Universitatea Babeş-Bolyai
Facultatea de Matematică și Informatică

Optimizarea rețelelor neuronale convoluționale prin tăiere și proiectare de arhitectură

Rezumat teză de doctorat

Autor:
Csanád Sándor

Coordonator științific:
Prof. Dr. Horia F. Pop

2024

Cuvinte cheie: tăiere structurată, proiectare de arhitectură, rețele neuronale convoluționale, importanța filtrelor, rețele neuronale convoluționale ierarhice

Cuprins

Cuprinsul tezei

Listă de publicații

Toate clasamentele sunt listate conform clasificării conferințelor [1] din domeniul Informatică.

- Categoria A:
  - **Csanád Sándor**, Szabolcs Pável and Lehel Csató. Pruning cnn's with linear filter ensembles. In Proceedings of the 24th European Conference on Artificial Intelligence (ECAI 2020), volume 325, series: Frontiers in Artificial Intelligence and Applications, pages 1435–1442. IOS Press, 2020.
- Categoria B:
  - **Csanád Sándor**, Szabolcs Pável and Lehel Csató. Neural network pruning based on filter importance values approximated with monte carlo gradient estimation. In Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2022) - volume 5, pages 315–322. INSTICC, SciTePress, 2022.
  - Levente Tempfli and **Csanád Sándor**. HierNet: Image Recognition with Hierarchical Convolutional Networks. In Proceedings of the 16th International Conference on Agents and Artificial Intelligence (ICAART 2024), volume 2, pages 147-155, SciTePress, 2024.
- Categoria D:
  - **Csanád Sándor**. Finding dense supermasks in randomly initialized neural networks. In Proceedings of the 11th International Conference on Applied Informatics (ICAI 2020), volume 2650, pages 288–295. Ceur Workshop Proceedings, 2020.

Punctaj: **17** de puncte.

[1] `https://www.core.edu.au/`

Capitolul 1

Introducere

În ultimul deceniu, rețelele neuronale profunde au revoluționat multe domenii ale informaticii, inclusiv viziunea pe calculator, recunoașterea vorbirii, traducerea automată și prelucrarea limbajului natural. Prin aplicarea rețelelor profunde, am dobândit capacitatea de a aborda o mare varietate de probleme foarte complexe din lumea reală.

Întrucât rețelele neuronale profunde pot rezolva din ce în ce mai multe probleme, există o cerere crescătoare de a le implementa nu numai pe servere cu resurse de calcul și de energie "nelimitate" (cu GPU-uri puternice și zeci sau sute de gigaocteți de RAM), ci și pe diverse dispozitive periferice cu constrângeri energetice și de calcul, cum ar fi smartphone-uri, ceasuri, diverse sisteme integrate și dispozitive IoT. Pentru a satisface această cerere, este esențială dezvoltarea continuă, nu numai pe partea hardware, ci și pe partea software. În ceea ce privește hardware-ul, sunt introduse arhitecturi specializate specifice domeniului, capabile să execute eficient operațiunile de bază din rețelele neuronale profunde. În același timp, pe partea de software, sunt dezvoltați diverși algoritmi de compresie, cum ar fi cuantizarea și restrângerea rețelei, pentru a reduce dimensiunea ei, minimizând astfel accesul la memorie și operațiile în virgulă mobilă sau fixă în timpul inferenței.

Dintre aceste tehnici de compresie, accentul principal al acestei teze se pune pe restrângerea rețelei, care implică identificarea și eliminarea parametrilor redundanți din rețea (figura 1.1a) Procesul de tăiere abordează următoarele probleme la implementarea rețelelor neuronale pe dispozitive cu resurse computaționale limitate:

- **Cerințe de memorie**: În timpul inferenței, parametrii rețelei sunt stocați în memorie, care este adesea limitată pe diverse dispozitive integrate. O rețea mai mică are mai puțini parametri, ceea ce necesită mai puțină memorie. În plus, restrângerea structurată elimină filtre întregi din rețea, ceea ce înseamnă mai puține canale de output în timpul inferenței. Reducerea numărului de canale de output generat de un layer poate reduce semnificativ

cerințele de memorie ale rețelei.

- **Bandwith de memorie**: Parametrii rețelei sunt citiți din memorie, înmulțiți cu intrarea layer-ului, iar rezultatele sunt scrise înapoi în memorie. Dacă rețeaua conține mai puțini parametri, sunt necesare mai puține citiri în memorie. Deoarece accesul la memorie este, în general, o operațiune costisitoare, acest lucru poate accelera semnificativ procesul de inferență. În plus, după cum s-a menționat anterior, tăierea structurată elimină canale întregi din rețea, ceea ce înseamnă că sunt eliminate accesările suplimentare de memorie, deoarece aceste canale (feature maps) nu ar trebui să fie accesate.

- **Complexitate computațională**: Îndepărtarea parametrilor din rețea duce la mai puține operații de adunare și multiplicare, ceea ce accelerează inferența.

- **Consumul de energie**: După cum s-a menționat anterior, o rețea mai mică necesită mai puține accesări de memorie și mai puține operații în virgulă mobilă. Dintre acestea, accesul la memorie în special este o operațiune care necesită multă energie (în comparație cu multiplicările pe 32 de biți, există o discrepanță de aproape două ordine de mărime [19]) și este important să fie redusă cât mai mult posibil numărul acestor operațiuni.

- **Confidențialitate**: Dacă rețelele sunt suficient de mici pentru a încăpea în memoria dispozitivului, diferitele date ale utilizatorului pot fi prelucrate local, fără a fi trimise în cloud. Acest lucru poate fi crucial în diferite aplicații în care rețeaua trebuie să proceseze date sensibile.

În plus față de reducerea dimensiunii rețelei prin reducerea parametrilor sau a lățimii de biți, abordările alternative vizează îmbunătățirea performanței rețelei prin *optimizarea arhitecturii* sau a unui layer specific din cadrul arhitecturii pentru o mai mare eficiență [82, 61, 6]. De exemplu, domeniul proiectării arhitecturilor compacte se concentrează pe crearea de arhitecturi foarte eficiente. Acest lucru se realizează prin înlocuirea operației convenționale de convoluție cu alternative mai eficiente, cum ar fi utilizarea convoluției separabile în adâncime [6, 61]. Această operație alternativă poate extrage un număr echivalent de output canale utilizând mai puțini parametri și mai puține operații în virgulă mobilă. Alte abordări încearcă să reproiecteze structura convențională cu o singură ramură a rețelelor prin introducerea de ramuri suplimentare [80] (figura 1.1b) care efectuează operații în funcție de input-ul rețelei. Deși această abordare conduce de obicei la rețele mai mari, cu mai mulți parametri și mai multe operații în virgulă mobilă, care necesită timpi de antrenări mai lungi, în timpul inferenței trebuie evaluată doar o singură (sau câteva) ramură(e). Ca urmare, rețeaua necesită o cantitate comparabilă de memorie și resurse de calcul ca o rețea tradițională cu o

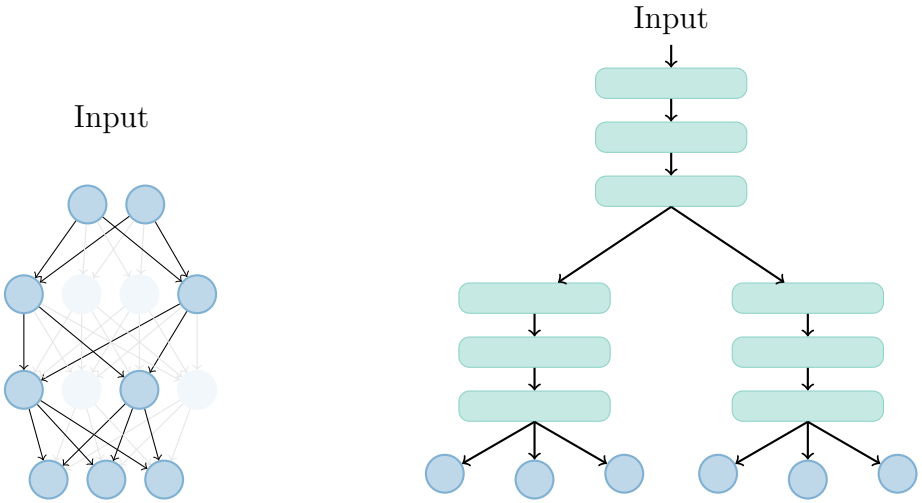


Figura 1.1: (a): Exemplu de rețea tăiată, în care neuronii și conexiunile lor sunt eliminate; (b): Exemplu de rețea neuronală convoluțională cu branch-uri suplimentare.

singură ramură, dar are o capacitate sporită de extragere a caracteristicilor care contribuie la predicții mai precise. În această teză, explorăm de asemenea conceptul de rețele ierarhice, în care rețeaua este organizată ca un arbore de decizie, cu margini reprezentând layere care extrag caracteristicile, și noduri reprezentând clasificatoarele.

1.1 Contribuții la Domeniu

Contribuția acestei teze la domeniul învățării profunde este triplă:

1. Sunt propuse două abordări pentru a estima valorile importanței filtrelor sau neuronilor în layere de convoluție, precum și în layere fully connected. Pe baza acestor valori ale importanței, filtrele cele mai puțin importante sunt eliminate din rețea, rezultând o rețea mai mică cu o pierdere minimă de precizie:

    - Prima abordare se bazează pe metoda linear filter ensembles (LFE), care estimează importanța filtrelor și le elimină iterativ pe cele mai puțin importante (secțiunea 2.1 și [63]).
    - A doua abordare se bazează pe o metodă care construiește o distribuție a probabilităților privind prezența sau absența filtrelor de rețea. Probabi-

lităţile asociate filtrelor sunt deduse prin optimizarea diferitelor funcţii de energie, utilizând trucul log-derivatei şi estimarea gradientului Monte Carlo (secţiunea 2.2 şi [64]).

- Sunt efectuate experimente cu metodele de tăiere pe arhitectura ResNet antrenată folosind setul de date CIFAR-10. Rezultatele demonstrează că metodele pot elimina aproximativ $30-70\%$ din parametri, în funcţie de mărimea modelului. Acest nivel de reducere a parametrilor este comparabil cu cel atins de metodele de ultimă generaţie [25, 47, 24, 45] care utilizează tehnici de tăiere structurată (secţiunea 2.3).

2. Se demonstrează că sub-reţelele aleatorii cu acurateţe ridicată sunt prezente în reţelele iniţializate aleatoriu nu numai în forma dispersată, ci şi în forma densă (Capitolul 3 şi [62]):

   - Prin aplicarea metodelor de tăiere structurată dezvoltate, se demonstrează că reţelele iniţializate aleatoriu conţin *subreţele dense* cu o precizie departe de întâmplare: subreţeaua arhitecturii *iniţializată aleatoriu* LeNet atinge o acurateţe de peste 50% în setul de date MNIST [37].
   - Se arată, de asemenea, că o reţea LeNet largă, neantrenată, are o subreţea cu o precizie de 80% pe setul de date MNIST.

3. Este prezentată o nouă arhitectură de reţea neuronală convoluţională (HierNet), construită sub forma unui arbore de decizie pentru a exploata ierarhia dintre clase (capitolul 4 şi [77]):

   - Se introduce o arhitectură de tip arbore, în care marginile reprezintă layere, care extrag caracteristicile, iar nodurile reprezintă clasificatoarele. Structura arborelui de decizie corespunde relaţiilor ierarhice dintre clase, ceea ce înseamnă că clasele similare din punct de vedere vizual sunt situate în acelaşi subarbore.
   - Se prezintă o metodă semi-supervizată pentru determinarea relaţiilor ierarhice dintre un număr mare de clase.
   - Se arată că această metodă depăşeşte precizia arhitecturii ResNet cu $1-3\%$, demonstrând eficienţa încorporării ierarhiei de intrare în CNN-uri.

Capitolul 2

Tăiere Structurată

Rețele neuronale profunde, deși eficiente, suferă adesea de redundanță și ineficiență datorită numărului vast de parametri pe care îi folosesc [9]. Tăiere structurată abordează aceste probleme prin eliminarea selectivă a întregi filtre sau neuroni, în opoziție cu weight-urile individuale (tăiere nestructurată), menținând astfel structura regulată a rețelei și asigurând compatibilitatea cu hardware-ul și software-ul existent de învățare profundă [42, 36].

Tăierea rețelelor nu numai reduce dimensiunea modelului și costurile computaționale, dar adesea duce și la o generalizare mai bună prin eliminarea overfitting-ului [19]. În acest capitol, propunem două metode avansate pentru tăiere structurată: ansamblurile de filtre liniare (Linear Filter Ensembles - LFE) [63] și tăierea bazată pe gradient probabilistic (Probabilistic Gradient-Based Pruning - PGBP) [64].

2.1 Ansamblurile de Filtre Liniare

Metoda LFE estimează importanța filtrelor dintr-o rețea neuronală convoluțională (CNN) prin considerarea impactului asupra performanței rețelei atunci când mai multe filtre sunt dezactivate simultan. Metoda folosește un model liniar pentru a prezice importanța filtrelor pe baza performanței lor în diferite ansambluri de filtre. Filtrele sunt clasificate în funcție de importanța lor, iar cele mai puțin importante sunt eliminate.

2.1.1 Metoda Propusă

Având un set de date $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}$ și o rețea convoluțională $f(\boldsymbol{x}|\mathcal{W})$ cu intrare $\boldsymbol{x}$ și parametri $\mathcal{W} = \{W^1, \ldots, W^L\}$ în straturile $\{1, 2, \ldots, L\}$, pierderea empirică este:

$$\mathcal{L}(f(\cdot|\mathcal{W}), \mathcal{D}_{train}) = \frac{1}{N} \sum_{i=1}^{N} C(f(\boldsymbol{x}^{(i)}; \mathcal{W}), y^{(i)}), \tag{2.1}$$

unde $C(\cdot, \cdot)$ este funcția de eroare, cum ar fi loss-ul cross-entropy, N este numărul imaginilor din setul de antrenare $\mathcal{D}_{train}$ și $y^{(i)}$ este eticheta adevărată pentru imaginea de intrare $\boldsymbol{x}^{(i)}$.

Introducem vectori de mască binară $\mathbf{z} \in \{0, 1\}^{N_l}$ pentru fiecare strat l, unde N_l reprezintă numărul de filtre sau neuroni din acel strat. Acești vectori indică care unități sunt active. Performanța rețelei cu o mască $\mathbf{z}^{(i)}$ este evaluată ca:

$$s^{(i)} = 1 - \frac{\mathcal{L}_i - \mathcal{L}_{\min}}{\mathcal{L}_{\max} - \mathcal{L}_{\min}}, \tag{2.2}$$

unde $\mathcal{L}_i$ este loss-ul când masca $\mathbf{z}^{(i)}$ este aplicată rețelei, $\mathcal{L}_{\min}$ și $\mathcal{L}_{\max}$ sunt loss-urile minime și maxime observate cu diferite măști, respectiv.

Importanța fiecărui filtru este calculată rezolvând ecuația:

$$\mathbf{Z} \cdot \boldsymbol{\theta} = \boldsymbol{s}, \tag{2.3}$$

unde $\mathbf{Z} = [\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(M)}]^T$ este matricea de măști binare (fiecare rând al matricei este un vector de mască $\mathbf{z}^{(i)}$), $\boldsymbol{s} = [s^{(1)}, \ldots, s^{(M)}]^T$ este un vector coloană de scoruri, iar $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_{N_l}]^T$ este un vector coloană de valori de importanță a filtrelor.

Procesul de tăiere implică eliminarea filtrelor cele mai puțin importante pe baza valorilor lor de import

Dând valorile de importanță $\boldsymbol{\theta}$, sortăm filtrele și determinăm un prag optim de tăiere evaluând acuratețea rețelei pe un set de validare $\mathcal{D}_{val}$. Pragul este ajustat pentru a obține un echilibru între eficacitatea tăierii și conservarea acurateței.

După tăiere, rețeaua este antrenată pe setul de antrenare $\mathcal{D}_{train}$ pentru a recupera orice pierdere de performanță datorită eliminării filtrelor.

2.1.2 Experimente cu LFE

Experimentele au fost efectuate pe un set de date XOR sintetic și pe setul de date CIFAR-10 folosind arhitecturi ResNet. Pentru setul de date XOR, o rețea cu 10 neuroni în stratul ascuns a fost tăiat. Succesul tăierii a fost măsurat de câte ori a fost realizată structura optimă a rețelei (3 neuroni în stratul ascuns).

2.2 Tăiere Bazată pe Metoda Gradient Probabilistic

Metoda LFE presupune că unitățile structurale (neuroni sau filtre) din același strat sunt independente unul de celălalt. Deși această presupunere poate fi valabilă

pentru un singur strat, nu este valabilă atunci când se evaluează importanța acestor unități pe mai multe straturi.

Tăierea rețelei strat cu strat poate aborda parțial această problemă, dar nu poate captura pe deplin adevărata influență a unităților unele asupra celorlalte. Pentru a aborda aceasta, am introdus o metodă mai generalizată. În această abordare, se învață o distribuție de probabilitate asupra unităților rețelei cu scopul de a scădea loss-ul empiric (sau de a crește scorul empiric).

Deși în această lucrare a fost folosită o distribuție Bernoulli simplă, vizând un singur strat la un moment dat, metoda de optimizare bazată pe estimarea gradientului Monte Carlo este suficient de flexibilă pentru a permite utilizarea de modele de probabilitate mai complexe care pot modela dependențele între straturi.

2.2.1 Metoda Propusă

Definim $\mathbf{z}$ ca un vector de variabile aleatoare *binare*, unde fiecare variabilă aleatoare z_i indică dacă unitatea de rețea asociată este activă sau nu.

$$P(z_i = 1) = p_i = \sigma(\theta_i), \tag{2.4}$$

unde σ este funcția sigmoidă.

Scopul este de a optimiza distribuția de probabilitate comună $P_{\boldsymbol{\theta}}(\mathbf{z})$ a lui $\mathbf{z}$ (vezi Ecuatia 2.12), parametrizată de $\boldsymbol{\theta}$, pentru a maximiza scorul așteptat al rețelei:

$$\boldsymbol{\theta} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \underset{\boldsymbol{z} \sim P_{\boldsymbol{\theta}}(\mathbf{z})}{\mathbb{E}} [s(\mu_{\mathcal{W}}(\boldsymbol{x}|\boldsymbol{z}))] = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, S(P_{\boldsymbol{\theta}}), \tag{2.5}$$

unde $\mu_{\mathcal{W}}(\boldsymbol{x})$ este rețeaua, parametrizată de $\mathcal{W}$, $\boldsymbol{x}$ este imaginea de intrare, iar $s(\mu_{\mathcal{W}}(\boldsymbol{x}|\boldsymbol{z}))$ este scorul rețelei când masca $\boldsymbol{z}$ este aplicată pe aceasta.

Folosim ascensiunea gradientului pentru a optimiza parametrii distribuției de probabilitate $P_{\boldsymbol{\theta}}(\mathbf{z})$:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \alpha \nabla_{\boldsymbol{\theta}} S(P_{\boldsymbol{\theta}})|_{\boldsymbol{\theta}_k}. \tag{2.6}$$

Datorită imposibilității evaluării tuturor celor $2^{|\mathcal{W}|}$ combinații de măști pentru a calcula $\nabla_{\boldsymbol{\theta}} S(P_{\boldsymbol{\theta}})|_{\boldsymbol{\theta}_k}$, folosim estimarea gradientului Monte Carlo cu trucul logaritmic pentru a aproxima gradientul. Din trucul logaritmic rezultă că:

$$\nabla_{\boldsymbol{\theta}} P_{\boldsymbol{\theta}}(\boldsymbol{z}) = P_{\boldsymbol{\theta}}(\boldsymbol{z}) \nabla_{\boldsymbol{\theta}} \log P_{\boldsymbol{\theta}}(\boldsymbol{z}). \tag{2.7}$$

Folosind aceasta, gradientul scorului așteptat poate fi reformulat ca:

$$\begin{aligned} \nabla_\theta S(P_{\boldsymbol{\theta}}) &= \int_{\boldsymbol{z}} \nabla_\theta P_{\boldsymbol{\theta}}(\boldsymbol{z}) s(\mu_{\mathcal{W}}(\boldsymbol{x}|\boldsymbol{z})) d\boldsymbol{z} && (2.8) \\ &= \int_{\boldsymbol{z}} \nabla_\theta P_{\boldsymbol{\theta}}(\boldsymbol{z}) \nabla_{\boldsymbol{\theta}} \log P_{\boldsymbol{\theta}}(\boldsymbol{z}) s(\mu_{\mathcal{W}}(\boldsymbol{x}|\boldsymbol{z})) d\boldsymbol{z} && (2.9) \\ &= \underset{\boldsymbol{z} \sim P_{\boldsymbol{\theta}}(\mathbf{z})}{\mathbb{E}} [\nabla \log P_{\boldsymbol{\theta}}(\boldsymbol{z}) s(\mu_{\mathcal{W}}(\boldsymbol{x}|\boldsymbol{z}))], && (2.10) \end{aligned}$$

Aproximăm gradientul folosind estimarea Monte Carlo cu N eșantioane:

$$\nabla_{\boldsymbol{\theta}} S(P_{\boldsymbol{\theta}}) \approx \frac{1}{N} \sum_{i=1}^{N} \nabla_{\theta} \log P_{\boldsymbol{\theta}}(\boldsymbol{z}^{(i)}) s(\mu_{\mathcal{W}}(\boldsymbol{x}|\boldsymbol{z}^{(i)})). \tag{2.11}$$

Tehnicile de reducere a variației, cum ar fi scăderea mediei mobile a scorului și normalizarea după varianța scorului, îmbunătățesc convergența. Diverse funcții de scor (loss-score, acc-score, exp-acc-score) sunt testate pentru eficacitate, ajustând calculațiile de scor pentru a îmbunătăți convergența.

Distribuția de probabilitate $P_{\boldsymbol{\theta}}(\mathbf{z})$ este modelată ca un produs de distribuții Bernoulli, presupunând independența între unități:

$$P_{\boldsymbol{\theta}}(\boldsymbol{z}) = \prod_{i} p_i^{z_i} (1 - p_i)^{1 - z_i}, \tag{2.12}$$

unde $p_i = \sigma(\theta_i)$ este probabilitatea ca $z_i = 1$.

Algoritmul de tăiere a rețelei optimizează iterativ distribuția de probabilitate pentru fiecare strat, taie stratul pe baza lui $P_{\boldsymbol{\theta}}(\mathbf{z})$ și ajustează rețeaua. Această tăiere strat cu strat ține cont de interdependențele din cadrul straturilor, îmbunătățind eficiența și eficacitatea procesului de tăiere.

2.2.2 Experimente cu PGBP

PGBP a fost testat pe o arhitectură de tip VGG și pe arhitecturi ResNet antrenate pe CIFAR-10. Pentru rețeaua de tip VGG, filtre aleatoare au fost adăugate straturilor, iar abilitatea algoritmului de a tăia aceste filtre a fost evaluată. În cazul arhitecturilor ResNet, metoda a fost aplicată pe ResNet-32, 56 și 110. Rezultatele au arătat că PGBP poate tăia eficient filtrele menținând în același timp o acuratețe ridicată (vezi Secțiunea 2.3).

2.3 Rezultatele Tăierii Structurate

Atât LFE, cât și PGBP au fost comparate cu alte metode de tăiere din literatură. Comparările s-au bazat pe reducerea parametrilor și FLOP-urilor și impactul corespunzător asupra acurateții.

Tabelul 2.1 rezumă rezultatele pentru ambele metode aplicate pe arhitecturi ResNet și le compară cu alte metode de ultimă oră.

2.4 Concluzii

Acest capitol a introdus două metode de tăiere structurată, LFE și PGBP, și a demonstrat eficacitatea lor pe CIFAR-10 cu arhitecturi ResNet. Ambele metode

ResNet	Metodă	Acuratețe (%)			↓(%)	
		Baseline	Tăiat	Diferență ↓	FLOPs	Parametri
32	SFP [25]	92.63	92.08	0.55	41.5	41.24
	FPGM [24]	92.63	92.82	-0.19	**53.2**	**53.2***
	LFE	92.97	92.42	0.55	46.4	49.35
	PGBP	92.97	92.29	0.68	50.22	43.65
56	PFEC[42]	93.04	93.06	-0.02	27.6	13.7
	SFP [25]	93.59	93.35	0.1	47.14	52.6
	ThiNet [47]	93.8	92.98	0.82	49.78	49.67
	FPGM [24]	93.59	93.49	0.1	47.14	52.6
	LFE	93.44	93.18	0.26	57.64	**68.14**
	Adapt-DCP [45]	93.74	93.77	-0.03	**68.48**	54.80
	PGBP	93.44	93.08	0.36	64.22	57.79
110	PFEC[42]	93.53	93.3	0.23	38.6	32.40
	SFP [25]	93.68	93.86	-0.18	40.8	40.72*
	FPGM [24]	93.68	93.85	-0.17	52.3	52.7*
	LFE	94.05	93.48	0.57	63.68	60.08
	PGBP	94.05	93.45	0.6	**72.53**	**68.89**

Tabela 2.1: Compararea arhitecturii ResNet (antrenate pe setul de date CIFAR-10) cu rezultatele din literatură.

au arătat rezultate comparabile cu tehnici de ultimă oră, cu PGBP obținând rate de compresie deosebit de ridicate. Lucrările viitoare ar putea explora explorarea acestor metode pe seturi de date mai mari și arhitecturi mai complexe pentru a valida în continuare eficacitatea și robustețea lor.

Capitolul 3

Măști Dense în Rețele Neuronale Inițializate Aleator

Capitolul prezintă conceptul de *supermăști dense* (dense supermasks) în rețele neuronale inițializate aleator, neantrenate, construind pe lucrarea de bază prezentată în [62].

Cercetările anterioare au stabilit prezența subrețelelor rare în rețelele inițializate aleator care se comportă comparabil cu rețelele antrenate [83, 57]. Inspirat de aceste descoperiri, ne propunem să explorăm subrețelele dense, numite supermăști, care demonstrează o acuratețe semnificativ mai mare decât nivelurile de șansă în rețelele neantrenate.

3.1 Metode de Tăiere

Am folosit mai multe metode de tăiere pentru a identifica aceste subrețele dense:

1. **Normele L_1 și L_2**: Aceste norme măsoară importanța neuronilor pe baza valorilor de parametri. Neuronii cu norme mai mici sunt considerați mai puțin importanți și sunt tăiați primii.

2. **Ansamblurile de filtre liniare (Linear Filter Ensembles - LFE)**: Această metodă evaluează importanța neuronilor prin analizarea loss-ul rețelei când sunt aplicate diferite ansambluri de filtre.

3. **Tăierea bazată pe gradient probabilistic (Probabilistic Gradient-Based Pruning - PGBP)**: Această abordare maximizează scorul așteptat al rețelei prin eșantionarea măștilor dintr-o distribuție de probabilitate și rafinarea acestor probabilități folosind estimarea gradientului Monte Carlo.

4. Ca un caz de control, am considerat și **tăierea aleatorie**, unde neuronii sunt tăiați aleator.

3.2 Configurația Experimentală

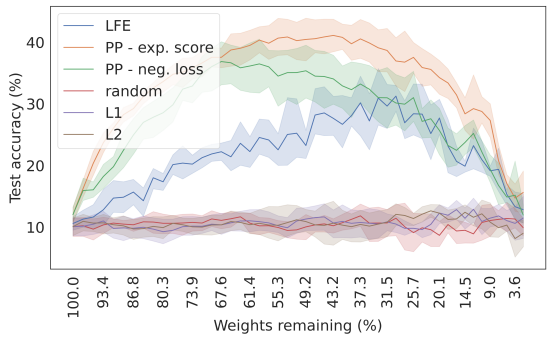
Am efectuat experimente pe setul de date MNIST folosind arhitectura LeNet-300-100. Această rețea cuprinde un strat de intrare cu 784 de unități, urmat de două straturi ascunse cu 300 și 100 de unități, și un strat de ieșire cu 10 unități. Parametrii rețelei au fost inițializați dintr-o distribuție normală, și nu s-a aplicat niciun antrenament. Tăierea a fost efectuată atât într-un singur pas, cât și în moduri iterative, reducând progresiv dimensiunea rețelei prin eliminarea neuronilor cel mai puțin importanți.

Am experimentat, de asemenea, cu arhitectura Wide-LeNet, o versiune extinsă a modelului tradițional LeNet, care prezintă două straturi ascunse complet conectate cu 3010 și 1010 neuroni. Această lățime crescută îmbunătățește probabilitatea de a identifica subrețele cu performanțe ridicate într-un spațiu de parametri mai mare.

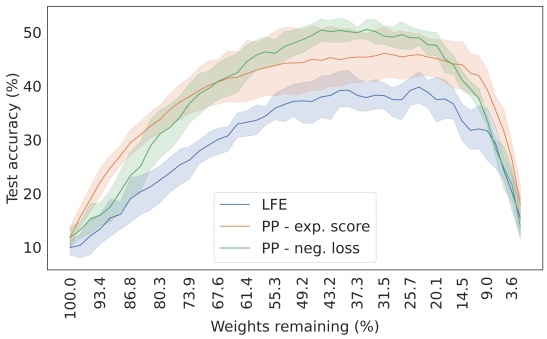
3.3 Rezultate

Figura 3.1 arată acuratețea rețelelor tăiate pe măsură ce sunt eliminate mai mulți parametri. Rezultatele sunt rezumate astfel:

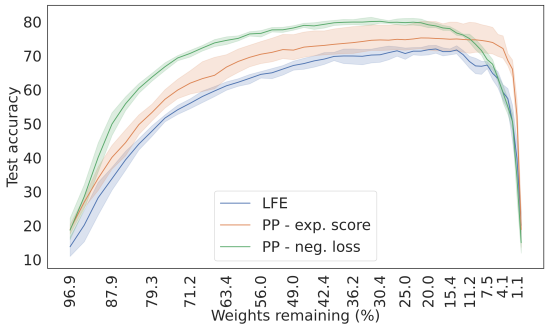
1. **Arhitectura LeNet-300-100**:
   - **Tăiere aleatorie**: Acuratețea a rămas la 10%.
   - **Tăierea cu normele L_1 și L_2**: Nu s-a observat nicio îmbunătățire semnificativă a acurateței.
   - **Tăierea cu ansamblurile de filtre liniare (LFE)**: A atins o acuratețe maximă de 34.2% cu 70% din parametri eliminați.
   - **Tăierea probabilistică**: Cu scorul de loss negativ, acuratețea a atins 36.86% după eliminarea a 33% din parametri, și 41.08% cu funcția de scor exponențială după eliminarea a 67% din parametri.
2. **Tăiere iterativă**:
   - **Metoda LFE**: A atins o acuratețe maximă de 39.8%.
   - **Tăierea probabilistică**: A obținut o acuratețe de 46.82% cu funcția de scor exponențială și 50.52% cu funcția de scor de loss negativ.



(a) LeNet, acuratețe la tăierea într-un singur pas



(b) LeNet, acuratețe la tăierea iterativă



(c) Wide-LeNet, acuratețe la tăierea iterativă

Figura 3.1: Rețele inițializate aleator (neantrenate), tăiate LeNet-300-100 și Wide-LeNet evaluate pe setul de date MNIST. Schimbarea acurateței pe măsură ce sunt eliminați mai mulți parametri din modele. Figura (a) arată rezultatele experimentului de tăiere într-un singur pas, Figura (b) arată rezultatele experimentului de tăiere iterativă, iar Figura (c) arată rezultatele experimentului de tăiere iterativă Wide-LeNet.

3. **Arhitectura Wide-LeNet**:

   - **Tăierea iterativă**: Rețeaua, inițial cu 3010 și 1010 neuroni în primul și al doilea strat ascuns, a obținut peste 80% acuratețe după ce 70% din parametri au fost tăiați folosind metoda probabilistică cu funcția de scor de loss negativ. Cu o dimensiune redusă chiar si cu 10%, rețeaua a menținut o acuratețe de 73.3%.

3.4 Concluzii

Experimentele noastre au confirmat că există subrețele dense cu o acuratețe semnificativ mai mare decât șansa aleatorie în rețelele neuronale neantrenate. Folosind ansambluri de filtre liniare și metode de tăiere probabilistice, am demonstrat prezența acestor supermăști atât în arhitecturile LeNet-300-100, cât și în Wide-LeNet. Această descoperire contestă viziunea tradițională asupra inițializării și antrenamentului rețelelor neuronale, sugerând că potențialul pentru subrețele de performanță ridicată există în mod inherent în parametrii inițiali aleatori.

Aceste descoperiri oferă noi direcții pentru proiectarea eficientă a rețelelor, unde accentul poate fi mutat de la antrenamentul extensiv la identificarea subrețelelor inherent care dețin deja o acuratețe ridicată.

Capitolul 4

HierNet: Rețele Neuronale Convoluționale Ierarhice

CNN-urile tradiționale urmează o construcție secvențială, aranjând straturi convoluționale de la stratul de intrare la straturi complet conectate. Această configurație permite rețelei să captureze detalii fine în straturile inițiale și caracteristici de nivel progresiv superior în etapele ulterioare, oferind o reprezentare vizuală cuprinzătoare a unui obiect. Cu toate acestea, aceste modele tratează toate clasele în mod egal, neglijând ierarhia inerentă dintre clasele de date. De exemplu, unele clase au asemănări vizuale, cum ar fi câinii și pisicile sau florile care seamănă mai mult între ele decât cu animalele.

Pentru a valorifica această ierarhie inerentă a claselor, introducem *HierNet*, o arhitectură CNN ierarhică asemănătoare unui arbore de decizie (figura 4.1). În HierNet, marginile reprezintă operații convoluționale pentru extragerea caracteristicilor, în timp ce nodurile efectuează clasificare pentru a determina următoarea rută pe baza caracteristicilor extrase. Predicțiile finale de clasă sunt generate de nodurile frunză. Întregul arbore este antrenat, dar în timpul inferenței, este evaluată doar o singură cale de la rădăcină la un nod frunză.

4.1 Arhitectura și Antrenarea HierNet

Arhitectura constă din noduri (V) și margini (E), formând un arbore cu un nod rădăcină unic. Fiecare margine are operații de extragere a caracteristicilor, iar nodurile conțin funcții de clasificare pentru a direcționa inputul către nodul copil corespunzător. Nodul rădăcină trimite imaginea de intrare la prima margine. Funcțiile de clasificare de la noduri includ flattening sau pooling, transformări liniare și operații softmax.

HierNet folosește o secvență de operații similare cu CNN-urile standard, cum

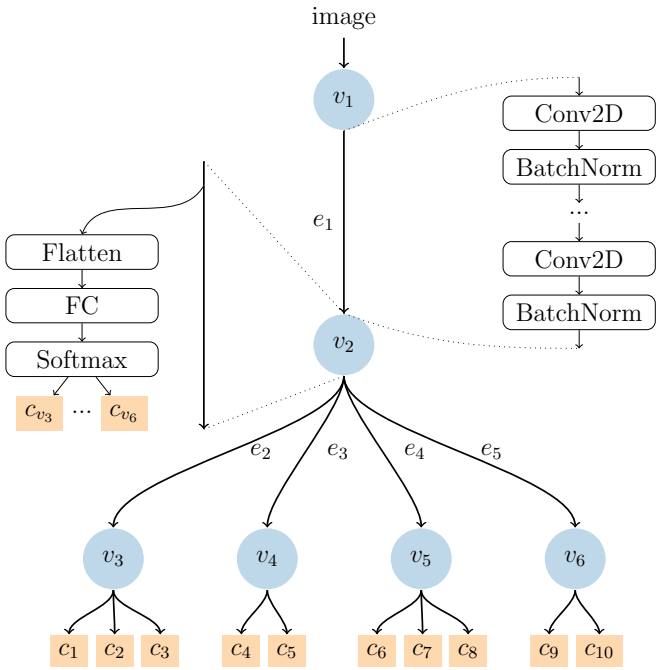


Figura 4.1: Arhitectura HierNet cu o topologie de arbore, operațiile de extragere a caracteristicilor în margini și operațiile de clasificare în noduri.

ar fi ResNet. Așa cum arată figura 4.2, straturile CNN-ului standard sunt distribuite între margini de la rădăcină la frunză, menținând aceeași ordine ca în modelul original. Diferența principală este că HierNet are clasificatori în fiecare nod (cu excepția rădăcinii), spre deosebire de stratul de clasificare unic din modelele tradiționale.

HierNet poate fi antrenat end-to-end, cu probabilitatea de ieșire a fiecărui nod frunză fiind produsul propriei ieșiri și a probabilităților tuturor nodurilor anterioare. Ordinea arborelui ierarhic necesită reordonarea etichetelor setului de date pentru a se potrivi ierarhiei. Se folosește lossul cross-entropy pentru antrenare, iar setările de antrenare reflectă cele ale modelului backbone. Metricile includ "acuratețea condiționată" pentru probabilitățile de ieșire concatenate și "acuratețea de rutare" pentru calea corectă de la rădăcină la nodul frunză.

Inferența implică evaluarea unei singure rute de la rădăcină la un nod frunză. Acest proces include selectarea primei margini, rularea extragerii de caracteristici, calcularea probabilităților în nodul de clasificare și determinarea următoarei rute până la atingerea unui nod frunză.

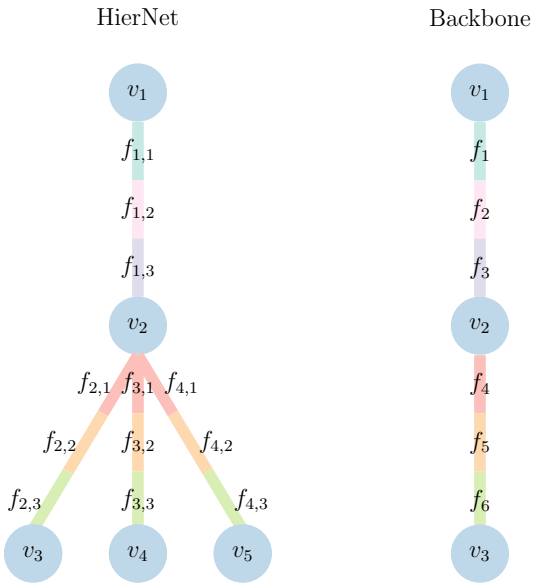


Figura 4.2: Straturile din HierNet și modelul backbone sunt evidențiate în culori diferite, indicând seturi de parametri specifici. Așa cum se arată, HierNet reflectă straturile modelului backbone.

4.2 Construcția Ierarhiei și Algoritmul de Grupare

Ierarhia claselor este reprezentată de topologia arborelui. În timp ce construcția manuală este posibilă pentru seturi de date mici, introducem o metodă automată folosind o matrice de probabilitatede confuzie (Confusion Propability Matrix - CPM) pentru a grupa clasele cu aspect similar. CPM-ul capturează probabilitățile de clasificare greșite între clase, ceea ce ajută la definirea proximității dintre clase. Clasele sunt grupate pe baza proximității lor, cu constrângeri privind dimensiunea grupului și similaritatea minimă, pentru a asigura ierarhii semnificative.

4.3 Experimente și Rezultate

Am experimentat cu HierNet pe setul de date CIFAR-100, care are 100 de clase și este potrivit pentru testarea algoritmului nostru de grupare. Hiperparametrii cheie includ punctul de divizare al CNN-ului backbone, numărul de straturi de clasificare suplimentare, proximitatea minimă a membrilor grupului și dimensiunea maximă a grupului. Valorile optime au fost găsite prin teste extinse. Am augmentat setul de date cu flipping și translație orizontală aleatorie, și am evaluat modelele folosind

#straturi		ResNet	HierNet		ELU ResNet	HierNet
20		65.96	**68.08**		65.54	**68.16**
32		67.08	**70.45**		67.88	**70.43**
44		68.12	**70.75**		68.79	**70.79**
56		68.38	**72.01**		69.03	**72.29**
110		71.33	**73.27**		72.93	**74.15**

Tabela 4.1: Comparație a acurateții HierNet-ului și a backbone-urilor ResNet și ELU ResNet pentru diferite dimensiuni de rețea

acuratețea ca metrică principală.

Rezultatele demonstrează avantajul semnificativ al HierNet față de modelele backbone (tabelul 4.1). HierNet depășește în mod constant modelele corespunzătoare ResNet și ELU ResNet pentru toate dimensiunile de rețea testate. De exemplu, modelul HierNet cu 32 de straturi atinge o acuratețe de 70.45%, depășind acuratețea de 68.38% a ResNet-ului cu 56 de straturi. Similar, modelul HierNet cu 32 de straturi bazat pe backbone-ul ELU ResNet atinge o acuratețe de 70.43%, comparativ cu acuratețea de 69.03% a ELU ResNet-ului cu 56 de straturi. Acest lucru evidențiază eficiența și performanța superioară a HierNet-ului, în ciuda dimensiunii sale mai mici.

4.4 Concluzii

Acest capitol a introdus HierNet, un CNN ierarhic care valorifică asemănările vizuale și ierarhiile de clase pentru a îmbunătăți acuratețea clasificării. HierNet a depășit modelele tradiționale ResNet, demonstrând eficacitatea exploatării ierarhiilor de clase. Lucrările viitoare ar putea include rafinarea algoritmului de grupare pentru a echilibra mai bine distribuțiile de clase și optimizarea configurațiilor de antrenare specifice HierNet-ului.

Capitolul 5

Concluzii și direcții de cercetare viitoare

Această teză aduce contribuții în domeniul învățării profunde prin introducerea unor metode noi de pruning structurat și a unei noi arhitecturi proiectate pentru a îmbunătăți eficiența rețelelor neuronale convoluționale (CNN).

Pentru a reduce numărul de filtre și neuroni din CNN-uri, metoda ansamblurilor de filtre liniare atribuie valori de importanță filtrelor din straturile convoluționale și neuronilor din straturile complet conectate prin construirea și evaluarea ansamblurilor de filtre liniare. Experimentele, efectuate pe modele antrenate pe un set de date mic asemănător cu XOR și pe setul de date CIFAR-10, au demonstrat capacitatea acestei metode de a identifica și elimina filtre redundante din rețea. În special, când tehnica de pruning a fost aplicată la diferite arhitecturi ResNet, rezultatele obținute au fost comparabile cu cele obținute de diverse metode state-of-the-art. Metoda de pruning bazată pe metoda gradient probabilistic (PGBP) a fost introdusă ca o abordare alternativă pentru a estima importanța filtrelor și neuronilor prin construirea unei distribuții de probabilitate asupra filtrelor folosind trucul log-derivative și estimarea gradientului Monte Carlo. Experimentele au demonstrat eficacitatea acestei metode în identificarea filtrelor aleatorii adăugate la rețelele pre-antrenate și în pruningul arhitecturii ResNet-110 antrenate pe setul de date CIFAR-10, eliminând aproximativ 70% din parametri.

A fost investigată existența subrețelelor dense în rețelele inițializate aleator, neantrenate, care obțin o acuratețe departe de întâmplare. Cu ajutorul metodelor LFE și PGBP, s-a arătat că arhitectura LeNet-300-100 inițializată aleator conține o subrețea care obține o acuratețe de 50% pe setul de date MNIST, în timp ce pruningul versiunii mai late a acestei arhitecturi găsește o subrețea care obține o acuratețe de 80%.

A fost introdusă o arhitectură nouă de tip arbore, HierNet, care poate exploata relațiile ierarhice dintre clase. În această rețea, marginile reprezintă straturile

de extragere a caracteristicilor, în timp ce nodurile efectuează clasificările pentru a clasifica mai întâi imaginile în superclase și apoi pentru a efectua clasificarea detaliată pe ramura selectată. Această abordare a obținut o acuratețe cu $2-3\%$ mai mare pe setul de date CIFAR-100 în comparație cu modelul de bază ResNet, cu doar câteva operații în virgulă mobilă suplimentare necesare în timpul inferenței.

Posibilele direcții de cercetare viitoare din partea pruningului includ dezvoltarea de modele de probabilitate care pot estima importanța unităților nu doar într-un singur strat, ci și pe mai multe straturi. Acest lucru este posibil, deoarece metoda PGBP este suficient de flexibilă pentru a permite utilizarea de modelelor de probabilitate mai complexe. O altă direcție interesantă este de a experimenta cu pruningul în timpul antrenării, similar cu metoda dropout, care dezactivează aleator neuronii în timpul antrenării pentru a crește robustețea rețelei și a îmbunătăți generalizarea. Cu ajutorul PGBP, rețeaua ar putea să se adapteze continuu la structura evolutivă prin activarea sau dezactivarea selectivă a unităților în funcție de probabilitățile învățate.

HierNet ar putea fi îmbunătățit prin automatizarea construcției structurii modelului. Acest lucru poate fi realizat prin construirea dinamică și evaluarea continuă a modelului HierNet, începând cu o arhitectură CNN mai mică fără noduri de decizie intermediare și adăugând ulterior noi straturi și noduri de decizie pe baza matricei de probabilitate a confuziei. Un avantaj al acestei metode este că generează în mod inherent un graf asimetric bazat pe similaritatea claselor. Acest lucru concentrează operațiile de extragere a caracteristicilor scumpe cu privința resurselor acolo unde sunt cel mai mult necesare, rezultând într-o procesare mai eficientă din punct de vedere al costurilor.

Bibliografie

[1] Kambiz Azarian și alții, "Learned Threshold Pruning", în *CoRR* abs/2003 00075 (2020), arXiv: 200300075.

[2] Yoshua Bengio, Nicholas Léonard și Aaron C. Courville, "Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation", în *CoRR* abs/1308.3432 (2013), arXiv: 1308.3432, URL: http://arxiv.org/abs/1308.3432.

[3] Christopher M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*, Berlin, Heidelberg: Springer-Verlag, 2006, ISBN: 0387310738.

[4] Davis Blalock și alții, "What is the State of Neural Network Pruning?", în *Proceedings of Machine Learning and Systems*, ed. de I. Dhillon, D. Papailiopoulos și V. Sze, vol. 2, 2020, pp. 129–146, URL: https://proceedings.mlsys.org/paper/2020/file/d2ddea18f00665ce8623e36bd4e3c7c5-Paper.pdf.

[5] Tom Brown și alții, "Language Models are Few-Shot Learners", în *Advances in Neural Information Processing Systems*, ed. de H. Larochelle și alții, vol. 33, Curran Associates, Inc., 2020, pp. 1877–1901.

[6] François Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions", în *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 1800–1807, DOI: 10.1109/CVPR.2017.195.

[7] Djork-Arné Clevert, Thomas Unterthiner și Sepp Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)", în *arXiv preprint arXiv:1511.07289* (2015).

[8] Jia Deng și alții, "Imagenet: A large-scale hierarchical image database", în *2009 IEEE conference on computer vision and pattern recognition*, Ieee, 2009, pp. 248–255.

[9] Misha Denil și alții, "Predicting Parameters in Deep Learning", în *Advances in Neural Information Processing Systems*, ed. de C.J. Burges și alții, vol. 26, Curran Associates, Inc., 2013, URL: https://proceedings.neurips.cc/paper_files/paper/2013/file/7fec306d1e665bc9c748b5d2b99a6e97-Paper.pdf.

[10] Jacob Devlin și alții, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", în *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota: Association for Computational Linguistics, Iun. 2019, pp. 4171–4186, DOI: 10.18653/v1/N19-1423, URL: https://aclanthology.org/N19-1423.

[11] Xiaohan Ding și alții, "Repvgg: Making vgg-style convnets great again", în *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 13733–13742.

[12] Alexander Finkelstein, Uri Almog și Mark Grobman, *Fighting Quantization Bias With Bias*, 2019, arXiv: 1906.03193 [cs.LG].

[13] Jonathan Frankle și Michael Carbin, "The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks", în *ICLR'2019*, 2019, URL: https://openreview.net/forum?id=rJl-b3RcF7.

[14] Jonathan Frankle și alții, *Stabilizing the Lottery Ticket Hypothesis*, 2020, arXiv: 1903.01611 [cs.LG].

[15] Amir Gholami și alții, "A Survey of Quantization Methods for Efficient Neural Network Inference", în *CoRR* abs/2103.13630 (2021), arXiv: 2103.13630, URL: https://arxiv.org/abs/2103.13630.

[16] Aidan N. Gomez și alții, "Learning Sparse Networks Using Targeted Dropout", în *CoRR* abs/1905.13678 (2019), arXiv: 1905.13678, URL: http://arxiv.org/abs/1905.13678.

[17] Ian Goodfellow, Yoshua Bengio și Aaron Courville, *Deep Learning*, http://www.deeplearningbook.org, MIT Press, 2016.

[18] Shixiang Shane Gu și alții, "MuProp: Unbiased Backpropagation for Stochastic Neural Networks", în *CoRR* abs/1511.05176 (2016).

[19] Song Han, Huizi Mao și William J. Dally, "Deep Compression: Compressing Deep Neural Network with Pruning, Trained Quantization and Huffman Coding", în *CoRR* abs/1510.00149 (2015), arXiv: 1510.00149, URL: http://arxiv.org/abs/1510.00149.

[20] Song Han și alții, "Learning Both Weights and Connections for Efficient Neural Networks", în *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, Montreal, Canada: MIT Press, 2015, pp. 1135–1143.

[21] Babak Hassibi și alții, "Optimal Brain Surgeon: Extensions and Performance Comparisons", în *Proceedings of the 6th International Conference on Neural Information Processing Systems*, NIPS'93, Denver, Colorado: Morgan Kaufmann Publishers Inc., 1993, pp. 263–270, URL: http://dl.acm.org/citation.cfm?id=2987189.2987223.

[22] Kaiming He și Jian Sun, "Convolutional neural networks at constrained time cost", în *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2014), pp. 5353–5360.

[23] Kaiming He și alții, "Deep Residual Learning for Image Recognition", în *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, 2016, pp. 770–778, DOI: 10.1109/CVPR.2016.90, URL: https://doi.org/10.1109/CVPR.2016.90.

[24] Yang He și alții, "Filter Pruning via Geometric Median for Deep Convolutional Neural Networks Acceleration", în *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[25] Yang He și alții, "Soft Filter Pruning for Accelerating Deep Convolutional Neural Networks", în *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, International Joint Conferences on Artificial Intelligence Organization, Iul. 2018, pp. 2234–2240, DOI: 10.24963/ijcai.2018/309, URL: https://doi.org/10.24963/ijcai.2018/309.

[26] John L. Hennessy și David A. Patterson, *Computer Architecture: A Quantitative Approach*, a 5-a ed., Amsterdam: Morgan Kaufmann, 2012, ISBN: 978-0-12-383872-8.

[27] Torsten Hoefler și alții, "Sparsity in Deep Learning: Pruning and Growth for Efficient Inference and Training in Neural Networks", în *J. Mach. Learn. Res.* 22.1 (Ian. 2021), ISSN: 1532-4435.

[28] Hengyuan Hu și alții, *Network Trimming: A Data-Driven Neuron Pruning Approach towards Efficient Deep Architectures*, 2016, arXiv: 1607.03250 [cs.NE].

[29] Sergey Ioffe și Christian Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift", în *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, Lille, France: JMLR.org, 2015, pp. 448–456.

[30] Ruyi Ji și alții, "Attention convolutional binary neural tree for fine-grained visual categorization", în *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 10468–10477.

[31] John M. Jumper și alții, "Highly accurate protein structure prediction with AlphaFold", în *Nature* 596 (2021), pp. 583–589.

[32] Khronos OpenCL Working Group, *The OpenCL Specification, Version 1.1*, ed. de Aaftab Munshi, 2011, URL: `https://www.khronos.org/registry/cl/specs/opencl-1.1.pdf`.

[33] Diederik P. Kingma și Jimmy Ba, "Adam: A Method for Stochastic Optimization", în *ICLR'2015*, ed. de Yoshua Bengio și Yann LeCun, 2015.

[34] Alex Krizhevsky, Vinod Nair și Geoffrey Hinton, *Learning Multiple Layers of Features from Tiny Images*, rap. teh., Faculty of Computer Science, University of Toronto, 2009, URL: `http://www.cs.toronto.edu/~kriz/cifar.html`.

[35] Alex Krizhevsky, Ilya Sutskever și Geoffrey E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks", în *Advances in Neural Information Processing Systems 25*, ed. de F. Pereira și alții, Curran Associates, Inc., 2012, pp. 1097–1105.

[36] Aditya Kusupati și alții, "Soft Threshold Weight Reparameterization for Learnable Sparsity", în *Proceedings of the 37th International Conference on Machine Learning*, ICML'20, JMLR.org, 2020.

[37] Yann LeCun, Corinna Cortes și CJ Burges, "MNIST handwritten digit database", în *ATT Labs [Online]* 2 (2010).

[38] Yann LeCun, John S. Denker și Sara A. Solla, "Optimal Brain Damage", în *Advances in Neural Information Processing Systems 2*, ed. de D. S. Touretzky, Morgan-Kaufmann, 1990, pp. 598–605, URL: `http://papers.nips.cc/paper/250-optimal-brain-damage.pdf`.

[39] Yann Lecun și alții, "Gradient-based learning applied to document recognition", în *Proceedings of the IEEE*, 1998, pp. 2278–2324.

[40] Namhoon Lee, Thalaiyasingam Ajanthan și Philip H. S. Torr, "Snip: single-Shot Network Pruning based on Connection sensitivity", în *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, OpenReview.net, 2019, URL: `https://openreview.net/forum?id=B1VZqjAcYX`.

[41] Bowen Li și alții, "DFQF: Data Free Quantization-aware Fine-tuning", în *Proceedings of The 12th Asian Conference on Machine Learning*, ed. de Sinno Jialin Pan și Masashi Sugiyama, vol. 129, Proceedings of Machine Learning Research, PMLR, Nov. 2020, pp. 289–304, URL: https://proceedings.mlr.press/v129/li20a.html.

[42] Hao Li și alții, "Pruning Filters for Efficient ConvNets", în *International Conference on Learning Representations*, 2017, URL: https://openreview.net/forum?id=rJqFGTslg.

[43] Yuhang Li și alții, "{BRECQ}: Pushing the Limit of Post-Training Quantization by Block Reconstruction", în *International Conference on Learning Representations*, 2021, URL: https://openreview.net/forum?id=POWv6hDd9XH.

[44] Ji Lin și alții, "Runtime Neural Pruning", în *Advances in Neural Information Processing Systems*, ed. de I. Guyon și alții, vol. 30, Curran Associates, Inc., 2017, URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/a51fb975227d6640e4fe47854476d133-Paper.pdf.

[45] Jing Liu și alții, "Discrimination-aware Network Pruning for Deep Model Compression", în *TPAMI'2021* PP (2021), (early access), DOI: 10.1109/TPAMI.2021.3066410.

[46] Christos Louizos, Max Welling și Diederik P. Kingma, "Learning Sparse Neural Networks through L0 Regularization", în *ArXiv* abs/1712.01312 (2017).

[47] J. Luo, J. Wu și W. Lin, "ThiNet: A Filter Level Pruning Method for Deep Neural Network Compression", în *2017 IEEE International Conference on Computer Vision (ICCV)*, vol. 00, Oct. 2018, pp. 5068–5076, DOI: 10.1109/ICCV.2017.541, URL: doi.ieeecomputersociety.org/10.1109/ICCV.2017.541.

[48] Jian-Hao Luo și Jianxin Wu, "AutoPruner: An End-to-End Trainable Filter Pruning Method for Efficient Deep Model Inference", în *CoRR* abs/1805 08941 (2018), arXiv: 180508941.

[49] Martín Abadi și alții, *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*, Software available from tensorflow.org, 2015, URL: https://www.tensorflow.org/.

[50] Eldad Meller și alții, "Same, Same But Different: Recovering Neural Network Quantization Error Through Weight Factorization", în *Proceedings of the 36th International Conference on Machine Learning*, ed. de Kamalika Chaudhuri și Ruslan Salakhutdinov, vol. 97, Proceedings of Machine Learning Research, PMLR, Iun. 2019, pp. 4486–4495.

[51] Pavlo Molchanov și alții, "Pruning Convolutional Neural Networks for Resource Efficient Inference", în *International Conference on Learning Representations*, 2017, URL: https://openreview.net/forum?id=SJGCiw5gl.

[52] MICHAEL C. MOZER și PAUL SMOLENSKY, "Using Relevance to Reduce Network Size Automatically", în *Connection Science* 1.1 (1989), pp. 3–16, DOI: 10.1080/09540098908915626, eprint: https://doi.org/10.1080/09540098908915626, URL: https://doi.org/10.1080/09540098908915626.

[53] Markus Nagel și alții, "A White Paper on Neural Network Quantization", în *ArXiv* abs/2106.08295 (2021).

[54] Markus Nagel și alții, "Up or Down? Adaptive Rounding for Post-Training Quantization", în *CoRR* abs/2004.10568 (2020), arXiv: 2004.10568, URL: https://arxiv.org/abs/2004.10568.

[55] NVIDIA Corporation, *NVIDIA CUDA C Programming Guide*, Version 3.2, 2010.

[56] Adam Paszke și alții, "PyTorch: An Imperative Style, High-Performance Deep Learning Library", în *Advances in Neural Information Processing Systems*, ed. de H. Wallach și alții, vol. 32, Curran Associates, Inc., 2019.

[57] Vivek Ramanujan și alții, *What's Hidden in a Randomly Weighted Neural Network?*, 2019, arXiv: 1911.13299 [cs.CV].

[58] Minsoo Rhu și alții, "Compressing DMA Engine: Leveraging Activation Sparsity for Training Deep Neural Networks", în *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2018, pp. 78–91, DOI: 10.1109/HPCA.2018.00017.

[59] Minsoo Rhu și alții, "Compressing DMA Engine: Leveraging Activation Sparsity for Training Deep Neural Networks", în *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2018, pp. 78–91, DOI: 10.1109/HPCA.2018.00017.

[60] Christian P. Robert și George Casella, *Monte Carlo Statistical Methods*, Springer Publishing Company, Incorporated, 2010, ISBN: 1441919392.

[61] Mark Sandler și alții, "MobileNetV2: Inverted Residuals and Linear Bottlenecks", în *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Iun. 2018.

[62] Csanád Sándor, "Finding Dense Supermasks in Randomly Initialized Neural Networks", în *Proceedings of the 11th International Conference on Applied Informatics (ICAI)* (Eger, Hungary, 29–31 Ian. 2020), ed. de István Fazekas, Gergely Kovásznai și Tibor Tómács, CEUR Workshop Proceedings 2650, Aachen, 2020, pp. 288–295, URL: http://ceur-ws.org/Vol-2650/#paper30.

[63] Csanád Sándor, Szabolcs Pável și Lehel Csató, "Pruning CNN's with Linear Filter Ensembles", în *ECAI 2020 - 24th European Conference on Artificial Intelligence*, 2020, pp. 1435–1442, DOI: 10.3233/FAIA200249, URL: https://doi.org/10.3233/FAIA200249.

[64] Csanád Sándor., Szabolcs Pável. și Lehel Csató., "Neural Network Pruning based on Filter Importance Values Approximated with Monte Carlo Gradient Estimation", în *Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2022) - Volume 5: VISAPP*, INSTICC, SciTePress, 2022, pp. 315–322, ISBN: 978-989-758-555-5, DOI: 10.5220/0010786700003124.

[65] Victor Sanh, Thomas Wolf și Alexander Rush, "Movement Pruning: Adaptive Sparsity by Fine-Tuning", în *Advances in Neural Information Processing Systems*, ed. de H. Larochelle și alții, vol. 33, Curran Associates, Inc., 2020, pp. 20378–20389, URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/eae15aabaa768ae4a5993a8a4f4fa6e4-Paper.pdf.

[66] Shibani Santurkar și alții, "How Does Batch Normalization Help Optimization?", în *Advances in Neural Information Processing Systems*, ed. de S. Bengio și alții, vol. 31, Curran Associates, Inc., 2018.

[67] Pedro Savarese, Hugo Silva și Michael Maire, *Winning the Lottery with Continuous Sparsification*, 2020, URL: https://openreview.net/forum?id=BJe4oxHYPB.

[68] John R. Searle, "Minds, Brains, and Programs", în *Mind Design*, Cambridge, MA, USA: MIT Press, 1985, pp. 282–307, ISBN: 0262580527.

[69] Anish Shah și alții, "Deep Residual Networks with Exponential Linear Unit", în *Proceedings of the Third International Symposium on Computer Vision and the Internet*, VisionNet'16, Jaipur, India: Association for Computing Machinery, 2016, pp. 59–65, ISBN: 9781450343015, DOI: 10.1145/2983402.2983406, URL: https://doi.org/10.1145/2983402.2983406.

[70] Sietsma și Dow, "Neural net pruning-why and how", în *IEEE 1988 International Conference on Neural Networks*, 1988, 325–333 vol.1, DOI: 10.1109/ICNN.1988.23864.

[71] David Silver și alții, "Mastering the game of Go with deep neural networks and tree search", în *Nature* 529 (2016), pp. 484–503, URL: http://www.nature.com/nature/journal/v529/n7587/full/nature16961.html.

[72] Karen Simonyan și Andrew Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition", în *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015, URL: http://arxiv.org/abs/1409.1556.

[73] Nitish Srivastava și alții, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting", în *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958, URL: http://jmlr.org/papers/v15/srivastava14a.html.

[74] Rupesh Kumar Srivastava, Klaus Greff și Jürgen Schmidhuber, "Highway Networks", în *CoRR* abs/1505.00387 (2015), arXiv: 1505.00387, URL: http://arxiv.org/abs/1505.00387.

[75] Chong Min John Tan și Mehul Motani, "DropNet: Reducing Neural Network Complexity via Iterative Pruning", în *Proceedings of the 37th International Conference on Machine Learning*, ed. de Hal Daumé III și Aarti Singh, vol. 119, Proceedings of Machine Learning Research, PMLR, Iul. 2020, pp. 9356–9366, URL: https://proceedings.mlr.press/v119/tan20a.html.

[76] Ryutaro Tanno și alții, "Adaptive neural trees", în *International Conference on Machine Learning*, PMLR, 2019, pp. 6166–6175.

[77] Levente Tempfli. și Csanád Sándor., "HierNet: Image Recognition with Hierarchical Convolutional Networks", în *Proceedings of the 16th International Conference on Agents and Artificial Intelligence - Volume 2: ICAART*, INSTICC, SciTePress, 2024, pp. 147–155, ISBN: 978-989-758-680-4, DOI: 10.5220/0012321100003636.

[78] Georg Thimm și Emile Fiesler, "Evaluating pruning methods", în *1995 International Symposium on Artificial Neural Networks (ISANN'95)*, National Chiao-Tung University, Hsinchu, Taiwan, Republic of China, 1995, A2 20–25, URL: http://infoscience.epfl.ch/record/82305.

[79] Stijn Verdenius, Maarten Stol și Patrick Forré, "Pruning via Iterative Ranking of Sensitivity Statistics", în *CoRR* abs/2006.00896 (2020), arXiv: 2006.00896, URL: https://arxiv.org/abs/2006.00896.

[80] Zhicheng Yan și alții, "HD-CNN: Hierarchical Deep Convolutional Neural Networks for Large Scale Visual Recognition", în *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 2740–2748, DOI: 10.1109/ICCV.2015.314.

[81] Shuochao Yao și alții, "DeepIoT: Compressing Deep Neural Network Structures for Sensing Systems with a Compressor-Critic Framework", în *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems*, SenSys '17, Delft, Netherlands: ACM, 2017, 4:1–4:14, ISBN: 978-1-4503-5459-2, DOI: 10.1145/3131672.3131675, URL: http://doi.acm.org/10.1145/3131672.3131675.

[82] Xiangyu Zhang și alții, "ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices", în *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Iun. 2018.

[83] Hattie Zhou și alții, "Deconstructing Lottery Tickets: Zeros, Signs, and the Supermask", în *Advances in Neural Information Processing Systems 32*, Curran Associates, Inc., 2019, pp. 3592–3602.

[84] Barret Zoph și alții, "Learning Transferable Architectures for Scalable Image Recognition", în *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2017), pp. 8697–8710.