MSc Artificial Intelligence
Master Thesis

Your Title Here

by

Your Name

student number

December 20, 2022

Number of Credits
Period in which the research was carried out

Supervisor:
Dr A Person

Examiner:
Dr A Person

Second reader:
Dr A Person

Universiteit van Amsterdam

Contents

Abstract

Provide a short overview and description of your thesis here.

Chapter 1

Introduction

This document provides an empty template for your thesis. Please keep the following things in mind when writing your thesis:

- The chapter organization of this template is a generic suggestion, please customize to your needs and vision.
- The guideline is to have a 40 page upper limit for the thesis, with additional optional pages for appendices if needed, for example to provide additional experimental results or mathematical details. Most theses do not end up with an appendix.
- For references, you are free to choose your own style. A standard style is included in this template, for example [**lecun2015deep**].
- The official website of the thesis AI (https://student.uva.nl/ai/content/az/master-thesis-ai/master-thesis-ai-2020.html) contains all information regarding the procedures and details of the thesis process.

Chapter 2

Related work

2.1 Pass-through trick

BinnaryConnect: Use the passthrough trick to learn binary (-1 or 1) neural networks [2, 6]

Ternary weight networks [10, 18]

Piggyback: Earliest paper I found (2018) which does the submasking method with the 'pass-through' trick. They actually use it for finetuning models like us. [11]

NASpaper: NAS paper which, independently from the Piggyback paper, used similar tricks for performing neural architecture search [16]

Ramanujan et al: Inspired by the NASPaper, they rediscovered the Piggyback method except applied it to training models from scratch and they also control the sparsity (piggy uses a threshold method). [14]

Movement Pruning: They prune models by applying ramanujan's submasking and the Piggyback method in tandem with training the weights of the neural network (NLP). They use the term 'hard' movement pruning to refer to Ramanujan's method (top k%) and 'soft' pruning to refer to the Piggyback method (prune if score higher than threshold). They additionally enhance the pruning by adding a knowledge distillation loss as well. [15]

Block Movement Pruning: They do movement pruning with also distillation but by grouping 'blocks' of weights together into the same masking unit (so all weights in the block share a mask). Seems to work quite well. [9]

Supsup: Very cool use of the submasking method: They make a random network learn thousands of different tasks, each task with it's own supermask. They base it primarily on Ramanujan et al but also mention Piggyback and argue that their method is superior. [17]

Packnet: Paper doing iterative pruning and retraining to teach a network new tasks [12]

Very recent paper that explores the best ways to fine-tune several models including resnets, vits, and convnexts. Their significant contribution is to freeze the the transformer embeddings in order to get the same performance with SGD as you can get with Adam. They get the highest CLIP-finentuned cifar-10 performance I have seen so far so it is a very good baseline to compare accuracies against [8]

2.2 Self-supervision

Momentum contrast uses an embedding queue in combination with distinct key and query embedders such that the query embedder is trained through backpropagation on the current mini-batch (contrasted against all key embeddings in the queue) and the key embedder is updated by slowly adding the query embedder's parameters to it. Importantly, batch normalization needs to be disabled or shuffled. [4]

Dino is similar to momentum contrast except rather than using embeddings and a queue it just applies a cross-entropy loss between the teacher (key embedder) and student (query embedder) logits. It however needs to apply 'centering' and 'sharpening' to avoid collapse. [1]

SLIP is an improvement over CLIP which also makes use of self-supervision for pretraining. [13]

Pretraining using self-supervision, then pre-training again using self-supervision but on the target dataset, and finally supervised-finetuning of a classifier head works well in NLP [3, 5]

2.3 Ensembling

Ensembles of smaller models appear to outperform larger models. [7]

Chapter 3

Method

3.1 SGD is an EMA

$$\theta_{t+1} = \theta_t - g_{t+1}\gamma_{t+1} \tag{3.1}$$
$$\theta_{t+1} = \theta_t - g_{t+1}\gamma_{t+1} + \theta_t\gamma_{t+1} - \theta_t\gamma_{t+1} \tag{3.2}$$
$$\theta_{t+1} = \theta_t + (\theta_t - g_{t+1})\gamma_{t+1} - \theta_t\gamma_{t+1} \tag{3.3}$$
$$\theta_{t+1} = (\theta_t - g_{t+1})\gamma_{t+1} + \theta_t(1 - \gamma_{t+1}) \tag{3.4}$$
$$\theta_{t+1} = \theta_{t+1\|\theta_t, g_{t+1}}\gamma_{t+1} + \theta_t(1 - \gamma_{t+1}) \tag{3.5}$$

3.2 Translation invariance of threshold and score initialization

$$m_i^t = [\theta_i^t > k] \tag{3.6}$$

where k is the threshold

$$\theta_i^t = \theta_i^0 - \sum_{\hat{t}=0}^{t-1} \gamma^{\hat{t}} g_i(\mathcal{M}^{\hat{t}}, \mathcal{B}^{\hat{t}}) \tag{3.7}$$

where $g_i(\mathcal{M}^{\hat{t}}, \mathcal{B}^{\hat{t}})$ is the gradient of the i'th weight given the mask $\mathcal{M}^{\hat{t}}$ and batch $\mathcal{B}^{\hat{t}}$ (including momentum if enabled)

$$m_i^0 = [\theta_i^0 > k] \qquad = [\theta_i^0 + a > k + a] \tag{3.8}$$
$$m_i^t = [\theta_i^0 - \sum_{\hat{t}=0}^{t-1} \gamma^{\hat{t}} g_i(\mathcal{M}^{\hat{t}}, \mathcal{B}^{\hat{t}}) > k] \qquad = [\theta_i^0 + a - \sum_{\hat{t}=0}^{t-1} \gamma^{\hat{t}} g_i(\mathcal{M}^{\hat{t}}, \mathcal{B}^{\hat{t}}) > k + a] \tag{3.9}$$

For the base case we then have that replacing θ_i^0 with $\theta_i^0 + a$ and k with $k + a$ will not change m_i^0 for any i, which means the network mask $\mathcal{M}^0$ is also invariant to this change. Consequently, m_i^1 is also invariant to this change because of eq. 3.9 and because the gradient $g_i(\mathcal{M}^0, \mathcal{B}^0)$ does not change. The same reasoning can be applied recursively to m_i^2 and so on. Thus, by induction, translating the initial score and threshold by the same amount will not change any of the network masks during training (under simple sgd without weight decay).

3.3 Scale invariance of learning rate and score initialization

Equation for sgd with weight decay:

$$\theta_i^t = \theta_i^{t-1} - \gamma^t \left(g_i(\mathcal{M}^{t-1}, \mathcal{B}^{t-1}) + \lambda \theta_i^{t-1} \right) \tag{3.10}$$

Say we replace γ^t with $\gamma^t \alpha$, θ_i^{t-1} with $\theta_i^{t-1}\alpha$ and λ with $\frac{\lambda}{\alpha}$ for some $\alpha \in \mathbb{R}^+$, then we get:

$$\alpha\theta_i^{t-1} - \alpha\gamma^t \left(g_i(\mathcal{M}^{t-1}, \mathcal{B}^{t-1}) + \frac{\lambda}{\alpha}\alpha\theta_i^{t-1} \right) = \alpha \left(\theta_i^{t-1} - \gamma^t \left(g_i(\mathcal{M}^{t-1}, \mathcal{B}^{t-1}) + \lambda\theta_i^{t-1} \right) \right) \tag{3.11}$$

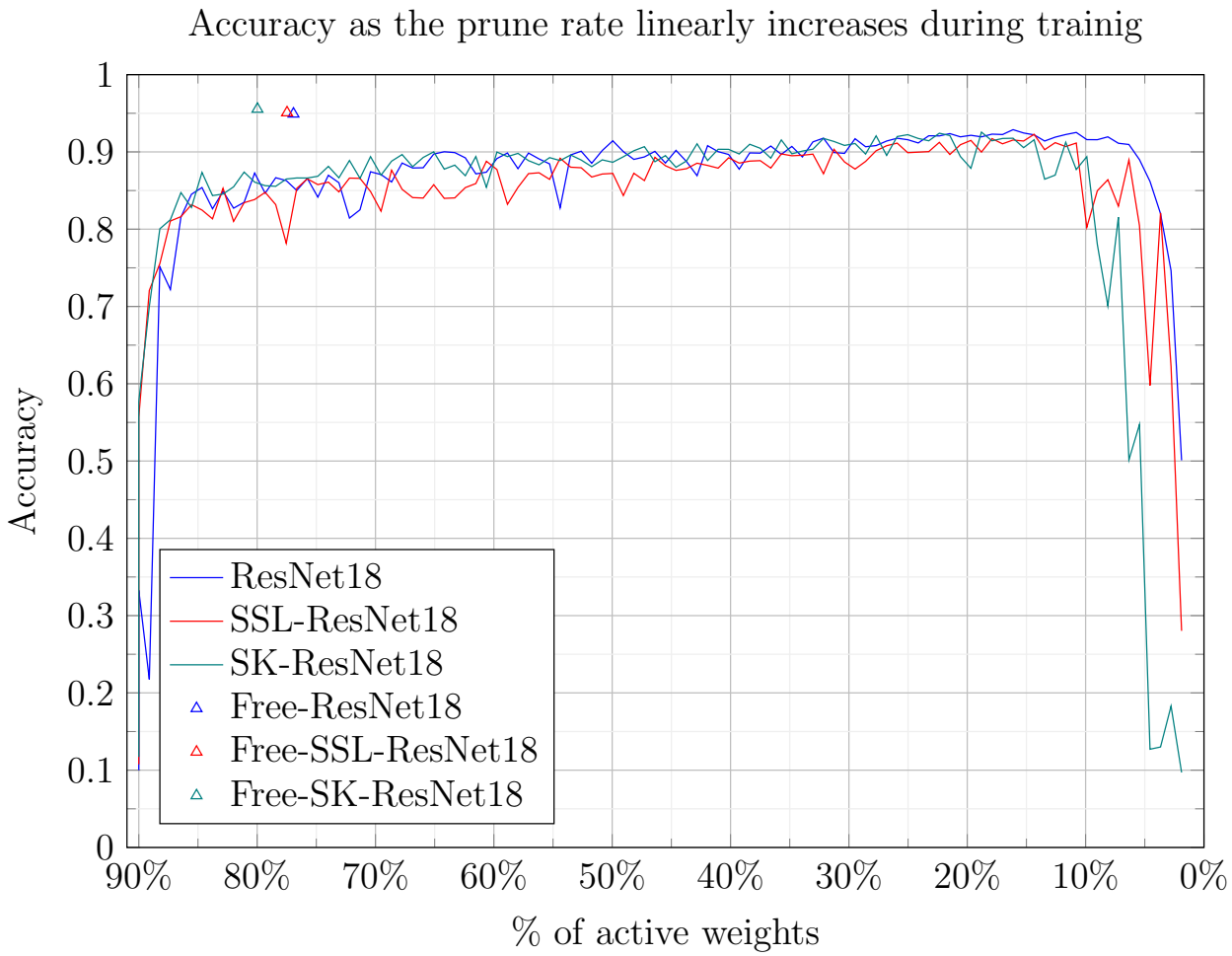
$$= \alpha\theta_i^t \tag{3.12}$$

Similarly to the previous proof, the initial masks $m_i^0 = [\theta_i^0 > 0]$ are invariant to the scale change $m_i^0 = [\alpha\theta_i^0 > 0]$, so the replacement of θ_i^0 with $\alpha\theta_i^0$, combined with the other replacements, does not change the gradient $g_i(\mathcal{M}^0, \mathcal{B}^0)$. Combined with eq 3.12, this means that the updated parameter after the first SGD step is only different in scale when compared to what it would have been without the scale change ($= \alpha\theta_i^t$). Apply this reasoning recursively and it can be seen through induction that the network masks will be the same during training as for the original learning rate, score initialization and weight decay.

Note that the conclusion still holds when momentum is enabled

Chapter 4

Experiments



Chapter 5

Conclusions

Appendix A

Optional appendix

Bibliography

[1] Mathilde Caron et al. *Emerging Properties in Self-Supervised Vision Transformers*. May 24, 2021. DOI: 10.48550/arXiv.2104.14294. arXiv: 2104.14294 [cs]. URL: http://arxiv.org/abs/2104.14294 (visited on 11/22/2022).

[2] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. *BinaryConnect: Training Deep Neural Networks with Binary Weights during Propagations*. Apr. 18, 2016. DOI: 10.48550/arXiv.1511.00363. arXiv: 1511.00363 [cs]. URL: http://arxiv.org/abs/1511.00363 (visited on 11/22/2022).

[3] Suchin Gururangan et al. "Don't Stop Pretraining: Adapt Language Models to Domains and Tasks". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. ACL 2020. Online: Association for Computational Linguistics, July 2020, pp. 8342–8360. DOI: 10.18653/v1/2020.acl-main.740. URL: https://aclanthology.org/2020.acl-main.740 (visited on 12/14/2022).

[4] Kaiming He et al. *Momentum Contrast for Unsupervised Visual Representation Learning*. Mar. 23, 2020. arXiv: 1911.05722 [cs]. URL: http://arxiv.org/abs/1911.05722 (visited on 11/22/2022).

[5] Jeremy Howard and Sebastian Ruder. "Universal Language Model Fine-tuning for Text Classification". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. ACL 2018. Melbourne, Australia: Association for Computational Linguistics, July 2018, pp. 328–339. DOI: 10.18653/v1/P18-1031. URL: https://aclanthology.org/P18-1031 (visited on 12/14/2022).

[6] Itay Hubara et al. "Binarized Neural Networks". In: (), p. 9.

[7] Dan Kondratyuk et al. *When Ensembling Smaller Models Is More Efficient than Single Large Models*. May 1, 2020. arXiv: 2005.00570 [cs, stat]. URL: http://arxiv.org/abs/2005.00570 (visited on 12/14/2022).

[8] Ananya Kumar et al. *How to Fine-Tune Vision Models with SGD*. Nov. 17, 2022. arXiv: 2211.09359 [cs]. URL: http://arxiv.org/abs/2211.09359 (visited on 11/27/2022).

[9] François Lagunas et al. *Block Pruning For Faster Transformers*. Sept. 10, 2021. arXiv: 2109.04838 [cs]. URL: http://arxiv.org/abs/2109.04838 (visited on 11/22/2022).

[10] Fengfu Li et al. *Ternary Weight Networks*. Nov. 20, 2022. DOI: 10.48550/arXiv.1605.04711. arXiv: 1605.04711 [cs]. URL: http://arxiv.org/abs/1605.04711 (visited on 12/16/2022).

[11] Arun Mallya, Dillon Davis, and Svetlana Lazebnik. *Piggyback: Adapting a Single Network to Multiple Tasks by Learning to Mask Weights*. Mar. 16, 2018. DOI: 10.48550/arXiv.1801.06519. arXiv: 1801.06519 [cs]. URL: http://arxiv.org/abs/1801.06519 (visited on 11/22/2022).

[12] Arun Mallya and Svetlana Lazebnik. "PackNet: Adding Multiple Tasks to a Single Network by Iterative Pruning". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Salt Lake City, UT: IEEE, June 2018, pp. 7765–7773. ISBN: 978-1-5386-6420-9. DOI: 10.1109/CVPR.2018.00810. URL: https://ieeexplore.ieee.org/document/8578908/ (visited on 11/22/2022).

[13] Norman Mu et al. *SLIP: Self-supervision Meets Language-Image Pre-training*. Dec. 23, 2021. arXiv: 2112.12750 [cs]. URL: http://arxiv.org/abs/2112.12750 (visited on 11/29/2022).

[14] Vivek Ramanujan et al. "What's Hidden in a Randomly Weighted Neural Network?" Mar. 30, 2020. arXiv: 1911.13299 [cs]. URL: http://arxiv.org/abs/1911.13299 (visited on 07/27/2022).

[15] Victor Sanh, Thomas Wolf, and Alexander M Rush. "Movement Pruning: Adaptive Sparsity by Fine-Tuning". In: (), p. 12.

[16] Mitchell Wortsman, Ali Farhadi, and Mohammad Rastegari. *Discovering Neural Wirings*. Nov. 16, 2019. arXiv: 1906.00586 [cs]. URL: http://arxiv.org/abs/1906.00586 (visited on 11/22/2022).

[17] Mitchell Wortsman et al. "Supermasks in Superposition". In: (), p. 12.

[18] Chenzhuo Zhu et al. *Trained Ternary Quantization*. Feb. 23, 2017. DOI: 10.48550/arXiv.1612.01064. arXiv: 1612.01064 [cs]. URL: http://arxiv.org/abs/1612.01064 (visited on 12/16/2022).