Efficient Methods for NLP

Model Compression and Parameter-efficient Finetuning

Vladislav Mikhailov

Language Technology Group, University of Oslo



Table of contents



1. Background

2. Model Compression

3. Parameter-efficient Finetuning

4. References

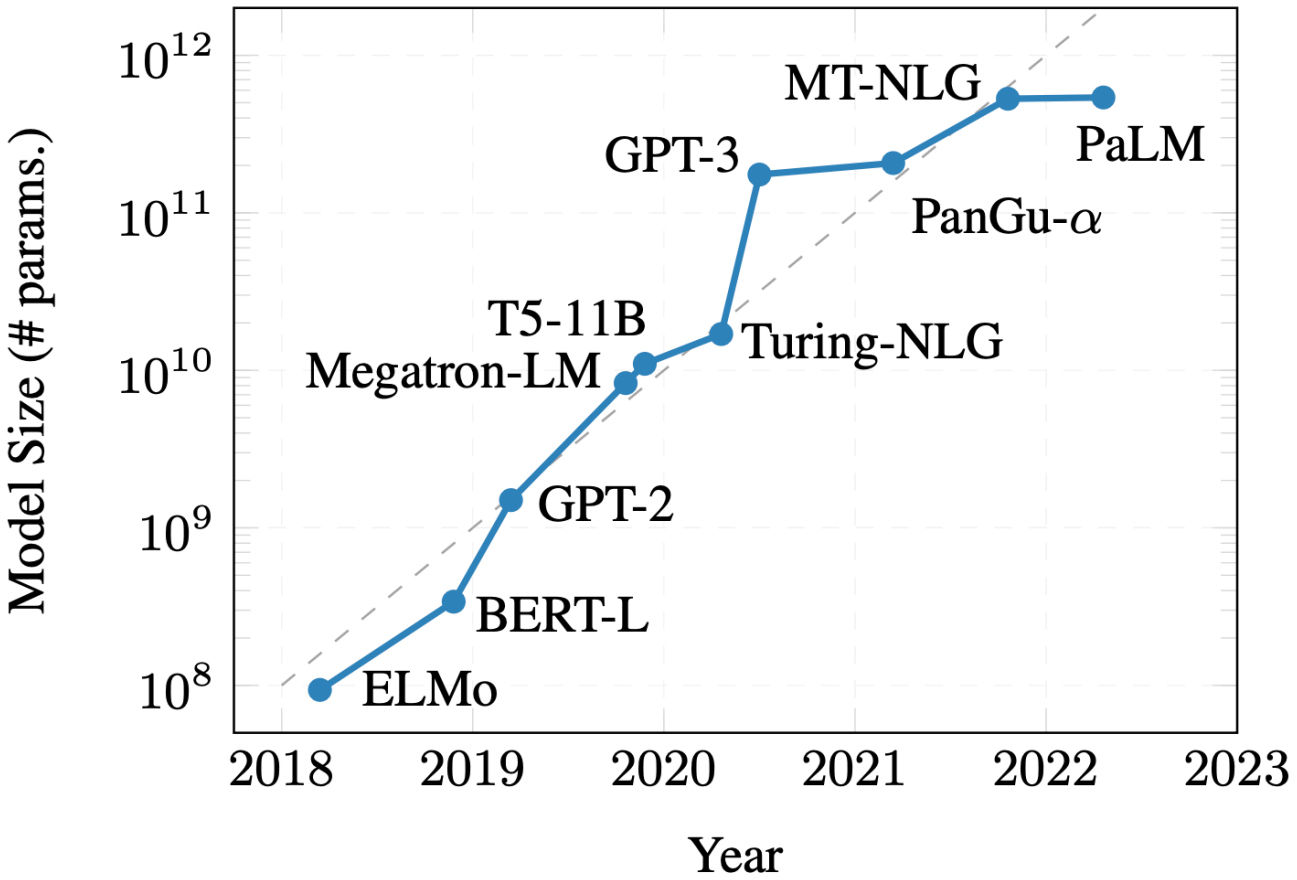


Figure 1: Growth in the number of parameters in LMs (Treviso et al., 2023).

Impacts of the computational race

- Scaling the number of parameters in LMs facilitates new state-of-the-art results, but:
  - significantly increases the carbon footprint (Bender et al., 2021);
  - facilitates the problem of unequal access to computational resources (Couldry and Mejias, 2020);
  - promotes difficulties using such models in academia and industry (Schick and Schütze, 2021).

How can LMs' efficiency be improved?

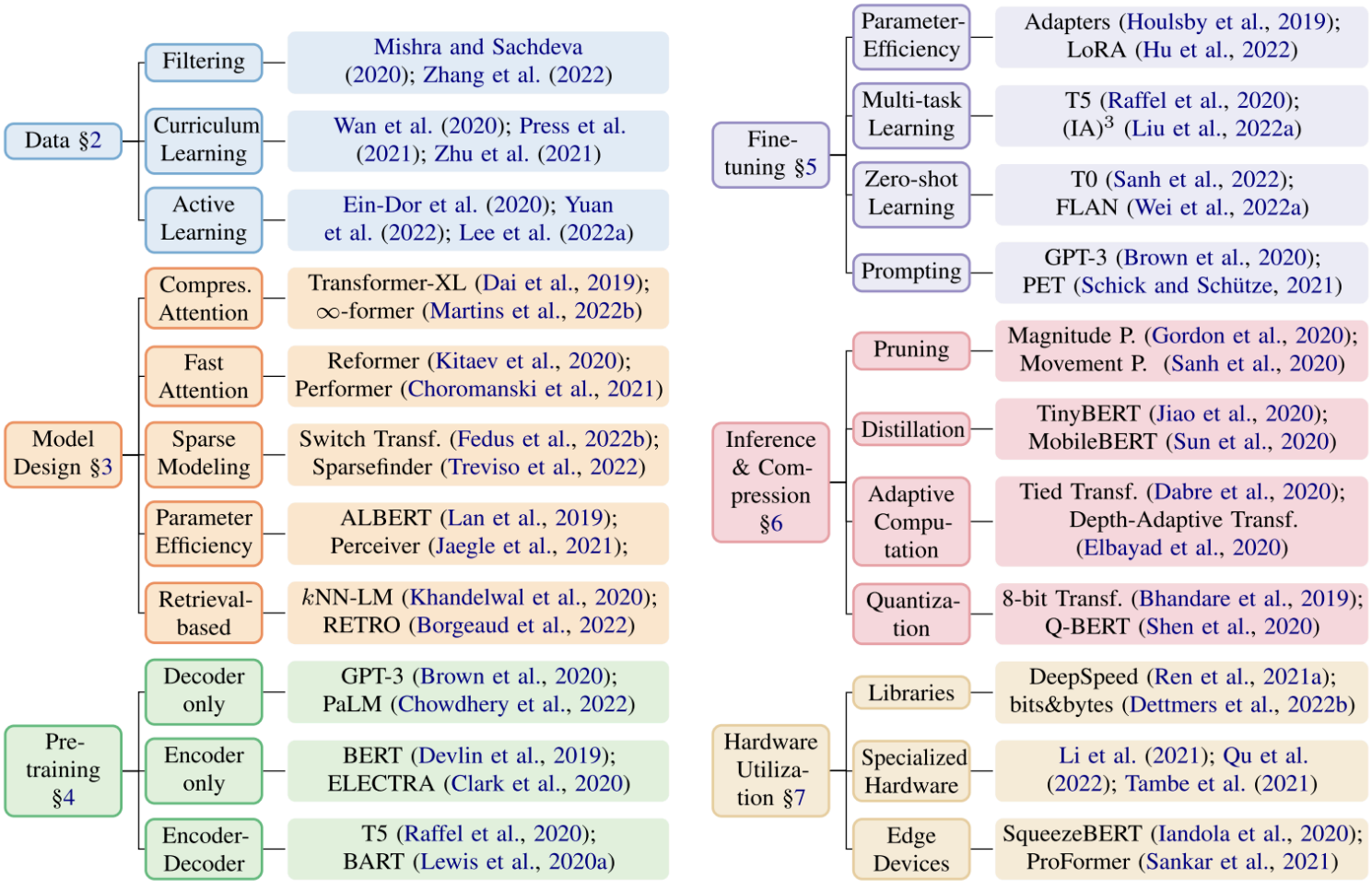


Figure 2: Typology of efficient methods for NLP (Treviso et al., 2023).

Table of contents



1. Background

2. Model Compression

3. Parameter-efficient Finetuning

4. References

Pruning

- Pruning aims to identify and remove redundant parameters.
  - **Unstructured pruning:** zeroing out parameters below a threshold;
  - **Structured pruning:** removing model components, e.g.:
    - attention heads (Voita et al., 2019; Jo and Myaeng, 2020);
    - hidden layers (Sajjad et al., 2023).

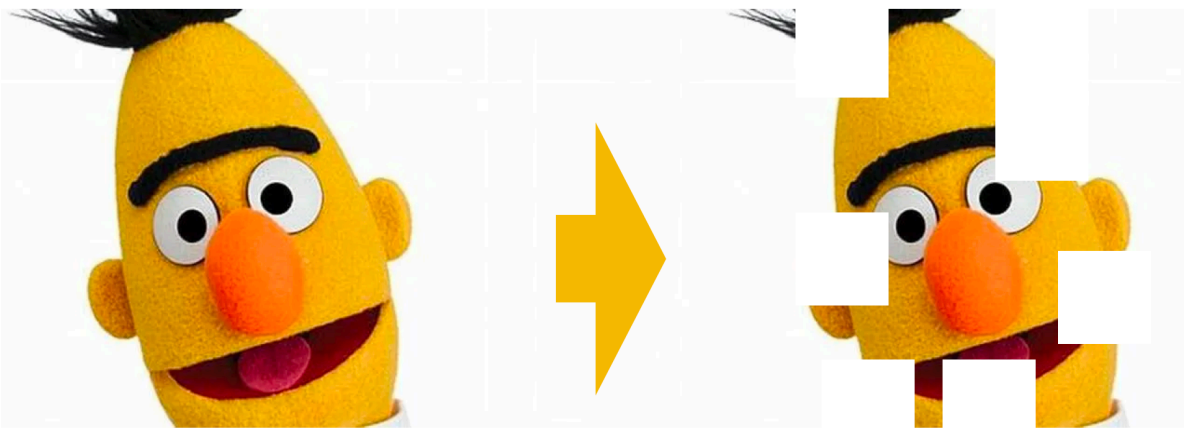


Figure 3: Illustration for pruning.

Key findings based on Treviso et al. (2023)

- Different pruning methods promote different benefits:
  - **Unstructured pruning:** may better preserve the model performance;
  - **Structured pruning:** may better improve the inference speed.
- Pruning during pretraining reduces the costs but increases the risk of over-pruning.
- Additional costs can relate to tuning the hyperparameters, such as the number of preserved weights.

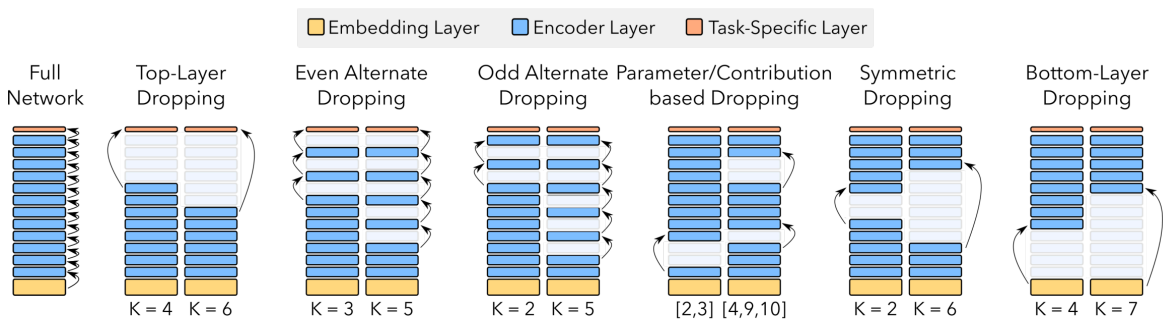


Figure 4: Examples of layer pruning strategies (Sajjad et al., 2023).

Knowledge Distillation

- Knowledge distillation aims to pretrain a smaller LM (a student) that mimics the behavior of the original LM (a teacher).
- The student has fewer weights and achieves a performance similar to that of the teacher.

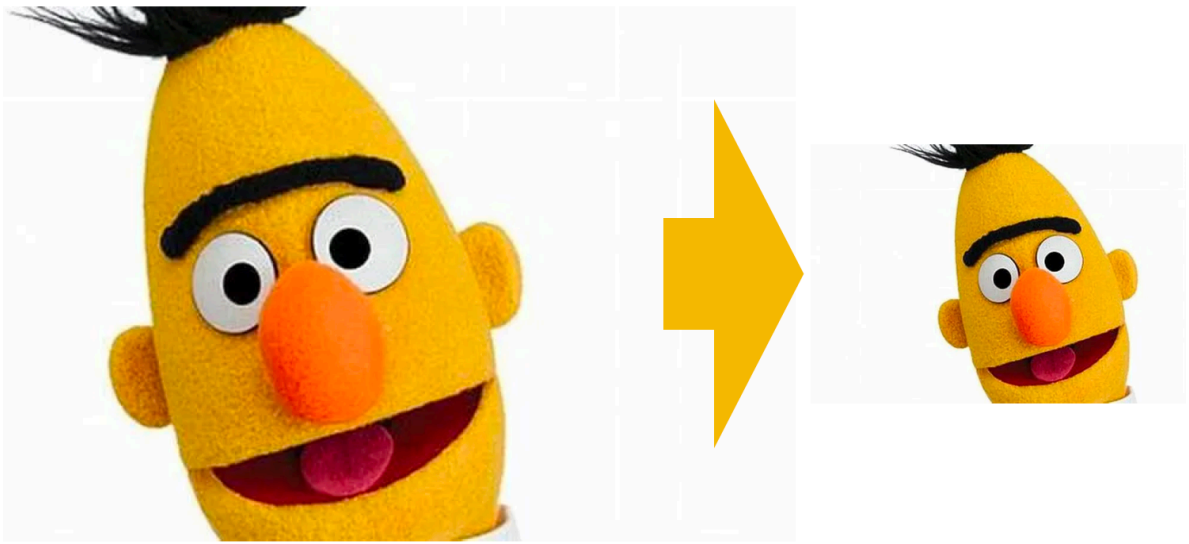


Figure 5: Illustration for knowledge distillation.

DistilBERT (Sanh et al., 2019)

- Initialized from BERT by taking one out of two layers.
- Pretrained using a linear combination of the distillation and masked language modeling losses.
- Compared to BERT (the teacher):
  - has 40% less parameters (66M vs 110M);
  - retains 97% of the language understanding performance on GLUE;
  - is 60% faster.

Model	**Score**	CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST-2	STS-B	WNLI
ELMo	68.7	44.1	68.6	76.6	71.1	86.2	53.4	91.5	70.4	56.3
BERT-base	79.5	56.3	86.7	88.6	91.8	89.6	69.3	92.7	89.0	53.5
DistilBERT	77.0	51.3	82.2	87.5	89.2	88.5	59.9	91.3	86.9	56.3

Figure 6: Performance of the BERT and DistilBERT models on GLUE.

Table of contents



1. Background

2. Model Compression

3. Parameter-efficient Finetuning

4. References

Overview

- Standard finetuning creates a *new* set of model parameters.
- Training a classification head with the model weights frozen usually leads to lower performance than standard finetuning.
- Main idea: train a small set of parameters which are either a subset of the model parameters or newly added ones.

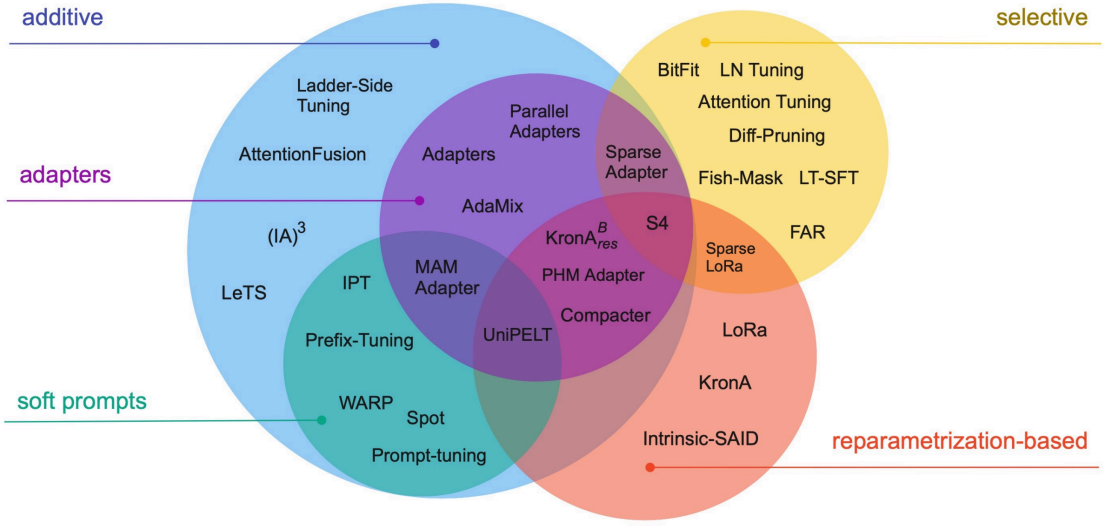


Figure 7: Parameter-efficient finetuning method taxonomy (Lialin et al., 2023).

Adapters

- Main idea: add fully-connected networks after attention and FFN layers in Transformer (Houlsby et al., 2019).
- The performance is competitive to standard finetuning by tuning less than 4% of the total model parameters.

```
def transformer_block_with_adapter(x):
    residual = x
    x = SelfAttention(x)
    x = FFN(x)  # adapter
    x = LN(x + residual)
    residual = x
    x = FFN(x)  # transformer FFN
    x = FFN(x)  # adapter
    x = LN(x + residual)
    return x
```

Figure 8: A schematic adapter implementation (Lialin et al., 2023).

Additive Methods: Soft Prompts (Prompt Tuning)

- Main idea: prepend the model input embeddings with a trainable tensor "soft prompt", which is optimized directly through gradient descent (Lester et al., 2021).
- The method is more parameter efficient the larger the model.

```
def soft_prompted_model(input_ids):
    x = Embed(input_ids)
    x = concat([soft_prompt, x], dim=seq)
    return model(x)
```

Figure 9: A schematic prompt tuning implementation (Lialin et al., 2023).

Additive Methods: Soft Prompts (Prefix Tuning)

- Main idea: trainable parameters are prepended to the hidden states of *all* layers (Li and Liang, 2021).
- In contrast to Lester et al. (2021), the "soft prompt" is optimized through the FFN.
- Training only 0.1% parameters is similar in performance to the standard BART finetuning.

```
def transformer_block_for_prefix_tuning(x):
    soft_prompt = FFN(soft_prompt)
    x = concat([soft_prompt, x], dim=seq)
    return transformer_block(x)
```

Figure 10: A schematic prompt tuning implementation (Lialin et al., 2023).

Reparametrization-based Methods: LoRA

- Main idea: reparametrizing the model weights using a low-rank transformation. Parameter update for a weight matrix is decomposed into a product of two low-rank matrices (Hu et al., 2021).
- The trainable matrices can be merged with the frozen weights, introducing no inference latency compared to standard LM finetuning.
- The right rank choice as a hyperparameter is important.

```
def lora_linear(x):
    h = x @ W # regular linear
    h += x @ W_A @ W_B # low-rank update
    return scale * h
```

Figure 11: A schematic LoRA implementation (Lialin et al., 2023).

Reparametrization-based Methods: LoRA

- Main idea: reparametrizing the model weights using a low-rank transformation. Parameter update for a weight matrix is decomposed into a product of two low-rank matrices (Hu et al., 2021).
- The trainable matrices can be merged with the frozen weights, introducing no inference latency compared to standard LM finetuning.
- The right rank choice as a hyperparameter is important.

```
def lora_linear(x):
    h = x @ W # regular linear
    h += x @ W_A @ W_B # low-rank update
    return scale * h
```

Figure 12: A schematic LoRA implementation (Lialin et al., 2023).



Reparametrization-based Methods: LoRA

Model & Method	# Trainable Parameters	MNLI	SST-2	MRPC	CoLA	QNLI	QQP	RTE	STS-B	Avg.
RoB_{base} (FT)*	125.0M	**87.6**	94.8	90.2	**63.6**	92.8	**91.9**	78.7	91.2	86.4
RoB_{base} (BitFit)*	0.1M	84.7	93.7	**92.7**	62.0	91.8	84.0	81.5	90.8	85.2
RoB_{base} (Adpt^{D})*	0.3M	$87.1_{\pm.0}$	$94.2_{\pm.1}$	$88.5_{\pm1.1}$	$60.8_{\pm.4}$	$93.1_{\pm.1}$	$90.2_{\pm.0}$	$71.5_{\pm2.7}$	$89.7_{\pm.3}$	84.4
RoB_{base} (Adpt^{D})*	0.9M	$87.3_{\pm.1}$	$94.7_{\pm.3}$	$88.4_{\pm.1}$	$62.6_{\pm.9}$	$93.0_{\pm.2}$	$90.6_{\pm.0}$	$75.9_{\pm2.2}$	$90.3_{\pm.1}$	85.4
RoB_{base} (LoRA)	0.3M	$87.5_{\pm.3}$	$\mathbf{95.1}_{\pm.2}$	$89.7_{\pm.7}$	$63.4_{\pm1.2}$	$\mathbf{93.3}_{\pm.3}$	$90.8_{\pm.1}$	$\mathbf{86.6}_{\pm.7}$	$\mathbf{91.5}_{\pm.2}$	**87.2**
$\text{RoB}_{\text{large}}$ (FT)*	355.0M	90.2	**96.4**	**90.9**	68.0	94.7	**92.2**	86.6	92.4	88.9
$\text{RoB}_{\text{large}}$ (LoRA)	0.8M	$\mathbf{90.6}_{\pm.2}$	$96.2_{\pm.5}$	$\mathbf{90.9}_{\pm1.2}$	$\mathbf{68.2}_{\pm1.9}$	$\mathbf{94.9}_{\pm.3}$	$91.6_{\pm.1}$	$\mathbf{87.4}_{\pm2.5}$	$\mathbf{92.6}_{\pm.2}$	**89.0**
$\text{RoB}_{\text{large}}$ (Adpt^{P})†	3.0M	$90.2_{\pm.3}$	$96.1_{\pm.3}$	$90.2_{\pm.7}$	$\mathbf{68.3}_{\pm1.0}$	$\mathbf{94.8}_{\pm.2}$	$\mathbf{91.9}_{\pm.1}$	$83.8_{\pm2.9}$	$92.1_{\pm.7}$	88.4
$\text{RoB}_{\text{large}}$ (Adpt^{P})†	0.8M	$\mathbf{90.5}_{\pm.3}$	$\mathbf{96.6}_{\pm.2}$	$89.7_{\pm1.2}$	$67.8_{\pm2.5}$	$\mathbf{94.8}_{\pm.3}$	$91.7_{\pm.2}$	$80.1_{\pm2.9}$	$91.9_{\pm.4}$	87.9
$\text{RoB}_{\text{large}}$ (Adpt^{H})†	6.0M	$89.9_{\pm.5}$	$96.2_{\pm.3}$	$88.7_{\pm2.9}$	$66.5_{\pm4.4}$	$94.7_{\pm.2}$	$92.1_{\pm.1}$	$83.4_{\pm1.1}$	$91.0_{\pm1.7}$	87.8
$\text{RoB}_{\text{large}}$ (Adpt^{H})†	0.8M	$90.3_{\pm.3}$	$96.3_{\pm.5}$	$87.7_{\pm1.7}$	$66.3_{\pm2.0}$	$94.7_{\pm.2}$	$91.5_{\pm.1}$	$72.9_{\pm2.9}$	$91.5_{\pm.5}$	86.4
$\text{RoB}_{\text{large}}$ (LoRA)†	0.8M	$\mathbf{90.6}_{\pm.2}$	$96.2_{\pm.5}$	$\mathbf{90.2}_{\pm1.0}$	$68.2_{\pm1.9}$	$\mathbf{94.8}_{\pm.3}$	$91.6_{\pm.2}$	$\mathbf{85.2}_{\pm1.1}$	$\mathbf{92.3}_{\pm.5}$	**88.6**
DeB_{XXL} (FT)*	1500.0M	91.8	**97.2**	92.0	72.0	**96.0**	92.7	93.9	92.9	91.1
DeB_{XXL} (LoRA)	4.7M	$\mathbf{91.9}_{\pm.2}$	$96.9_{\pm.2}$	$\mathbf{92.6}_{\pm.6}$	$\mathbf{72.4}_{\pm1.1}$	$\mathbf{96.0}_{\pm.1}$	$\mathbf{92.9}_{\pm.1}$	$\mathbf{94.9}_{\pm.4}$	$\mathbf{93.0}_{\pm.2}$	**91.3**

Figure 13: Evaluation of parameter-efficient methods on GLUE (Hu et al., 2021).



Summary

- One can efficiently build small task-specific modules using a frozen pretrained LM.
- The open challenge is finding trade-offs between end-task performance and resource consumption.
- Hyperparameter choices play a crucial role in the model performance and may not be transferable between different tasks.

Table of contents



1. Background

2. Model Compression

3. Parameter-efficient Finetuning

4. References

References I



Bender, E. M., Gebru, T., McMillan-Major, A., and Shmitchell, S. (2021). On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, pages 610–623.

Couldry, N. and Mejias, U. A. (2020). *The costs of connection: How data is colonizing human life and appropriating it for capitalism*. Stanford University Press.

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S. (2019). Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR.

Hu, E. J., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W., et al. (2021). Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Jo, J.-y. and Myaeng, S.-H. (2020). Roles and utilization of attention heads in transformer-based neural language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3404–3417, Online. Association for Computational Linguistics.

Lester, B., Al-Rfou, R., and Constant, N. (2021). The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

References II



Li, X. L. and Liang, P. (2021). Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.

Lialin, V., Deshpande, V., and Rumshisky, A. (2023). Scaling down to scale up: A guide to parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.15647*.

Sajjad, H., Dalvi, F., Durrani, N., and Nakov, P. (2023). On the effect of dropping layers of pre-trained transformer models. *Computer Speech & Language*, 77:101429.

Sanh, V., Debut, L., Chaumond, J., and Wolf, T. (2019). Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Schick, T. and Schütze, H. (2021). It's not just size that matters: Small language models are also few-shot learners. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2339–2352, Online. Association for Computational Linguistics.

Treviso, M., Lee, J.-U., Ji, T., van Aken, B., Cao, Q., Ciosici, M. R., Hassid, M., Heafield, K., Hooker, S., Raffel, C., Martins, P. H., Martins, A. F. T., Forde, J. Z., Milder, P., Simpson, E., Slonim, N., Dodge, J., Strubell, E., Balasubramanian, N., Derczynski, L., Gurevych, I., and Schwartz, R. (2023). Efficient methods for natural language processing: A survey. *Transactions of the Association for Computational Linguistics*, 11:826–860.

References III

Voita, E., Talbot, D., Moiseev, F., Sennrich, R., and Titov, I. (2019). Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, Florence, Italy. Association for Computational Linguistics.