pristup globalnog rangiranja. U radu [20] autori se bave upravo različitim pristupima za rangiranje težina, odnosno odabir relevantnih težina te razmatraju zašto je jednostavno podrezivanje težina malih magnituda toliko efikasno te kako se nosi sa raznim kriterijima (Slika 4.2).



Slika 4.2: Slika prikazuje kriterije maskiranja, odnosno funkcije za sortiranje težina prema njihovoj relevantnosti [20].

4.2. Strukturirano podrezivanje

Prilikom optimizacije dubokih modela često podrazumijevamo da je stroj za izvođenje GPU. Programski okviri za duboko učenje (primjerice CuDNN) ne znaju optimizirati ugašene veze, odnosno pojedine težine u matricama težina. Upravo radi toga, potrebno je podrezivati cijele strukture (cijele filtere kod konvolucije ili cijele neurone kod FC slojeva). Glavni benefit takvog pristupa je što se može doći do izravnog utjecaja na brzinu zaključivanja na grafičkim procesorima.

4.2.1. Strukturno podrezivanje evolucijskim algoritmom

Strukturno podrezivanje ovog tipa spomenuto je primjerice u [2]. Da bi odabrali koje filtere je dobro podrezati, u radu se koristi metoda filtriranja čestica (evolucijski algoritam) gdje svaka čestica predstavlja jedan set veza i maski. Na Slici 4.3 iz [2] vidimo primjer smanjivanja matrice konvolucije prema strukuturiranom uzorku (vrijednosti označene sa crvenom bojom) nakon koje imamo redukciju u broju stupaca matrice konvolucija.

Slika 4.3: Primjer metode smanjivanja matrice konvolucije obzirom na strukturirani uzorak (vrijednosti crvene boje) [2]

4.2.2. L1 podrezivanje filtera

Ideja je predložena u [10]. Podrezivanje se obavlja na način da se najmanje relevantni filtri i njihova izlazna mapa značajki pronalaze i uklanjaju. Odabir filtera koji će se ukloniti je jednostavna heuristika kojom se filteri sortiraju prema sumi apsolutnih vrijednosti njihovih težina (L1 norma). Najmanjih *m* filtera (i njihovih izlaza) se uklanja. Treniranje se nastavlja te se postupak ponavlja.

4.2.3. ThiNet metoda podrezivanja filtera

U radu [13] predožen je ThiNet, odnosno programski okvir za kompresiju i akceleraciju dubokih konvolucijskih modela baziran na podrezivanju. Ideja autora za odabir filtera koji će biti podrezani se svodi na pronalazak podskupa filtera koji iz ulaznih vrijednosti sloja (i+1) može aproksimirati izlaz sloja (i+1). Podskup koji dobro aproksimira je novi skup filtera sloja (i+1), dok se ostatak može ukloniti. Pronalazak optimalnog riješenja je NP težak, pa autori koriste pohlepni (eng. greedy) algoritam. Algoritam se temelji na iterativnom isprobavanju izbacivanja filtra ili kanala iz cijelog skupa u skup za podrezivanje. Onaj filter (ili kanal ako se normiraju težine filtra po kanalima) koji je nakon izbacivanja najmanje utjecao na rezultat trajno ostaje u skupu za podrezivanje (Slika 4.4). Nakon podrezivanja, autori sa preostalim dijelom mreže ugađaju parametre (eng. fine-tuning), odnosno nastavljaju trenirati model s ciljem popravljanja generalizacije.

Eq. 6

$$\arg\min_{T} \sum_{i=1}^{m} \left(\sum_{j \in T} \hat{\mathbf{x}}_{i,j} \right)^2$$

$$\text{s.t.} \quad |T| = C \times (1 - r), \quad T \subset \{1, 2, \ldots, C\}.$$

Algorithm 1 A greedy algorithm for minimizing Eq. 6

Input: Training set $\{(\hat{\mathbf{x}}_i, \hat{y}_i)\}$, and compression rate r
Output: The subset of removed channels: T

```
T ← ∅; I ← {1, 2, ..., C};
while |T| < C × (1 − r) do
    min_value ← +∞;
    for each item i ∈ I do
        tmpT ← T ∪ {i};
        compute value from Eq. 6 using tmpT;
        if value < min_value then
            min_value ← value; min_i ← i;
        end if
    end for
    move min_i from I into T;
end while
```

Slika 4.4: Pseudokod pohlepnog algoritma baziranog na iterativnom izbacivanju jednog po jednog filtra. Funkcionira na način da se izbaci filter, izmjeri točnost na testnom skupu te usporedi sa dosad najvećom točnošću. Onaj filter čije je izbacivanje rezultiralo najvećom točnošću biva izbačen te se treniranje nastavlja. Jednadžba 6. predstavlja optimizacijski problem ponalaska filtera koji se smiju ukloniti, gdje *m* označava broj primjera za treniranje $\{(\widehat{\mathbf{x}_i}, \widehat{y_i})\}$, a *r* stopu kompresije.

4.2.4. Podrezivanje filtera geometrijskim medijanom

Rad [6] predlaže metodu podrezivanja koja se temelji na otkrivanju i uklanjanju redundantnih filtera. Takvi se pronalaze uz pomoć geometrijskog medijana $x^* \in \mathbb{R}^d$ koji uz de-

finiran set od n točaka $a^{(1)}, ..., a^{(n)}$, takvih da je $a^{(i)} \in \mathbb{R}^d$, minimizira sumu Euklidskih udaljenosti do tih točaka.

$$x^* = \arg\min_{x \in \mathbb{R}^d} f(x) \quad \text{gdje je} \quad f(x) = \sum_{i \in 1,..,n} ||x - a^{(i)}||_2$$

Geometrijski medijan računa se za sve filtre sloja *i*:

$$x^{GM} = \arg\min_{x \in \mathbb{R}^{N_1 xKxK}} \sum_{j' \in [1, N_{i+1}]} ||x - F_{i,j'}||_2,$$

gdje je N_i ukupan broj filtera dimenzija *KxK* u sloju *i*. Zatim se pronalaze svi filtri tog sloja koji su najbliži geometrijskom medijanu:

$$F_{i,j^*} = \arg\min_{F_{i,j'}} ||F_{i,j'} - x^{GM}||_2, \quad j' \in [1, N_{i+1}]$$

Pronalazak geometrijskog medijana je vremenski zahtjevan problem te se zamijenjuje pristupom pronalaska konkretnog filtra sloja *i* koji minimizira zbroj euklidskih udaljenosti do ostalih filtera u tom sloju:

$$F_{i,x^*} = \arg\min_{x} \sum_{j' \in [1, N_{i+1}]} ||x - F_{i,j'}||_2, \quad x \in F_{i,1}, ..., F_{i,N_{i+1}}$$

Takvi filtri mogu biti predstavljeni drugim filtrima tog sloja. Njihovim micanjem, distribucija težina će se neznatno promijeniti, a performanse mreže prije i poslje njihovog uklanjanja će biti jako slične.

Algorithm 1 Algorithm Description of FPGM

Input: training data: $\mathbf{X}$.
1: **Given**: pruning rate P_i
2: **Initialize**: model parameter $\mathbf{W} = \{\mathbf{W}^{(i)}, 0 \leq i \leq L\}$
3: **for** $epoch = 1; epoch \leq epoch_{max}; epoch++$ **do**
4: Update the model parameter $\mathbf{W}$ based on $\mathbf{X}$
5: **for** $i = 1; i \leq L; i++$ **do**
6: Find $N_{i+1}P_i$ filters that satisfy $\arg\min_x \sum_{j' \in [1, N_{i+1}]} ||x - F_{i,j'}||_2, \ x \in F_{i,1}, ..., F_{i,N_{i+1}}$
7: Zeroize selected filters
8: **end for**
9: **end for**
10: Obtain the compact model $\mathbf{W}^*$ from $\mathbf{W}$
Output: The compact model and its parameters $\mathbf{W}^*$

Slika 4.5: Pseudokod podrezivanja filtra geometrijskim medijanom

4.2.5. Mekano podrezivanje filtera

U prethodnim algoritmima podrezivanja filtra, pravilo je da uklonjeni filtri više ne sudjeluju u daljnem treniranju modela. Ideja mekanog podrezivanja filtra iz rada [5] je nastaviti ažuriranje podrezanih filtra u idućim fazama treniranja. Na ovaj način, mekano podrezivanje

omogućuje da tijekom treniranja podrezani model ima isti kapacitet kao i originalan model. Za razliku od običnog podrezivanja, broj mapa značajki se ne smanjuje te performanse modela ostaju iste.

Kriterij odabira filtra koji će biti podrezani je l_p norma, a računa se pomoću

$$||F_{i,j}||_p = \sqrt[p]{\sum_{n=1}^{N_i} \sum_{k_1=1}^{K} \sum_{k_2=1}^{K} |F_{i,j}(n, k_1, k_2)|^p},$$

gdje je $F_{i,j} \in \mathbb{R}^{N_i x K x K}$ j-ti filter u i-tom sloju mreže, N_i broj kanala, a K dimenzija filtra. Konvolucija filtrima koji imaju manju l_p normu rezultira relativno niskim aktivacijama i time manje utječe na konačnu predikciju dubokih konvolucijskih modela. Filtri s najnižom l_p normom postavljaju se na nulu.

N_i predstavlja broj ulaznih kanala tj. ukupan broj filtera u konvolucijskog sloja i. Koristi se empirijski određena stopa podrezivanja P_i kako bi se izabralo $N_{i+1}P_i$ filtera koji će se podrezati u i-tom sloju. Stopa je ista za svaki sloj mreže. Vrijednosti svih $N_{i+1}P_i$ filtera se postavljaju na 0, ali se i dalje ažuriraju u narednim fazama treniranja. Podrezivanje u svim slojevima se izvršava paralelno što pridonosi smanjenju vremena izvršavanja za razliku od pohlepnog podrezivanja koje se izvodi sloj po sloj, te je potrebno čekati da mreža konvergira nakon podrezivanja svakog sloja.

Mreža se zatim trenira kako bi se podrezani filtri čija je vrijednost 0 mogli rekonstruirati, tj. poprimiti vrijednosti različite od nule zahvaljući propagaciji unazad. Međutim, nakon što je model konvergirao, podrezani filtri i mape značajki koje nastaju konvolucijom tih filtra se i dalje promatraju kao da imaju vrijednosti 0. Dakle, za stopu podrezivanja P_i, u i-tom sloju, samo $N_{i+1}(1 - P_i)$ imaju utjecaj na završnu predikciju.

Algorithm 1 Algorithm Description of SFP

Input: training data: $\mathbf{X}$, pruning rate: P_i
the model with parameters $\mathbf{W} = \{\mathbf{W}^{(i)}, 0 \leq i \leq L\}$.
Initialize the model parameter $\mathbf{W}$
for $epoch = 1; epoch \leq epoch_{max}; epoch++$ **do**
  Update the model parameter $\mathbf{W}$ based on $\mathbf{X}$
  for $i = 1; i \leq L; i++$ **do**
    Calculate the ℓ_2-norm for each filter $\|\mathcal{F}_{i,j}\|_2, 1 \leq j \leq N_{i+1}$
    Zeroize $N_{i+1}P_i$ filters by ℓ_2-norm filter selection
  end for
end for
Obtain the compact model with parameters $\mathbf{W}^*$ from $\mathbf{W}$
Output: The compact model and its parameters $\mathbf{W}^*$

Slika 4.6: Pseudokod mekanog podrezivanja iz rada [5]

4.3. Programske metode implementacije podrezivanja

U okviru ovog poglavlja govorili smo puno o algoritmima i metodama podrezivanja. Programski pristupi ostvarivanja istih se razlikuju, pa tako govorimo o dva načina podrezivanja:

1. Blago podrezivanje - ovakav oblik podrezivanja najčešće je korišten. To je pristup u kojem ne mijenjamo fizičku arhitekturu modela, nego samo ugasimo težine (najčešće maskiranjem).

2. Grubo podrezivanje - podrezivanje u kojem trajno uklanjamo težine (obično cijele strukture) fizički iz definicije, odnosno arhitekture modela.

Obzirom da je prvonavedeni pristup prilično trivijalan, preskočit ćemo opis i odmah krenuti na metodu grubog podrezivanja. Kao što se da naslutiti, meko podrezivanje koristi se pretežno u istaživačke i eksperimentalne svrhe, dok pristup grubog podrezivanja bismo koristili kada želimo iskoristiti benefite podrezivanja u realnoj primjeni. Naime, pristup grubog podrezivanja može biti veoma koristan i za treniranje modela kao što to pokazuju eksperimenti u 6.2. Programski ostvarenje svih eksperimenata ovog rada je unutar programskog okvira PyTorch. Tamo grubo podrezivanje još uvijek nije podržano, pa je potrebno nešto dodatnog truda za ostvarenje istog. Postoje alternativni alati [21] [19] za ostvarenje ovakvog pristupa. Oba rada imaju vrlo sličan pristup. Naime, strogo definicijski, grubo podrezivanje je poces transformiranja strukture mreže prema unaprijed određenoj skripti pravila. Da bi takav proces bio moguć, potrebno je implementirati metodu ažuriranja svih dijelova mreže na koje promjena bilo koje od struktura mreže utječe. U [8], autori grubo podrezivanje zapravo zovu "Neuronski Sakupljač Smeća" (eng. Neural Garbage Collection). Ovdje se može povući paralela sa mehanizmom sakupljača smeća u programiranju gdje je cilj riješiti se objekata koji nisu referencirani, tj. oni koji se ne koriste. Isto tako se neuronski sakupljač smeća nastoji riješiti neurona koji nemaju utjecaj na predikcije mreže za vrijeme treniranja. U PyTorch okruženju, ovaj problem je iznimno kompleksan jer su strukture (slojevi, povezanost slojeva, veličine slojeva) definirane u kodu. Za usporedbu, Caffe programski okvir koristi format Protobuf kojim je prilično jednostavno baratati unutar koda. Kompleksan dio dolazi kada primjerice želimo ukloniti filter iz konvolucije praćene slojem normalizacije BN (eng. batch normalization), tada i BN sloj zahtjeva promjene u konfiguraciji (veličina ulaznog vektora i parametri BN sloja). Takve ovisnosti mogu postati vrlo kompleksne, pogotovo ako dodamo da na ove promjene moramo promjeniti i tenzor koji optimizator ima te pripadajuće gradijente da bismo mogli nastaviti učiti model. Obzirom da PyTorch ne stvara graf izvođenja sve do trenutka prolaska unaprijed (poziv funkcije "forward"), cijela priča se dodatno komplicira. Pristup je kreirati graf izvođenja, te na svaki zahtjev promjene neke od struktura, obilaskom grafa utvrditi koje su sve modifikacije potrebna na prethodnicima i

pratiteljima sloja koji je modificiran prema bazi pravila. Upravo radi ove kompleksnosti, oba navedena okvira nude promjene jedino nad konvolucijskim i potpuno povezanim slojem.

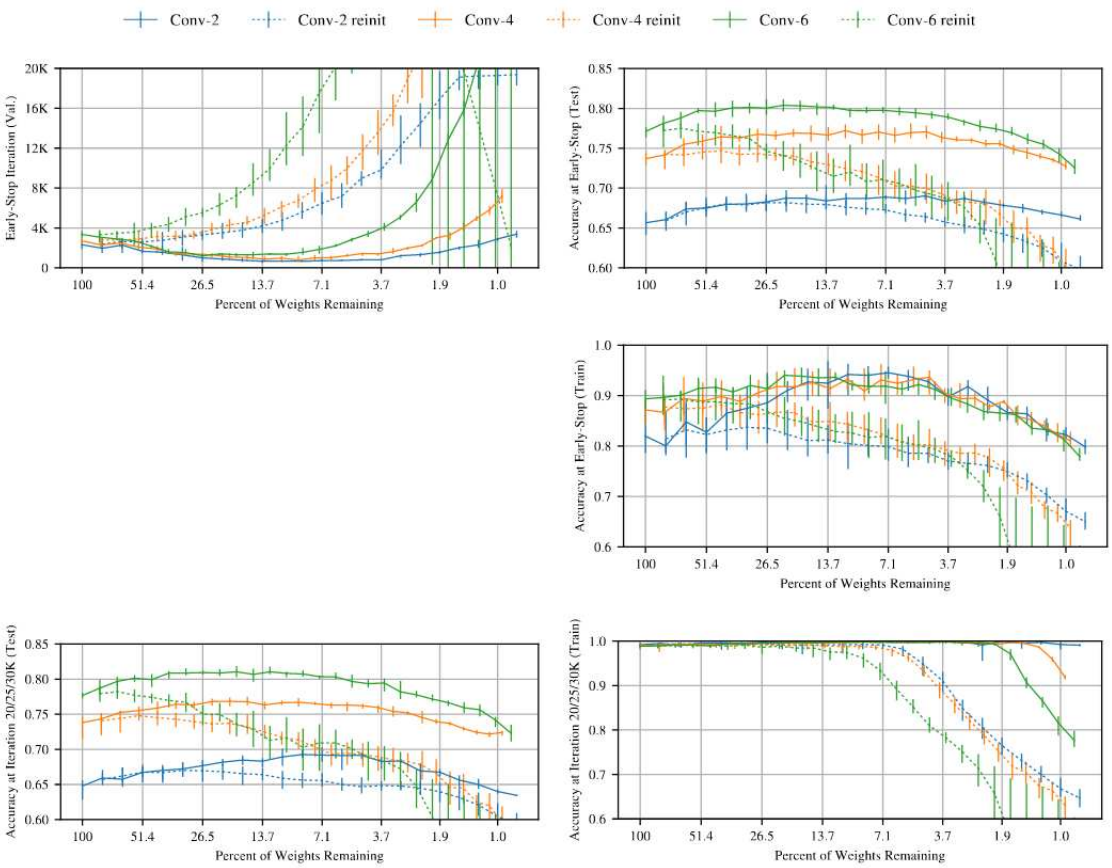
5. Hipoteza dobitnog listića lutrije

Do prije [3] pokazivalo se da je nakon podrezivanja teško ponovno trenirati dobiveni model. Rad [3] pokazuje da dosadašnje tehnike podrezivanja prirodno otkrivaju podmrežu čija ih inicijalizacija čini sposobnim za efikasno treniranje. Hipoteza glasi: "Gusta, slučajno inicijalizirana, unaprijedna mreža sadrži podmreže (dobitne listiće) koje, kada su trenirane u izolaciji, postižu točnost usporedivu sa originalnom mrežom u sličnom broju iteracija.".

Identifikacija redudantnih parametara u okviru [3] je temeljena na iterativnom podrezivanju onih parametara sa najmanjim apsolutnim iznosom. Preostale težine čine arhitekturu dobitnog listića. Ono što ovaj radi čini jedinstvenim je da se preostali parametri reinicijaliziraju na iste one prvotne vrijednosti koje su imali prije treniranja. Postupak glasi:

1. Slučajno inicijaliziraj neuronsku mrežu sa parametrima θ
2. Treniraj mrežu j iteracija sve do parametara θ_j
3. Podreži p% parametara iz θ_j kreirajući masku m
4. Resetiraj preostale parametre na vrijednosti iz θ, te na taj način kreiraj dobitni listić $f(x, m \odot \theta)$
5. Ponavljaj točku 2

Ponovna inicijalizacija parametara podmreže nakon podrezivanja ne pokazuje se stabilnom za razliku od inicijalizacije sa starim parametrima koja u pravilu pokazuje stabilnost jednako rano kao i originalna mreža. Interesantno je za primjetiti da autori ne pronalaze značajnu razliku između reinicijalizacije parametara nakon podrezivanja i slučajno odabrane podmreže sa istim postotcima preostalih parametara (Slika 5.1). Na MNIST skupu podataka uz potpuno povezanu LeNet [9] arhitekturu ova tvrdnja eksperimentalno ne vrijedi. Predloženo objašnjenje za to je da potpuno povezane mreže vide benefit samo od određenih dijelova slike, odnosno neki parametri i dijelovi slike će biti vrijedniji.



Slika 5.1: Prikaz reinicijalizacije u odnosu na nasumičnu gustoću.

U [20] autori rade na dekompoziciji ideje dobitnog listića lutrije. Ideja se može svesti na tri glavne tvrdnje:

1. Podrezati parametre na neku konstatnu vrijednost nije isto što i postaviti ih na 0. Postavljanje na nulu je ključno kod "gašenja" parametara.

2. Za efikasno treniranje podrezane mreže dovoljno je ponovno inicijalizirati parametre i dodjeliti predznak koji su imali prije podrezivanja.

3. Kriterij za postavljanje težine na nulu (maskiranje) je taj da se težina kreće prema nuli. Time se parametri efektivno postavljaju u vrijednost u koju konvergiraju učenjem. Zbog toga se može se reći da se maskiranje ponaša slično kao treniranje.

Konačno autori pokazuju da postoji supermaska koja, kada se primjeni na slučajno inicijalizirani netrenirani model, daje točnost od 86% na MNISTU sa oko 15% parametara i točnost od 41% na CIFAR10 skupu podataka za učenje sa oko 10% parametara. Oba rezultata su u slučaju podrezivanja sa očuvanjem predznaka, bez treniranja mreže. Također, puno je različitih kriterija za odabir maske predloženo, te se empirijski pokazuje da dva kriterija među njima daju najbolju predikciju irelevantnih parametara. To su razlika u magnitudi između

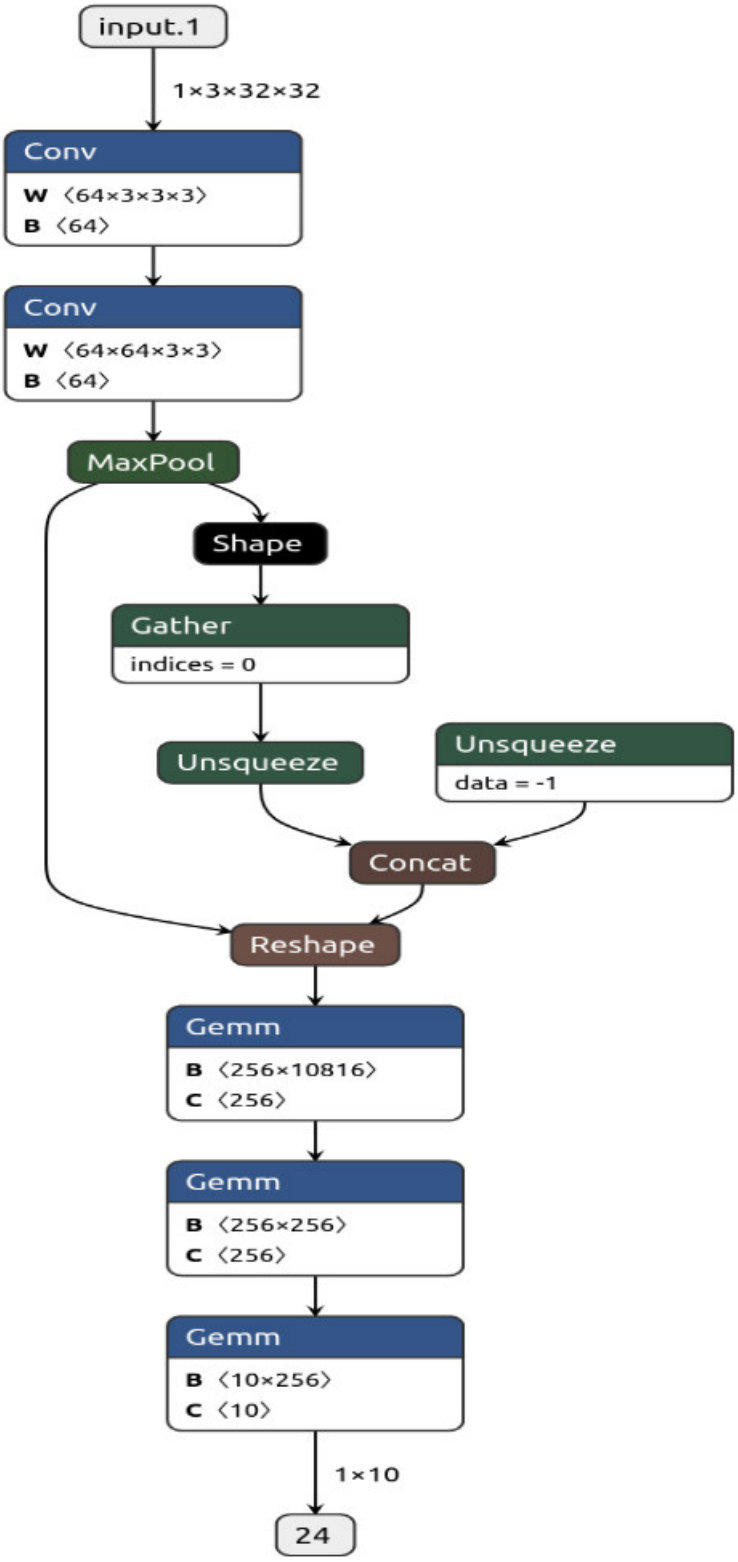
vrijednosti parametra nakon treniranja i parametra za vrijeme inicijalizacije ($|w_f| - |w_i|$) i apsolutna vrijednost konačne vrijednost parametra nakon treniranja ($|w_f|$).

6. Rezultati

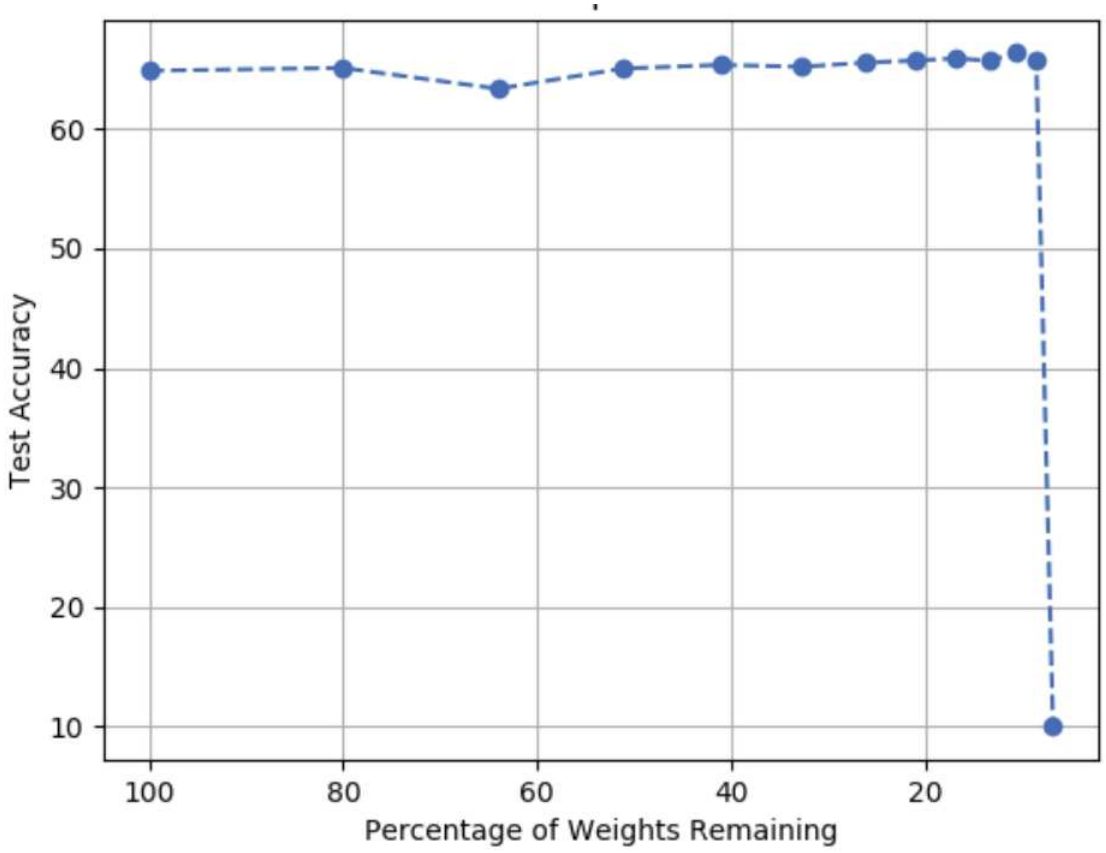
Skup za učenje CIFAR10 sadrži 60000 označenih slika rezolucije 3x32x32. Skup je podijeljen na skup za učenje od 50000 te skup za testiranje od 10000 slika. Oznake sadrže 10 mogućih klasa: avion, automobil, ptica, mačka, jelen, pas, žaba, konj, broj i kamion.

6.1. Iterativno podrezivanje težina

Prvi eksperiment svodi se na iterativno podrezivanje jednostavne konvolucijske mreže nad CIFAR10 skupom za učenje. Slika 6.1 prikazuje arhitekturu mreže conv2. Mreža se sastoji od danas već standardne arhitekture za rješavanje ovakvog tipa problema klasifikacije. Najprije dobivamo mapu značajki kroz dva konvolucijska sloja 3x3 nakon kojih imamo receptivno polje 5x5 pixela kroz svaki od 3 kanala. Nakon toga, u svrhu bolje generalizacije (manje prenaučenosti), manjeg broja parametara te invarijantnosti na translaciju dodajemo sloj sažimanja maksimumom veličine kernela 3 te koraka 2. Konačno, nakon poravnavanja na jednu dimenziju, dolazimo do potpuno povezanih slojeva nakon kojih dobivamo probabilistički izlaz za svaku od 10 mogućih klasa iz CIFAR10.



Slika 6.1: Arhitektura mreže conv2.

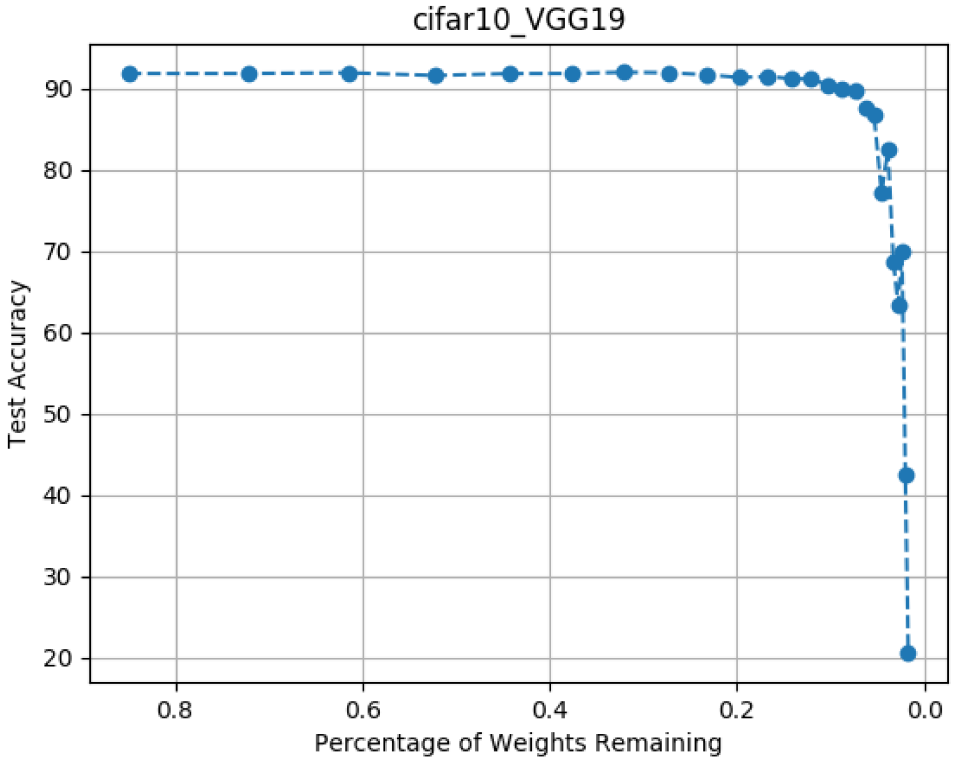


Slika 6.2: Rezultat podrezivanja težina mreže conv2 na skupu za učenje CIFAR10.

Cilj eksperimenta (Slika 6.2) je reprodukcija rezultata iz [3] pomoću jednostavnog iterativnog podrezivanja. Generalno, ideja je trenirati mrežu 40 epoha, uzeti mrežu sa najboljom točnošću nad skupom za testiranje te na temelju nje podrezati p=20% težina. Nakon toga, postavljamo parametre na inicijalne vrijednosti te maskom ugasimo podrezane težine. Takvu novodobivenu mrežu koja sadrži približno 80% parametara treniramo te ponavljamo podrezivanje nad ostatkom parametara, te dobivamo novu mrežu sa oko 80%*0.8=64% parametara. Podrezivanje radimo na način da globalno sortiramo parametre prema njihovoj apsolutnoj vrijednosti, te prođemo kroz slojeve i maskiramo svaki onaj parametar koji je ispod granične vrijednosti. Razlog zašto nije podrezano točno 20% parametara leži u tome da za zadnji potpuno povezani sloj podrezujemo one težine koje su manje od polovice granične vrijednosti. Granična vrijednost je vrijednost onog parametara iz skupa nepodrezanih parametara koji ima najmanju apsolutnu vrijednost.

Mreža conv2 u sa slike 6.2 je trenirana kroz 13 iteracija po 33 epoha. Nakon svake iteracije se radi podrezivanje opisano u prošlom odjeljku. Stopa učenja treniranja je 2e-4 te stopu propadanja (eng. weight decay) od 1e-3. Metoda optimizacije je ADAM, te gubitak unakrsna entropija. U svrhu treniranja korišten je samo skup za treniranje iz CIFAR10 koji je podijeljen na 49000 slika za treniranje te 1000 za validaciju. Iz grafa na slici 6.2 vidimo da najveću točnost na skupu za validaciju postižemo sa samo 12% parametara.

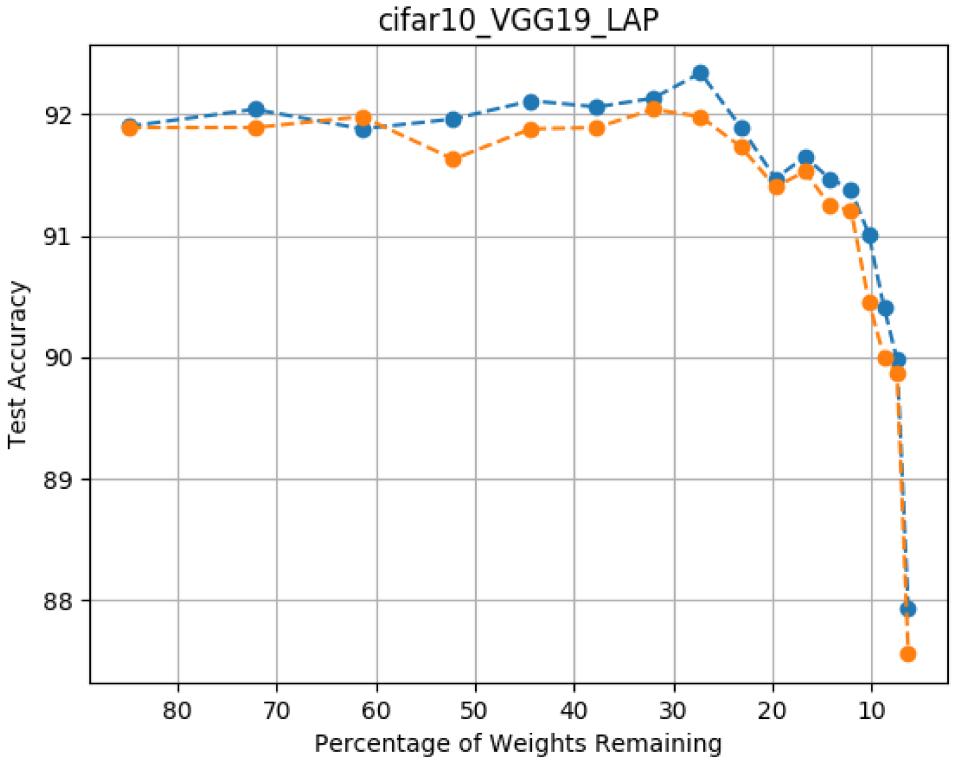
Idući eksperiment je sličan prethodnom, uz nekoliko promjena. Ovaj put je korištena mreža VGG-19 iz [18] sa slike 2.3, uz promjene u dimenzijama struktura u slojevima na način da odgovaraju dimenzijama slika iz skupa CIFAR10. Prvo treniranje kreće nakon podrezivanja (model je već predtreniran na CIFAR10 te ima točnost na skupu za testiranje od 92%). Iz slike 6.3 vidimo da rijedak model sa čak i oko 15% preostalih parametara ima performanse usporedive s modelom punog kapaciteta. Možemo reći da rezultati približno odgovaraju onima iz prethodnog eksperimenta.



Slika 6.3: Rezultat podrezivanja težina mreže VGG19 na CIFAR10 skupu za učenje.

Dodatno, usporedit ćemo još i dvije različite metode podrezivanja težina. Ideja je nešto pametnijom heuristikom od podrezivanja magnitudom MP (eng. magnitude pruning) dobiti nešto bolje rezultate i pokazati da postoji bolja alternativa MP-u. U tu svrhu koristit ćemo tehniku podrezivanja LAP (eng. lookahead pruning) iz [15] . Kod LAP-a, minimiziramo mjeru distorzije koja računa utjecaj podrezane težine na susjedne slojeve.

Na slici 6.4 vidimo da za ovu arhitekturu, metodu treniranja te parametre treniranja i podrezivanja LAP metoda pokazuje nešto bolje performanse, odnosno podrezivanjem dobivamo nešto bolji model. Bitan zaključak ovog eksperimenta je da je performansa podrezivanja magnitudom, iako u okviru eksperimenta nešto lošija, i dalje usporediva sa kompleksnijim heuristikama odabira manje relevantnih parametara koji će biti podrezani.



Slika 6.4: Usporedba dviju metoda podrezivanja težina modela VGG-19 na skupu CIFAR10. Plavom bojom označena je LAP metoda, dok narančastom metoda MP.

6.2. Strukturno iterativno podrezivanje

Kao što je ranije spomenuto, bitna stavka optimizacije u vidu podrezivanja je i korisnost u smislu brzine izvođenja nakon integracije na ciljnoj platformi (obično GPU). U okviru ovog dijela eksperimenata bavit ćemo se analizom podrezivanja filtera u konvolucijskim slojevima. Interesantna informacija bila bi i fizički ukloniti podrezane filtere ili kanale za vrijeme treniranja mreže. Na takav način bismo eliminirali potrebu za maskom te imali fizički manju mrežu za koju bismo lagano mogli izmjeriti performanse u vidu brzine izvođenja.

6.2.1. Podrezivanje podešavanjem težina

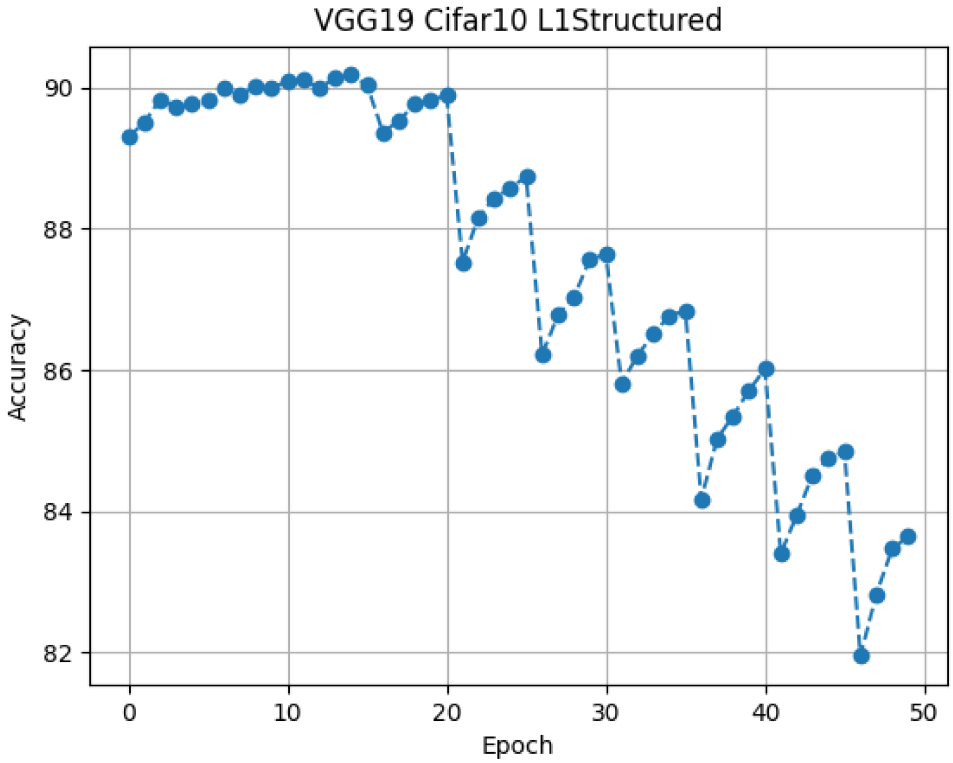
Za razliku od prethodnih, ovaj eksperiment uključit će cijeli proces optimizacije. Obzirom da su prethodni eksperimenti bili nad VGG-19 arhitekturom, za lakše uspoređivanje rezultata podrezivanja težina i struktura i za ovaj eksperiment koristit će se ista ta arhitektura mreže. Kofiguracija treniranja ovog eksperimenta je:

- Funkcija gubitka: unakrsna entropija
- Metoda optimizacije: stohastički gradijentni spust

- Stopa učenja = [1e-1, 1e-2, 1e-3, 1e-3, 1e-5]
- Momentum = 0.9
- Faktor propadanja težina = 1e-4
- Podjela skupa za učenje: 10000 slika za validaciju, 50000 slika za treniranje

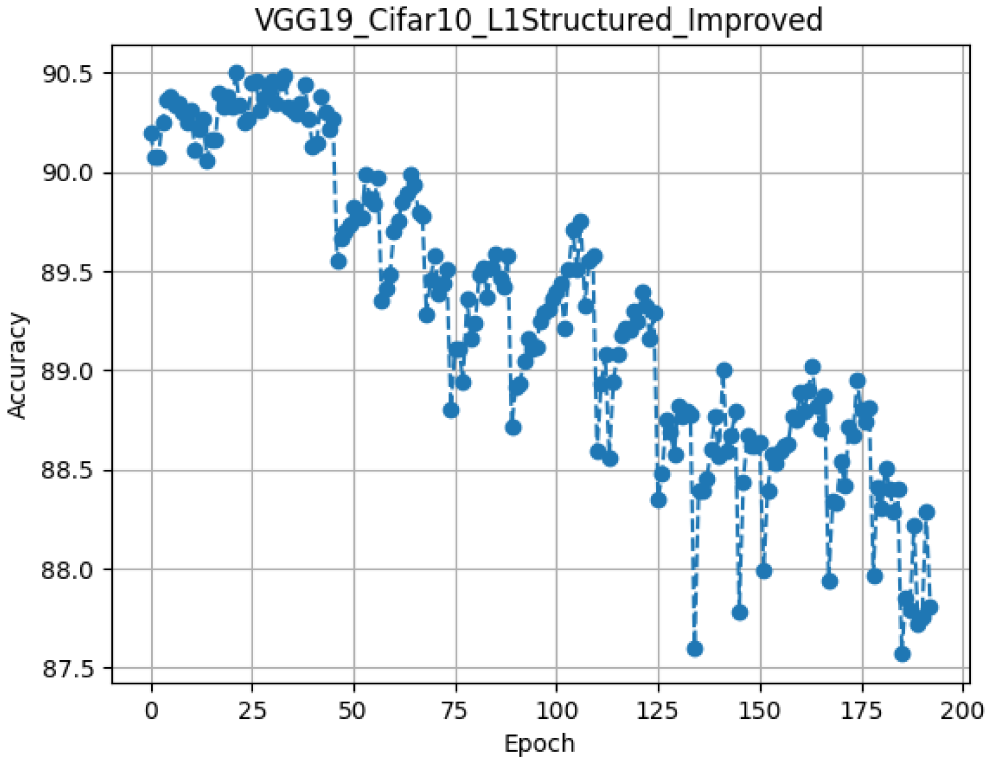
Podrezivanju prethodi 40 epoha treniranja modela bez podrezivanja sa gore navedenim stopama učenja. Točnost na skupu za validaciju je 89.63%. U okviru prvog eksperimenta strukturnog podrezivanja izvedeno je 50 epoha treniranja uz stopu učenja 1e-5. Nakon svakih 5 epoha obavljeno je lokalno podrezivanje filtera konvolucijskih slojeva. Nakon podrezivanja nastavljamo trenirati mrežu sa postojećim parametrima (eng. fine tuning). Može se primjetiti da u ovom eksperimentu ne koristimo hipotezu dobitnog listića lutrije. Također, koristi se lokalno podrezivanje (p% filtera sa najmanjom normom u svakom sloju posebno se uklanja).

Kao što se vidi iz grafa na slici 6.5, model jako dobro generalizira i održava točnost sve do 43% kapaciteta (epoha 15). Nakon svakog idućeg podrezivanja vidimo značajan pad u točnosti i nedovoljno vremena za oporavak. Upravo radi toga, pokrenut ćemo dodatan eksperiment, no ovaj puta ćemo podrezivati tek kada 3 epohe zaredom nismo dobili poboljšanje u točnosti modela. Isto tako, dosad smo podrezivali 10% težina u svakoj iteraciji podrezivanja. Ovog puta ćemo, obzirom da smo vidjeli značajne padove u točnosti kod kasnijih iteracija, podrezivati 5% parametara lokalno po iteraciji.



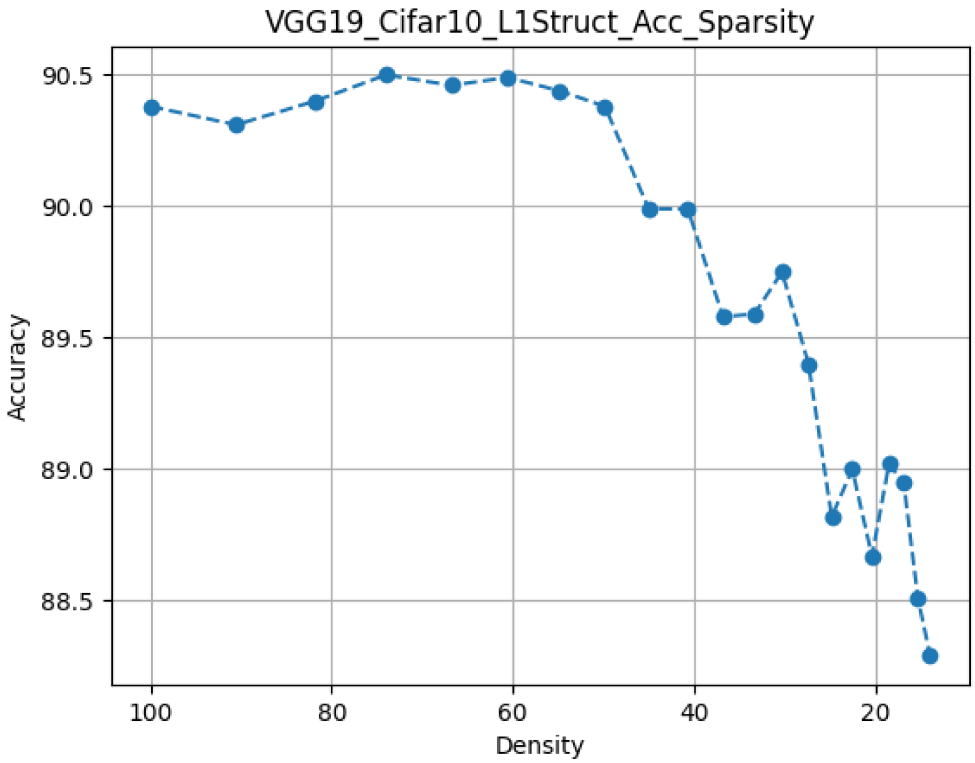
Slika 6.5: Rezultati lokalnog podrezivanja filtera konvolucije mreže VGG19 bazirano na L1 normi i podešavanju težina na skupu CIFAR10.

Iz grafa na slici 6.6 vidimo prilično bolje rezultate učenja i na visokom i na niskom kapacitetu mreže, odnosno čekanje konvergencije se isplatilo u generalnom slučaju. Na grafu 6.7 vidimo nešto interesantniji prikaz ovisnosti točnosti obzirom na kapacitet mreže. Vidimo da mreža na 50% kapaciteta ima bolju točnost na validacijskom skupu od mreže sa 100% kapaciteta. Ako pogledamo tablicu 6.1, vidimo da smo isto tako poboljšali performanse u vidu brzine evaluacije na CPU i GPU uređajima te osjetno smanjili broj operacija, a time i potrošnju snage uređaja.

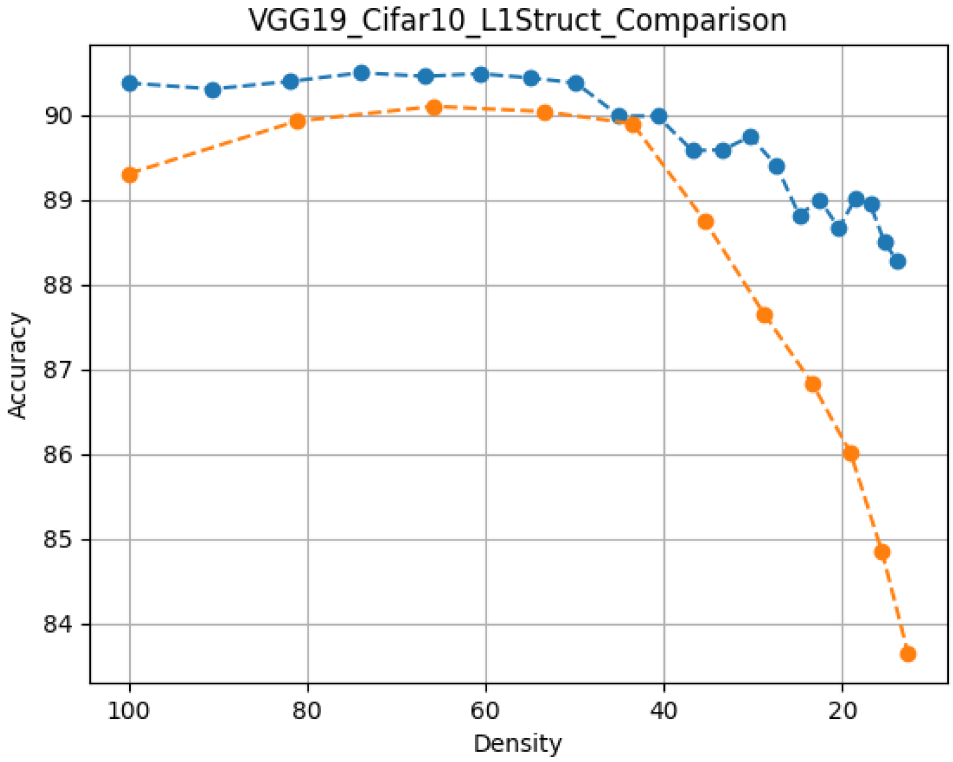


Slika 6.6: Rezultati lokalnog podrezivanja filtera konvolucije mreže VGG19 bazirano na L1 normi uz podešavanje težina te uz čekanje konvergencije od 3 epohe na skupu CIFAR10. Prikazana je točnost na skupu za testiranje.

Interesantno je pogledati usporedbu prethodna dva eksperimenta u ovisnosti o gustoći (kapacitetu) mreže. Na grafu sa slike 6.8 vidi se usporedba treniranja fiksnih 5 epoha i treniranja kada 3 epohe čekamo konvergenciju. No, iako smo u drugom eksperimentu krenuli sa boljim modelom, trenirali oko 200 epoha u odnosu na 50 iz prvog eksperimenta i podrezivali 5% filtera po sloju u odnosu na 10% iz prvog eksperimenta, i dalje vidimo da na gustoći od oko 42.5% obje metode daju istu točnost na validacijskom skupu uz odstupanje od 0.05%. Vidimo da u oba eksperimenta točnost na validacijskom skupu je održana ili raste sve do kapaciteta modela u rasponu od 40-50%.



Slika 6.7: Graf prikazuje točnost na skupu za testiranje u odnosu na gustoću mreže kod eksperimenta na slici 6.6.



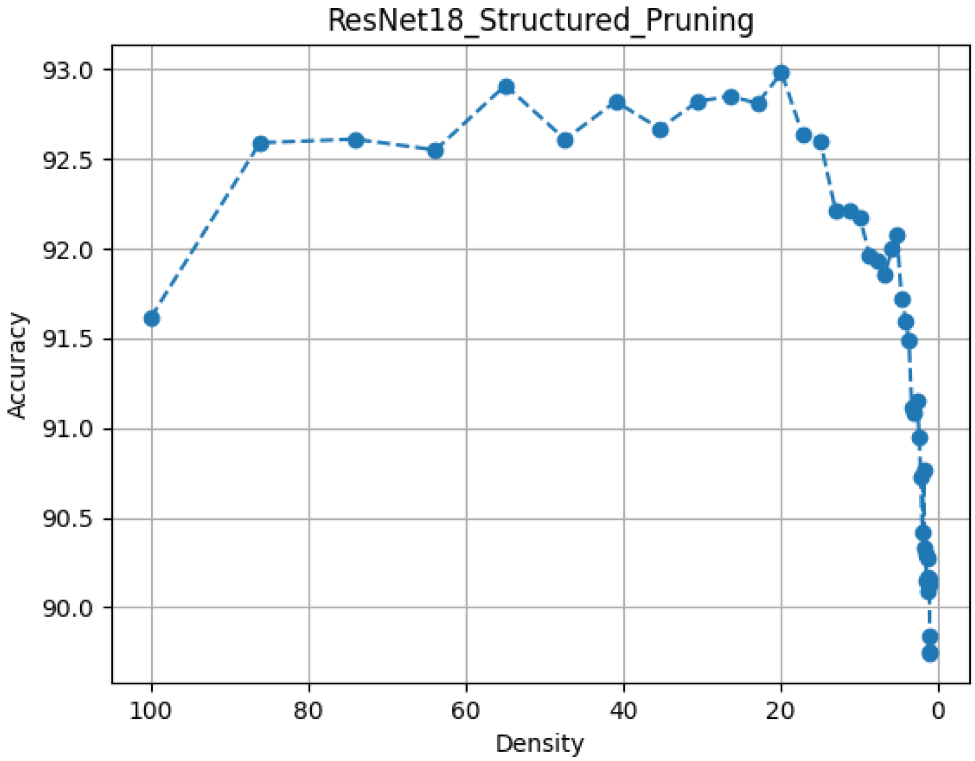
Slika 6.8: Na grafu je plavom bojom označen eksperiment 6.7 te narančastom točnosti iz eksperimenta sa slike 6.5.

U slučaju strukturnog podrezivanja fizički se uklanjaju filtri umjesto njehovog maskiranja. Takav pristup omogućuje usporedbe performansi modela u ovisnosti o brzini treniranja te brzini evaluacije modela nakon i prije podrezivanja. Kao što možemo vidjeti iz Tablice 6.1, podrezivanjem dobivamo poboljšanja. Vrijeme evaluacije jedne slike modela kapaciteta 100% traje 21.56 ms, dok to vrijeme iznosi 12.18 ms kada je model na kapacitetu od 43.44% na CPU. Zaključak je da smo reducirali vrijeme potrebno za obradu jedne slike za faktor 2 i to bez gubitka točnosti na validacijskom skupu. Ako napravimo istu tu usporedbu na GPU uređaju, brojke su nešto manje impresivne. Valjan razlog tome bi bio da moderni GPU-ovi jako dobro paraleliziraju svoje izvođenje te imaju vrlo pametno upravljanje resursima te na taj način, obzirom da je dubina mreže nepromjenjena, ne vidimo promjene u brzini izvođenja. Ono što je bitno napomenuti je da podrezani model ima puno manji broj operacija, pa samim time je za očekivati da će uređaj imati manju potrošnju snage.

Density	CPU time [ms]	GPU time [ms/epoha]	GFLOPs	Acc
100%	21.56	6.25	0.40	89.308%
81.14%	18.71	4.67	0.33	89.829%
65.78%	16.45	4.92	0.26	90.104%
53.48%	14.11	4.91	0.22	90.043%
43.48%	12.18	4.71	0.18	89.900%
35.37%	10.73	5.06	0.15	88.747%
28.75%	11.42	4.91	0.12	87.654%
23.40%	9.59	5.06	0.10	86.839%
19.07%	8.79	4.74	0.08	86.013%
15.52%	9.09	4.69	0.06	84.840%
12.60%	8.61	4.48	0.05	83.636%

Tablica 6.1: Tablica prikazuje razne omjere točnosti, brzine evaluacije te gustoće mreže. Posebno je interesantna gustoća mreže od 43.48% kada imamo oko 40% ubrzanje na CPU uređajima te značajno smanjenje broja operacija uz bolju točnost nego na punom kapacitetu.

Obzirom na interesantne rezultate sa klasičnom VGG-19 arhitekturom, bilo bi interesantno vidjet kakve performanse dobivamo ako iskoristimo neke od modernijih tehnika dubokog učenje kao što su preskočne veze i normalizacija. Teorijski se pretpostavlja da preskočne veze "zaglađuju" funkciju gubitka [11], dok normalizacija nad grupom (eng. batch normalization) pomaže stabilnosti i zaglađivanju funkcije gubitka [16] te potencijalno i reducira unutarnji kovarijantni pomak [7]. Upravo iz tih razloga, ovaj puta ćemo pokrenuti eksperiment sa ResNet-18 arhitekturom te CIFAR-10 skupom za učenje. Nakon nasumične inicijalizacije mrežu treniramo 100 epoha uz pad stope učenja na 10% svoje vrijednosti svakih 30 epoha. Nakon tog imamo lokalno podrezivanje na način da u svakom sloju konvolucije uklonimo 5% filtera sa najmanjom L1 normom, resetiramo stopu učenja te pokrećemo novo treniranje, odnosno podešavanje težina idućih 50 epoha. Stopu učenja sada skaliramo svakih 20 epoha, te nakon podrezivanja opet resetiramo.



Slika 6.9: Na grafu je prikazan eksperiment podrezivanja ResNet-18 nad skupom CIFAR10. Za kriterij podrezivanja filtera koristi se L1 norma. Podrezuje se

Na grafu 6.9 vidimo da mreže kapaciteta 100% i 4% imaju jednaku točnost na skupu za testiranje koja iznosi 91.6%. Odnosno, ako podrežemo 96% mreže uspjevamo zadržati generalizaciju mreže punog kapaciteta.

Također, interesantan dodatak ovom eksperimentu jesu vremena potrebna za evaluaciju slika, odnosno pitamo se jesmo li fizičkim podrezivanjem zaista uštedjeli vrijeme izvođenja na GPU.

Density	CPU [ms]	batch=1, GPU [ms/epoha]	batch=256, GPU [ms/epoha]
100.00 %	14.57	5.95	32.12
86.07 %	15.17	5.99	30.56
73.99 %	14.27	5.97	28.17
63.77 %	12.76	5.97	24.95
54.90 %	12.61	5.99	23.41
47.31 %	11.72	6.00	21.74
40.87 %	11.17	6.00	19.52
35.31 %	10.79	6.01	18.15
30.44 %	10.43	6.01	16.02
26.40 %	9.98	6.01	14.63
22.86 %	9.63	6.01	13.66
19.82 %	9.51	6.00	12.89
17.17 %	9.60	6.12	12.32
14.88 %	8.84	6.01	11.46
12.94 %	8.81	6.00	10.84
11.24 %	8.91	6.03	9.54
9.83 %	8.85	5.98	9.16
8.61 %	8.69	6.23	8.32
7.60 %	8.32	6.05	7.66
6.65 %	8.71	6.04	7.37
5.83 %	8.53	6.00	7.07
5.15 %	8.56	6.24	6.85
4.55 %	8.19	6.21	6.52
4.04 %	8.42	6.04	6.35
3.63 %	8.24	6.19	6.21
3.26 %	8.11	6.22	6.22
2.91 %	8.16	6.13	6.18
2.59 %	8.28	6.06	6.28
2.33 %	8.00	6.10	6.11

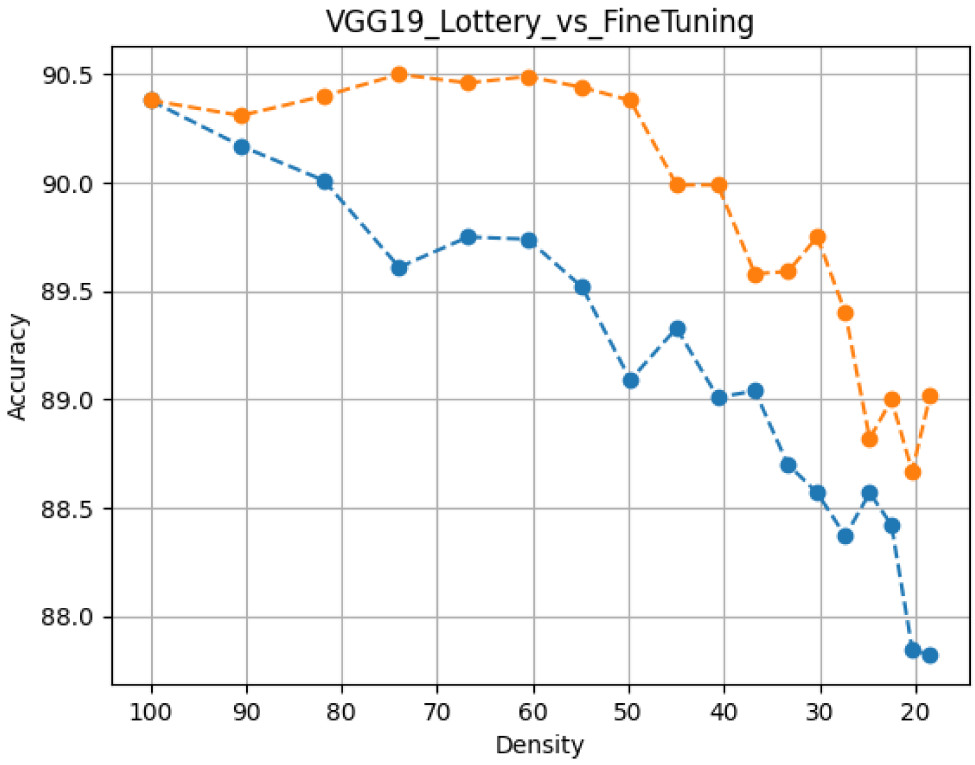
Tablica 6.2: Tablica prikazuje brzine evaluacije u ovisnosti o gustoći mreže ResNet-18 učenu nad skupom CIFAR10.

Na tablici 6.2 vidimo da, uz ulazni skup od 256 slika, dobivamo značajna ubrzanja u evaluaciji na GPU. Tako mreža kapaciteta 4% postiže točnost od 91.6% te brzinu od 157 FPS, mreža kapaciteta 19.82% postiže točnost od 92.98% te brzinu od 77 FPS, dok mreža

kapaciteta 100% postiže točnost od 91.61% te brzinu evaluacije 31 FPS.

6.2.2. Podrezivanje hipotezom dobitnog listića lutrije

U prethodnom potpoglavlju trenirali smo mrežu blizu konvergencije, uklonili dio filtera lokalnim podrezivanjem baziranim na L1 normi, te nastavili trenirati mrežu. Ovaj puta ćemo resetirati cijelu mrežu nakon podrezivanja. To znači da, nakon što napravimo početnu nasumičnu (eng. random) inicijalizaciju težina, takav model spremamo za svaku novu iteraciju testa hipoteze. U okviru eksperimenta takav model treniramo 100 epoha, nakon čega sortiramo filtre lokalno prema iznosu njihove L1 norme te zapamtimo koji su to filtri, odnosno spremimo "loše" filtre u listu filtera za podrezati. Nakon toga ponovno učitamo model, te sve filtre iz liste filtera za podrezati uklanjamo i opet treniramo model 100 epoha, nakon čega nadopunjujemo listu novim "lošim" filterima te ponovno učitavamo inicijalni model i uklanjamo filtre iz liste. Svi parametri treniranja su isti kao i u prethodnim eksperimentima, uz to da je faktor propadanja stope učenja jednak 2 uz početnu stopu učenja od lr=0.001.



Slika 6.10: Narančasta linija predstavlja točnost mrežu VGG-19 učenu nad skupom CIFAR-10 kroz različite gustoće podrezivanja L1 normom(eksperiment 6.6). Plavom linijom je označen eksperiment dobitnog listića lutrije strukturnim podrezivanjem nad VGG-19 arhitekturom.

Na grafu 6.10 vidimo usporedbu podrezivanja metodom dobitnog listića lutrije i meto-

dom podešavanja (eng. fine-tuning) težina. Vrlo je interesantno da za MP strukturno podrezivanje filtera za VGG-19 arhitekturu hipoteza listića lutrije ne pokazuje zavidne rezultate. Kao što je to spomenuto u [20], čini se da hipotezom dobitnog listića zapravo pronalazimo težine koje su ionako krenule u nulu. Podrezivanjem takvih težina zapravo samo ubrzamo proces njene konvergencije u nulu. Kod filtera očigledno uklanjamo neke od težina unutar podrezanog filtera koje su bile relevantne te nisu bile usmjerene prema nuli. Upravo radi toga ne podrezujemo nužno na optimalan način. To bi rezultiralo zaključkom da, podrežemo li nasumične filtre, potencijalno dobivamo točnost usporedivu onoj kada podrezujemo filtre s najmanjom L1 normom. Eksperimentalno se pokazuje da podrezivanje nasumičnih filtera daje lošije rezultate nego podrezivanje filtera sa najmanjom L1 normom. Primjerice, uzmimo mrežu 30% kapaciteta početne mreže. Takva mreža, ako nasumično podrežemo filtre, ostvaruje točnost od 88.190% na skupu za testiranje. No, ako se vratimo na eksperiment 6.6, vidimo da na primjerice 30% kapaciteta ostvarujemo točnost od 89.750%.

7. Zaključak

Podrezivanje je potencijalno vrlo korisna metoda dubokog učenja. U radu se pokazuje da postoji podmreža dubokog modela koja dodatnim podešavanjem težina (učenjem) može očuvati generalizaciju čak i uz 20 puta manje parametara i time značajno manje računarskih operacija. Hipoteza dobitnog listića lutrije se pokazala dobrom kod slobodnog podrezivanja, što nije slučaj kada se ista primjeni na strukturno podrezivanje. Tada uklanjamo cijele filtre čime povećavamo i vjerojatnost uklanjanja parametara koji treniranjem nisu konvergirali prema nuli. Čini se da je cijena takvog smanjenja kapaciteta prevelika u odnosu na benefit dobiven resetiranjem nepodrezanih filtera. Da postoji benefit takvim resetiranjem potvrđuje i činjenica da podrezivanjem filtera sa manjom normom mreža postiže veću točnost na testnom skupu nego mreža kojoj podrezujemo nasumične filtre pri strukturnom podrezivanju hipotezom dobitnog listića lutrije.

Nadalje, podrezivanje kotinuiranim ugađanjem (eng. fine-tuning) daje značajno bolje rezultate od hipoteze dobitnog listića lutrije pri strukturnom podrezivanju te se pokazuje iznimno korisnim u primjeni. Strukturno podrezivanje kod arhitektura sa preskočnim vezama i normalizacijom daje značajno bolje rezultate nego kod konvencionalne unaprijedne arhitekture. Kod velikih slika i modela velikog kapaciteta, podrezivanjem značajno ubrzavamo evaluaciju na grafičkim karticama. Ovaj rad pronalazi podmrežu mreže ResNet-18 koja bolje generalizira i daje 1.37% veću točnost na skupu za testiranje, čiji je kapacitet tek 20% kapaciteta originalne mreže te koji obrađuje duplo više slika u jedinici vremena od mreže punog kapaciteta.

Budući radovi uključivali bi eksperimente sa analizom hiperparametara te unakrsnom validacijom kod podrezivanja. Potrebno je dodatno eksperimentirati sa hipotezom dobitnog listića lutrije u slučaju strukturnog podrezivanja kod modela sa preskočnim vezama i skupnom normalizacijom. Također, bilo bi interesantno isprobati sofisticiranije heuristike strukturnog podrezivanja od podrezivanja magnitudom, podrezivati neurone u potpuno povezanim slojevima te isprobati tehnike podrezivanja na različitim arhitekturama i problemima. Problem ovih eksperimenata je da su računarski zahtjevni te uzimaju puno vremena, pa je i broj eksperimenata samim time limitiran računarskim mogućnostima uređaja.

8. Literatura

[1] Deep compression pruning. https://www.aitrends.com/ai-insider/deep-compression-pruning-machine-learning-ai-self-driving-cars-usin Accessed: 2020-06-02.

[2] Sajid Anwar, Kyuyeon Hwang, i Wonyong Sung. Structured pruning of deep convolutional neural networks. *CoRR*, abs/1512.08571, 2015. URL http://arxiv.org/abs/1512.08571.

[3] Jonathan Frankle i Michael Carbin. The lottery ticket hypothesis: Training pruned neural networks. *CoRR*, abs/1803.03635, 2018. URL http://arxiv.org/abs/1803.03635.

[4] Xavier Glorot i Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. U Yee Whye Teh i Mike Titterington, urednici, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, svezak 9 od *Proceedings of Machine Learning Research*, stranice 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR. URL http://proceedings.mlr.press/v9/glorot10a.html.

[5] Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, i Yi Yang. Soft filter pruning for accelerating deep convolutional neural networks, 2018.

[6] Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, i Yi Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration, 2018.

[7] Sergey Ioffe i Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015. URL http://arxiv.org/abs/1502.03167.

[8] Guillaume Leclerc, Manasi Vartak, Raul Castro Fernandez, Tim Kraska, i Samuel Madden. Smallify: Learning network size while training, 2018.

[9] Yann LeCun, Léon Bottou, Yoshua Bengio, i Patrick Haffner. Gradient-based learning applied to document recognition. U *Proceedings of the IEEE*, svezak 86, stranice 2278–2324, 1998. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.42.7665.

[10] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, i Hans Peter Graf. Pruning filters for efficient convnets. *CoRR*, abs/1608.08710, 2016. URL http://arxiv.org/abs/1608.08710.

[11] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, i Tom Goldstein. Visualizing the loss landscape of neural nets, 2017.

[12] Ji Lin, Yongming Rao, Jiwen Lu, i Jie Zhou. Runtime neural pruning. U I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, i R. Garnett, urednici, *Advances in Neural Information Processing Systems 30*, stranice 2181–2191. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/6813-runtime-neural-pruning.pdf.

[13] Jian-Hao Luo, Jianxin Wu, i Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. *CoRR*, abs/1707.06342, 2017. URL http://arxiv.org/abs/1707.06342.

[14] Marc Masana, Joost van de Weijer, Luis Herranz, Andrew D. Bagdanov, i Jose M. Álvarez. Domain-adaptive deep network compression. *CoRR*, abs/1709.01041, 2017. URL http://arxiv.org/abs/1709.01041.

[15] Sejun Park, Jaeho Lee, Sangwoo Mo, i Jinwoo Shin. Lookahead: A far-sighted alternative of magnitude-based pruning, 2020.

[16] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, i Aleksander Madry. How does batch normalization help optimization? U S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, i R. Garnett, urednici, *Advances in Neural Information Processing Systems 31*, stranice 2483–2493. Curran Associates, Inc., 2018. URL http://papers.nips.cc/paper/7515-how-does-batch-normalization-help-optimization.pdf.

[17] Connor Shorten. Deep Compression towarddatascience. https://towardsdatascience.com/deep-compression-7b771b3aa773. Accessed: 2020-06-02.

[18] Karen Simonyan i Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2014.

[19] VainF. Torch-pruning. https://github.com/VainF/Torch-Pruning, 2019.

[20] Hattie Zhou, Janice Lan, Rosanne Liu, i Jason Yosinski. Deconstructing lottery tickets: Zeros, signs, and the supermask. *CoRR*, abs/1905.01067, 2019. URL http://arxiv.org/abs/1905.01067.

[21] Neta Zmora, Guy Jacob, Lev Zlotnik, Bar Elharar, i Gal Novik. Neural network distiller: A python package for dnn compression research. October 2019. URL https://arxiv.org/abs/1910.12232.

Optimiranje konvolucijskih modela iterativnim podrezivanjem

Sažetak

Ovaj rad se bavi pronalažanjem podmreža malog kapaciteta i dobre generalizacije unutar većeg dubokog modela. Pretraživanje se obavlja postupkom strukturnog podrezivanja i podrezivanja težina. Eksperimenti su rađeni nad VGG-19 i ResNet-18 arhitekturama. Pokazuje se da postoji podmreža mreže ResNet-18 koja nad CIFAR-10 skupom za učenje postiže 1.37% veću točnost i ima 20% parametara originalnog modela. Uz to, takva podmreža postiže 2.5 puta bržu evaluaciju na GPU uređaju od originalne mreže.

Ključne riječi: Duboko učenje, podrezivanje, neuronska mreža, hipoteza dobitnog listića lutrije, CNN, VGG, ResNet.

Optimizing convolutional models by iterative pruning

Abstract

The goal of this thesis is to find small capacity and good generalization subnet within a larger deep network. The search is performed by the process of structural and weight pruning. Experiments were done using VGG-19 and ResNet-18 architectures. It is shown that there is a sub-network of the ResNet-18 network that achieves 1.37 % higher accuracy on the CIFAR-10 dataset and has 20 % parameters of the original model. Also, it achieves 2.5 times faster evaluations on the GPU device than the original network.

Keywords: Deep learning, pruning, neural network, the lottery ticket hypothesis, CNN, VGG, ResNet.