UC San Diego
JACOBS SCHOOL OF ENGINEERING

Scan me

Generalizable and Stable Finetuning of Pretrained Language Models on Low-Resource Texts

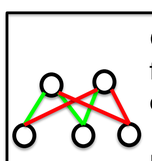
Sai Ashish Somayajula, Youwei Liang, Li Zhang, Abhishek Singh, Pengtao Xie
{ssomayaj, youwei, p1xie}@ucsd.edu

Motivation

- Pretrained language models (PLMs) have significantly improved the performance on various NLP tasks
- Finetuning PLMs on low-resource texts poses significant challenges
  - High variance in performance for different final layer weight initializations
  - Prone to overfitting leading to poor generalization on test set

Prior Methods

- Finetuning only a sub-network chosen based on empirical Fisher Information matrix (FIM)
  - Child-Tuning$_D$, DPS Dense
  - Promising direction with improved results



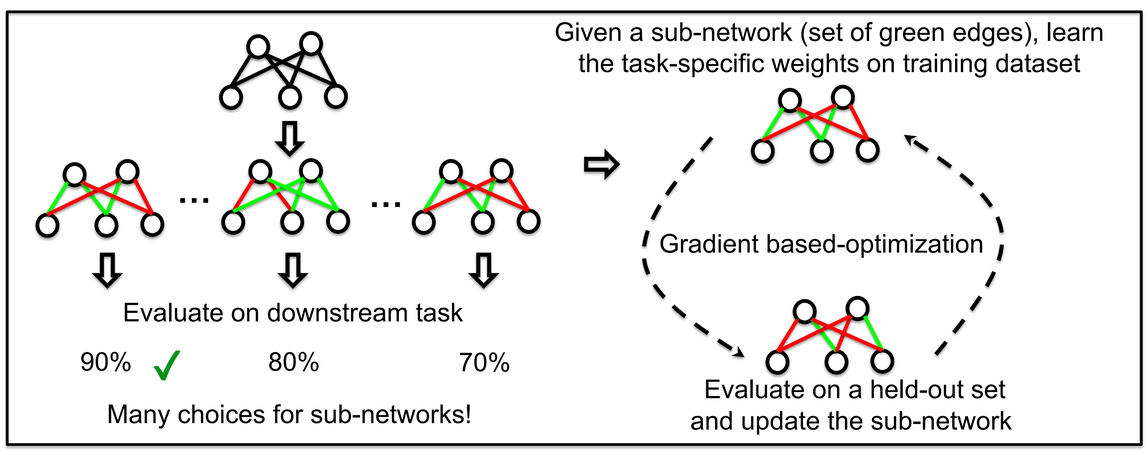
Green edges: Weights finetuned on the task, estimated using FIM

Red edges: Frozen pretrained weights

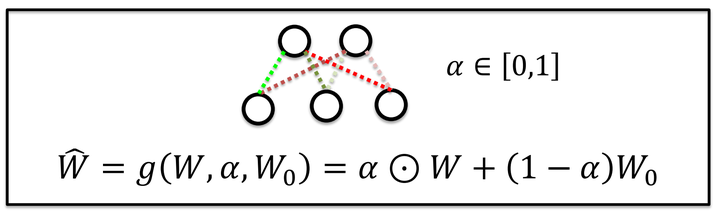
- Limitation of empirical FIM in low-resource scenarios
  - Data scarcity can skew the gradients used to compute the FIM
  - Theoretical results claiming empirically determined FIM deviates significantly from the true FIM when sample size is low

Proposed Method

- Deviate from FIM-based sub-network selection
- Downstream task performance guided sub-network selection



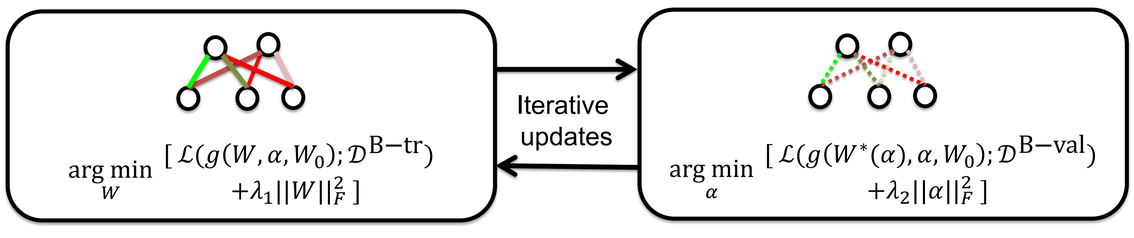
Attention-Guided Weights Mixup



- Linear interpolation of pretrained weight W_0 and task-specific weight W
- α the 'attention parameters', determine the chosen sub-network
  - $\alpha = 1$: Weight in sub-network
  - $\alpha = 0$: Frozen pretrained weight
- $\alpha \in [0,1]$ allowing a transition to continuous sub-network selection
  - α closer to 0: Greater influence from pretrained weight and vice-versa
  - Edges are depicted in shades of red and green

Bi-Level Optimization (BLO) Framework

- Learn α and W, which are interdependent, to improve downstream task performance
  - Stage 1: Given a sub-network determined by α, learn $W^*(\alpha)$
  - Stage 2: Evaluate learned network determined by $W^*(\alpha)$, learn α^*



Stage 1: Learn W on the train split

Stage 2: Evaluate on validation split and update α

$$\begin{aligned} \min_{\alpha} \quad & \mathcal{L}(g(W^*(\alpha), \alpha, W_0); \mathcal{D}^{\mathrm{B-val}}) + \lambda_2 ||\alpha||_F^2 \\ \text{s.t.} \quad & W^*(\alpha) = \arg\min_W \mathcal{L}(g(W, \alpha, W_0); \mathcal{D}^{\mathrm{B-tr}}) + \lambda_1 ||W||_F^2 \end{aligned}$$

Optimization Algorithm

- Use one-step gradient descent and finite difference approximation

$$W^*(\alpha) \approx W' = W - \eta_w \nabla_W [\mathcal{L}(g(W, \alpha, W_0); \mathcal{D}^{\mathrm{B-tr}}) + \lambda_1 ||W||_F^2]$$

$$\alpha^* \approx \alpha' = \alpha - \eta_\alpha \nabla_\alpha [\mathcal{L}(g(W', \alpha, W_0); \mathcal{D}^{\mathrm{B-val}}) + \lambda_2 ||\alpha||_F^2]$$

Experimental Results

Training split	Vanilla	CHILD-TUNING$_D$	DPS Dense	Ours
300	62.54 ± 6.57	62.47 ± 5.5	61.69 ± 5.62	**68.97** ± 3.09
500	65.85 ± 4.57	68.35 ± 4.36	68.99 ± 2.92	**72.42** ± 2.14
1000	73.19 ± 2.62	74.07 ± 2.75	75.00 ± 1.61	**76.68** ± 1.58

Table 1: Comparison of our method with vanilla, and prior FIM-based methods using BERT large across 300, 500, and 1000 training data splits: Averaged evaluation metrics over eight GLUE datasets

Models	Methods	CoLA		MRPC		RTE		STSB		Average	
		Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
BERT	Vanilla	64.11	1.33	90.80	1.77	70.69	2.83	89.92	0.61	78.88	1.64
	Ours	**66.07**	1.35	**91.84**	0.37	**73.43**	1.52	**90.34**	0.48	**80.42**(+1.54)	0.93(-0.71)
BART	Vanilla	58.54	1.41	92.03	0.73	81.84	1.41	91.54	0.40	80.99	0.99
	Ours	**60.15**	0.81	**92.33**	0.40	**84.26**	0.54	**92.20**	0.09	**82.23**(+1.24)	0.46(-0.53)
RoBERTa	Vanilla	66.06	2.07	92.25	0.57	78.52	13.01	91.89	0.31	82.18	3.99
	Ours	**66.52**	1.45	**92.58**	0.48	**84.22**	1.44	**92.21**	0.08	**83.88**(+1.70)	0.86(-3.13)
DeBERTa	Vanilla	63.74	1.34	92.31	0.37	85.59	1.58	91.74	0.17	83.34	0.86
	Ours	**65.96**	1.15	**92.32**	0.28	**86.17**	1.47	**91.99**	0.15	**84.11**(+0.77)	0.76(-0.10)
XLNet	Vanilla	40.93	27.28	91.83	0.91	71.17	14.40	91.68	0.19	73.90	10.69
	Ours	**61.66**	1.95	**92.19**	0.38	**83.54**	1.44	**92.12**	0.08	**82.38**(+8.48)	0.96(-9.73)

Table 2: Comparison of our method and vanilla finetuning on five PLMs over ten runs. Underlined values indicate occurrence of degenerate seeds

Methods	CoLA		MRPC		RTE		STSB		Average	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
Vanilla	64.11	1.33	90.80	1.77	70.69	2.83	89.92	0.61	78.88	1.64
Mixout	64.42	1.51	91.31	1.08	72.05	1.67	90.39	0.57	79.54	1.21
R3F	64.62	1.38	91.63	0.93	70.75	1.76	89.92	0.61	79.23	1.17
R-Dropout	64.14	1.58	**91.87**	0.78	70.24	2.83	90.25	0.49	79.13	1.42
CHILD-TUNING$_D$	64.85	1.32	91.52	0.81	71.69	1.95	90.42	0.44	79.62	1.13
Re-init	64.24	2.03	91.61	0.80	72.44	1.74	**90.71**	0.14	79.75	1.18
DPS Dense	64.98	1.08	91.50	0.83	73.14	1.97	90.51	0.55	80.03	1.11
DPS Dense (Our run)	64.08	1.50	90.25	2.21	71.92	1.45	90.20	0.47	79.11	1.41
Ours	**66.07**	1.35	91.84	0.37	**73.43**	1.52	90.34	0.48	**80.42**	0.93

Table 3: Comparison of our method with other regularization-based approaches on four small datasets (CoLA, RTE, STSB, and MRPC) known for causing instability in BERT large models. Mean and standard deviation over ten runs reported

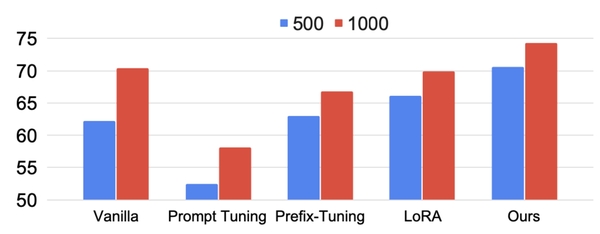


Figure1: Averaged performance across CoLA, RTE, STSB, and MRPC for vanilla, prompt tuning, prefix-tuning, LoRA, and our method using BERT large

Method	Mean	Std
Ours	**80.42**	0.93
Vanilla	78.88	1.64
Joint Training	78.86	1.48
Random$_\alpha$		
$\sigma_\alpha = 0.005$	79.36	1.03
$\sigma_\alpha = 0.1$	78.32	2.27
$\sigma_\alpha = 0.45$	69.29	5.44

Table 4: Ablation studies to understand the impact of using BLO and learning α

Conclusions

- Propose an attention-guided weights mixup strategy
- Introduce BLO framework to optimize both task weights (W) and attention parameters (α) using different splits of training data
- Demonstrate superior performance over various baselines in low-resource settings
- Improved stability across various PLM architectures
- Potential applications in continual learning