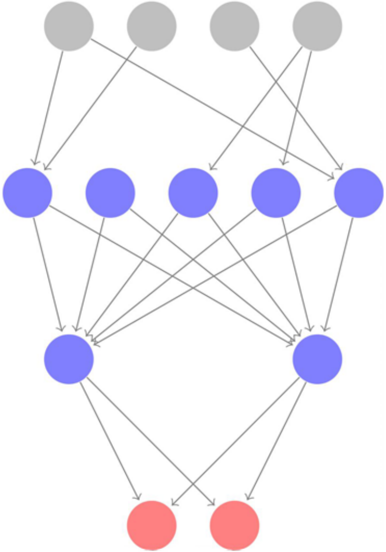
# Sparsity in Neural Networks

## Or: How I Learned To Stop Worrying and Love Retraining

Published in 11th International Conference on Learning Representations (ICLR23)

Max Zimmer - AI4Forest Kick-Off Meeting - Paris

November 2023



ZUSE INSTITUTE BERLIN

# Results are joint work of...

Max Zimmer
Zuse Institute Berlin

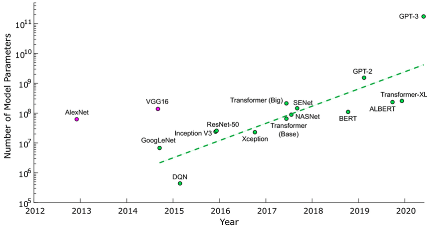
Christoph Spiegel
Zuse Institute Berlin

Sebastian Pokutta
Zuse Institute Berlin

*Research partially supported by the DFG Cluster of Excellence MATH+.*

# Why do we need sparsity?

- Neural Networks are exploding in size
- This yields several problems:
  - **Efficiency:** Longer training/inference times
  - **Storage:** Not deployable on phones, IOT, ...
  - **Costs:** Costly energy demands
    - Training of Large Language Models can emit as much $CO_2$ as five cars in their lifetime (Strubell et al., 2019)
    - GPT-3 Training: Estimated cost of 4.6 million USD




Source: Bernstein et al. (2021)

- One potential solution: **Pruning** - The removal of parameters of the network (LeCun et al., 1989; Han et al., 2015).
- Idea: introduce **sparsity** in the parameter tensors to reduce storage- and compute-demands

# How to decide what to prune?

Mathematical formulation: $\min_{\mathcal{W}} \mathcal{L}(\mathcal{W}, \mathcal{D})$ s.t. $\|\mathcal{W}\|_0 \leq k$. $\rightarrow$ Intractable!

Two different paradigms:

- Regularization: Force parameters towards zero throughout training (e.g. with penalities)
- Saliency criteria: Remove based on a heuristic, e.g. parameter magnitude.

A classical unstructured pruning approach:

**Iterative Magnitude Pruning** (IMP, Han et al., 2015);
**Input:** A pretrained network.
**repeat**
    PRUNE a fraction of the lowest-magnitude weights;
    RETRAIN the network for $T_{rt}$ epochs (How?);
**until** *the desired sparsity is reached*;

# Different sparsification paradigms

**Problem:** Pruning may improve generalization, but typically degrades model performance.

- Different paradigms to find well-performing sparse models:

### Pruning-instable approaches

- IMP-like three-stage approach: Pretrain, iteratively prune & retrain.
- Model performance drops when removing weights.

→ Retraining required.

### Pruning-stable approaches

- Require compute-intense regularization towards sparsity.
- Ultimate ‘hard’ pruning results in negligible performance degradation.

→ No retraining required.

# Disadvantages of IMP and our contribution

**(Claimed) Disadvantages compared to pruning-stable approaches:**

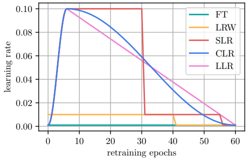
1. IMP is inferior to more complex algorithms that 'learn' the sparsity pattern throughout training and do not employ 'hard' pruning.
2. IMP is inefficient since it requires many retraining epochs. Pruning-stable approaches find a sparse solution throughout regular training.

**Central idea of our work:** These disadvantages fall apart when retraining properly.

# Existing retraining schedules

Let $(\eta_t)_{t \leq T}$ be the pretraining schedule and $T_{rt}$ the number of retraining epochs.

- **FT** (Han et al., 2015): Use the last learning rate $\eta_T$ for all epochs.
- **LRW** (Renda et al., 2020): Rewind the learning rate to epoch $T - T_{rt}$.
- **SLR** (Le and Hua, 2021): Schedule proportionally identical to original one.
- **CLR** (Le and Hua, 2021): 1-cycle cosine decay schedule.




These developments and improvements closely resemble the findings of Li et al. (2020) regarding optimal schedules in the budgeted setting.

# Our proposals:

- **Linear Learning Rate Restarting (LLR)**: Linear decay from $\eta_1$ to zero after a short warm-up phase.
- **Adaptive Linear Learning Rate Restarting (ALLR)**: Linear decay from $d \cdot \eta_1$, where discounting factor $d = \max(d_1, d_2) \in [0, 1]$ accounts for both the pruning-induced performance drop by $d_1 = \|\theta - \theta^p\|_2 / (\|\theta\|_2 \cdot \sqrt{s})$ and the retraining time by $d_2 = T_{rt}/T$, where $s$ is the target sparsity.

Table: ResNet-50 on ImageNet: Test accuracy comparison of the different learning rate translation schemes for One Shot IMP for retrain times of 2.22% (2 epochs), 5.55% (5 epochs) and 11.11% (10 epochs) of the initial training budget of 90 epochs.

| | **Model sparsity 70%** | | | **Model sparsity 80%** | | | **Model sparsity 90%** | | |
|---|---|---|---|---|---|---|---|---|---|
| Budget: | 2.22% | 5.55% | 11.11% | 2.22% | 5.55% | 11.11% | 2.22% | 5.55% | 11.11% |
| **FT** | **73.51 ±0.04** | 73.98 ±0.04 | 74.44 ±0.11 | 70.45 ±0.20 | 71.81 ±0.11 | 72.68 ±0.07 | 56.75 ±0.01 | 61.60 ±0.30 | 64.61 ±0.21 |
| **LRW** | 73.50 ±0.04 | **73.99 ±0.04** | **74.45 ±0.11** | 70.45 ±0.20 | 71.82 ±0.12 | 72.67 ±0.07 | 56.75 ±0.01 | 61.61 ±0.30 | 64.60 ±0.23 |
| **SLR** | 70.93 ±0.01 | 72.58 ±0.03 | 73.69 ±0.11 | 70.48 ±0.04 | 72.37 ±0.02 | 73.44 ±0.18 | 67.19 ±0.23 | 69.45 ±0.01 | 70.80 ±0.09 |
| **CLR** | 72.22 ±0.09 | 73.58 ±0.08 | **74.49 ±0.04** | **71.96 ±0.09** | **73.30 ±0.08** | **74.24 ±0.08** | **68.72 ±0.06** | **70.60 ±0.15** | **71.51 ±0.13** |
| **LLR (ours)** | **72.39 ±0.13** | **73.65 ±0.05** | 74.34 ±0.02 | **72.07 ±0.09** | **73.41 ±0.05** | **74.23 ±0.10** | **68.90 ±0.05** | **70.48 ±0.01** | **71.53 ±0.09** |
| **ALLR (ours)** | **73.69 ±0.03** | **74.37 ±0.05** | **74.89 ±0.04** | **72.96 ±0.15** | **74.02 ±0.08** | **74.71 ±0.04** | **69.56 ±0.07** | **71.19 ±0.01** | **71.99 ±0.07** |

# Budgeted IMP (BIMP)

Given a training budget of $T$ epochs, we propose BUDGETED IMP (BIMP), which:

- trains the network from scratch for some $T_0 < T$ epochs using a linear schedule,
- applies IMP with ALLR on the output for the remaining $T - T_0$ epochs.

BIMP maintains the key characteristics of IMP, i.e.,

- we prune 'hard' and do not allow weights to recover, and
- we do not impose any additional implicit bias during training.

Table: ResNet-50 on ImageNet: Comparison between BIMP and pruning-stable methods.

**ImageNet**

| | | Model sparsity 70% | | | Model sparsity 80% | | | Model sparsity 90% | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Method | # img/s | Accuracy | Speedup | Sparsity | Accuracy | Speedup | Sparsity | Accuracy | Speedup | Sparsity |
| **BIMP (ours)** | 1454 | **75.62 ±0.02** | 2 ±0.0 | 70.00 ±0.00 | **75.08 ±0.16** | 3 ±0.0 | 80.00 ±0.00 | **73.53 ±0.05** | 6 ±0.0 | 90.00 ±0.00 |
| **GMP** | 1425 | 74.62 ±0.08 | 2 ±0.0 | 70.00 ±0.00 | 74.19 ±0.17 | 4 ±0.0 | 80.00 ±0.00 | 72.80 ±0.03 | 7 ±0.1 | 90.00 ±0.00 |
| **GSM** | 1349 | 73.69 ±0.70 | 2 ±0.1 | 70.00 ±0.00 | 72.75 ±0.62 | 4 ±0.3 | 80.00 ±0.00 | 70.08 ±0.94 | 9 ±0.8 | 90.00 ±0.00 |
| **DPF** | 1456 | **75.59 ±0.07** | 2 ±0.0 | 70.00 ±0.00 | **75.30 ±0.02** | 3 ±0.0 | 80.00 ±0.00 | **74.05 ±0.05** | 6 ±0.0 | 90.00 ±0.00 |
| **DNW** | 530 | **75.60 ±0.01** | 2 ±0.0 | 70.00 ±0.00 | **75.27 ±0.01** | 3 ±0.0 | 80.00 ±0.00 | **74.29 ±0.03** | 5 ±0.1 | 90.00 ±0.00 |
| **LC** | 1436 | 75.03 ±0.20 | 2 ±0.0 | 70.00 ±0.00 | 73.87 ±0.62 | 3 ±0.0 | 80.00 ±0.00 | 67.57 ±2.71 | 5 ±0.0 | 90.00 ±0.00 |
| **STR** | 1396 | 70.66 ±0.13 | 3 ±0.0 | 75.34 ±0.01 | 70.70 ±0.13 | 4 ±0.0 | 80.93 ±0.00 | 70.13 ±0.01 | 8 ±0.0 | 90.00 ±0.00 |
| **DST** | 1219 | 74.63 ±0.22 | 4 ±0.1 | 70.00 ±0.00 | 73.16 ±0.11 | 6 ±0.1 | 80.00 ±0.00 | 71.35 ±0.09 | 13 ±0.4 | 90.00 ±0.00 |

# Conclusion

- Retraining is fundamentally about optimization; the learning rate is key.
- ALLR significantly improves upon previous approaches, often by a large margin.
- If proper care is taken of the learning rate, pruning-instable approaches such as (B)IMP are strong contenders, despite employing hard and heuristic pruning.
- Contrary to the existing narrative, retraining is not inherently bad.
- The focus should lie on understanding and improving the existing (simple) algorithms, instead of proposing more and more convoluted, compute-intense and hard-to-tune approaches.

Thank you for your attention!