The Power of Scale for Parameter-Efficient Prompt Tuning

Brian Lester*   Rami Al-Rfou   Noah Constant
Google Research
{brianlester,rmyeid,nconstant}@google.com

Abstract

- Prompt Tuning
  - In Downstream Task, Learning "Soft Prompt" + Frozen Language Models
  - more competitive with scale
  - benefits in robustness to domain transfer
  - efficient prompt ensembling

Introduction

- How to adapt general-purpose models to downstream tasks?
  - Model tuning (fine-tuning)
    - all model parameters are tuned during adaptation
  - Prompt design (priming)
    - prompts: a task description and/or several examples
    - freezing pre-trained models
    - make a single generalist model simultaneously serve many different tasks

Introduction

- Key drawbacks of prompt-based adaptation
  - Task description is error-prone and requires human involvement
  - The effectiveness depends on the prompt quality
  - Downstream task quality is lower than tuned models

- Solution: Prompt Tuning!

Prompt Tuning

- Conditional Text Generation of T5 (Classification)

$$\Pr_\theta(Y|X)$$

  - X is a series of tokens, Y is a sequence of tokens that represent a class label
  - θ is the weights of the transformers that make up its encoder and decoder

Prompt Tuning

- Prompting is the approach of adding extra information

$$\Pr_\theta(Y|[P;X])$$

  - P is a series of tokens prepended (prompts)
    - $P = \{p_1, p_2, ..., p_n\}$
  - The model maximizes the likelihood of the correct Y, while keeping the model parameters θ (fixed)

Prompt Tuning

- Prompt Design
  - selecting prompt tokens from a fixed vocabulary of frozen embeddings
- Prompt Tuning
  - using a fixed prompt of special tokens,
    where only the embeddings of these prompt tokens **can be updated**

Prompt Tuning

- New conditional generation model

$$\Pr_{\theta;\theta_P}(Y|[P;X])$$

  - The model maximizes the likelihood of Y via backpropagation
  - Only applying gradient updates to θ_P

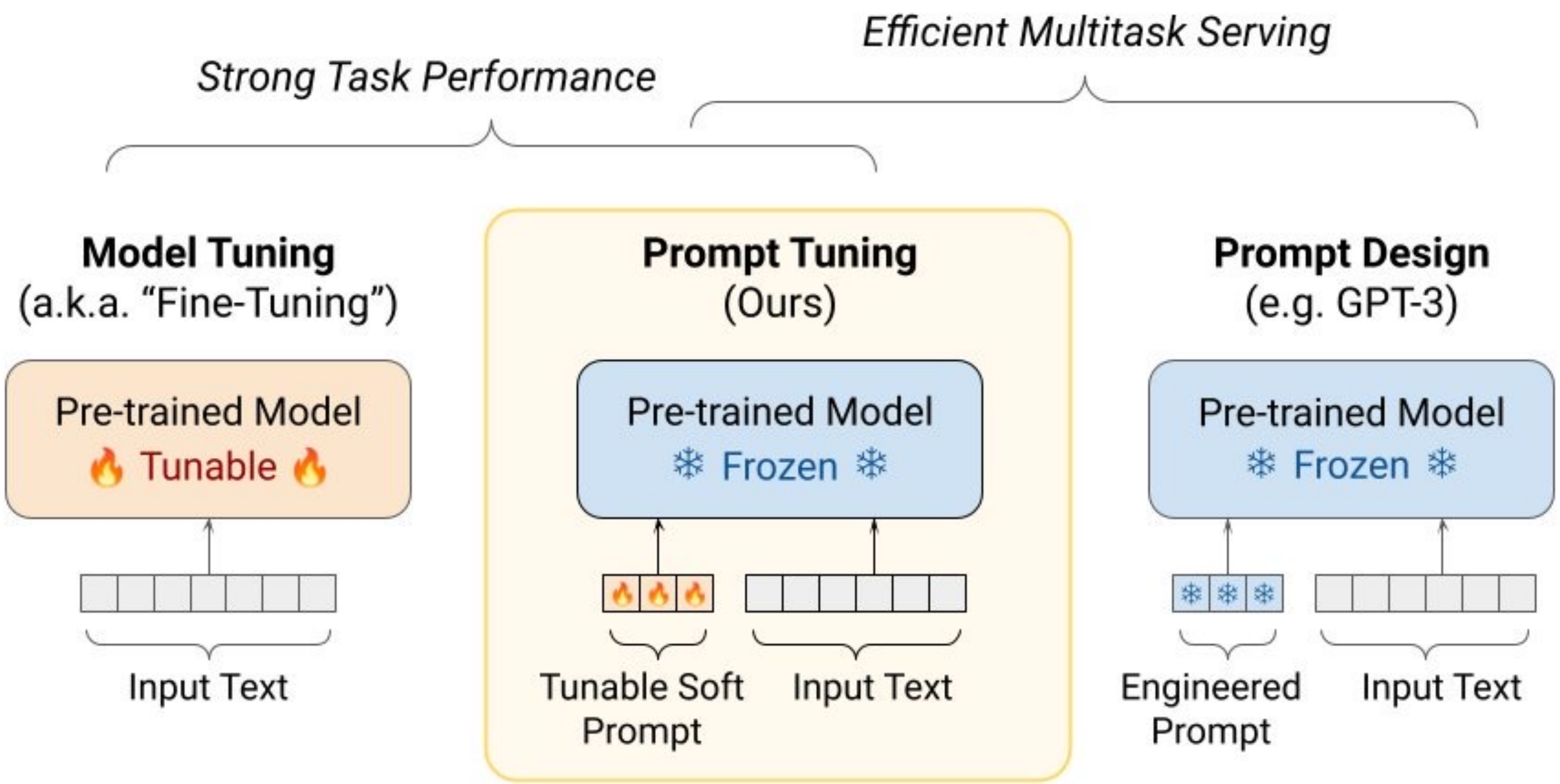
Prompt Tuning

- New conditional generation model

$$\Pr_{\theta;\theta_P}(Y|[P;X])$$

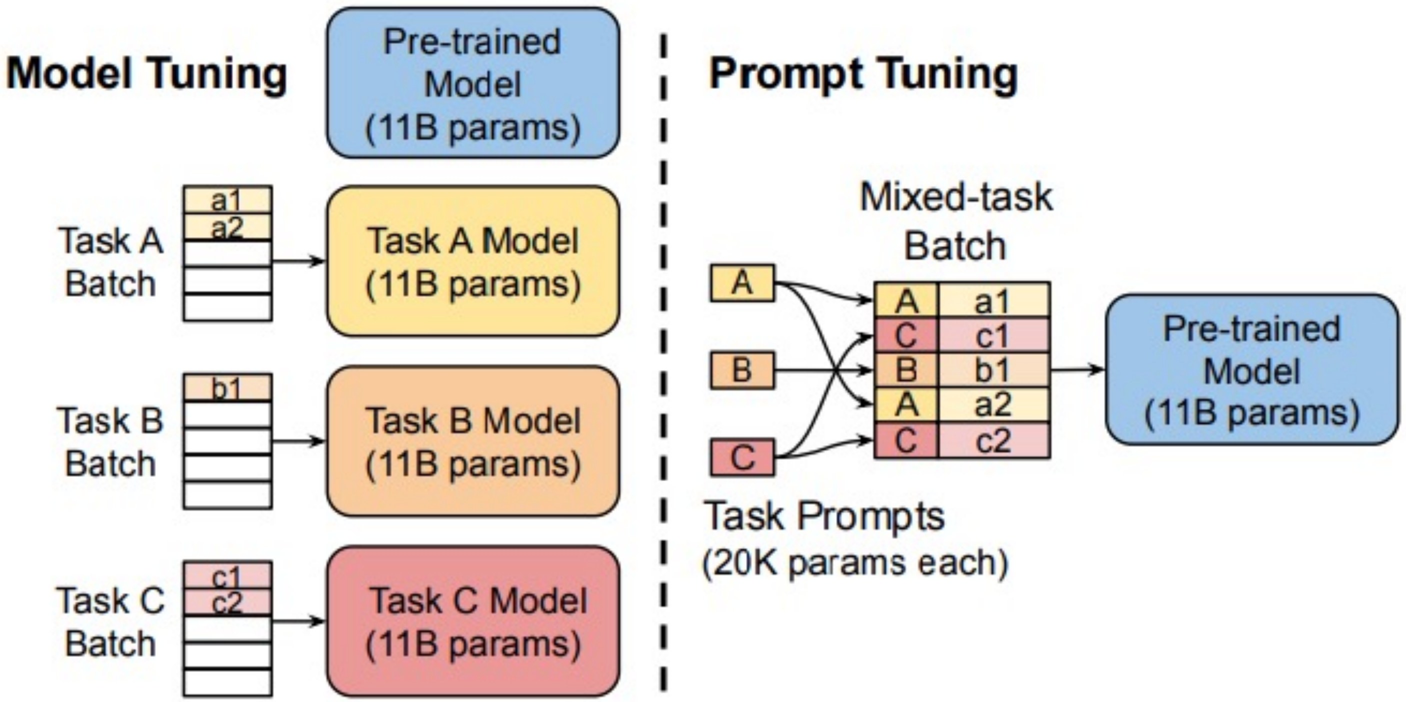
  - Given a series of n tokens, $\{x_1, x_2, \ldots, x_n\}$,
    T5 embed the tokens forming a matrix $X_e \in \mathbb{R}^{n \times e}$
    - e is the dimension of the embedding space
  - soft-prompts are represented as a parameter $P_e \in \mathbb{R}^{p \times e}$
  - Then, our prompt is $[P_e; X_e] \in \mathbb{R}^{(p+n) \times e}$
    - only the prompt parameters P_e are updated

Prompt Tuning



Reference: Google AI

Prompt Tuning



Prompt Tuning

- Design Decisions
  - How to initialize the prompt representations?
    - random initialization
    - initialize each prompt token to an embedding drawn from the model's vocab
    - For classification tasks,
      initialize the prompt with embeddings that enumerate the output classes
  - The length of the prompt
    - aim to find a minimal length that still performs well

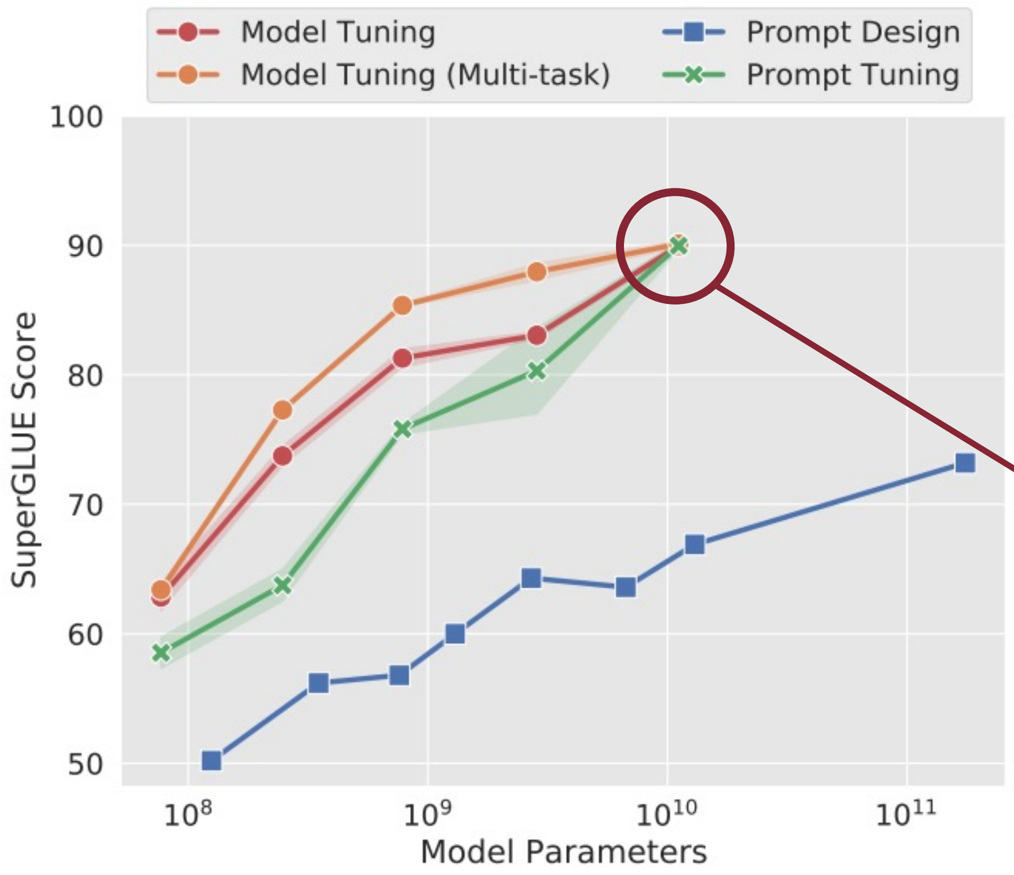
Prompt Tuning

- Unlearning Span Corruption
  - T5 is tasked with "reconstructing" masked spans in the input text
  - This setup is not a good fit for prompt tuning
  - Example)
    - Text: "Thank you for inviting me to your party last week"
    - Input: "Thank you <X> me to your party <Y> week"
    - Output: "<X> for inviting <Y> last <Z>"

Prompt Tuning

- Unlearning Span Corruption
  - Span Corruption
    - just use pre-trained T5 off-the-shelf as our frozen model
  - Span Corruption + Sentinel
    - use the same model, but prepend all downstream targets with a sentinel
  - LM Adaptation
    - continue T5's self-supervised training for a small number of additional steps
    - natural text prefix -> natural text continuation
    - quickly transform T5 into a model more similar to GPT-3

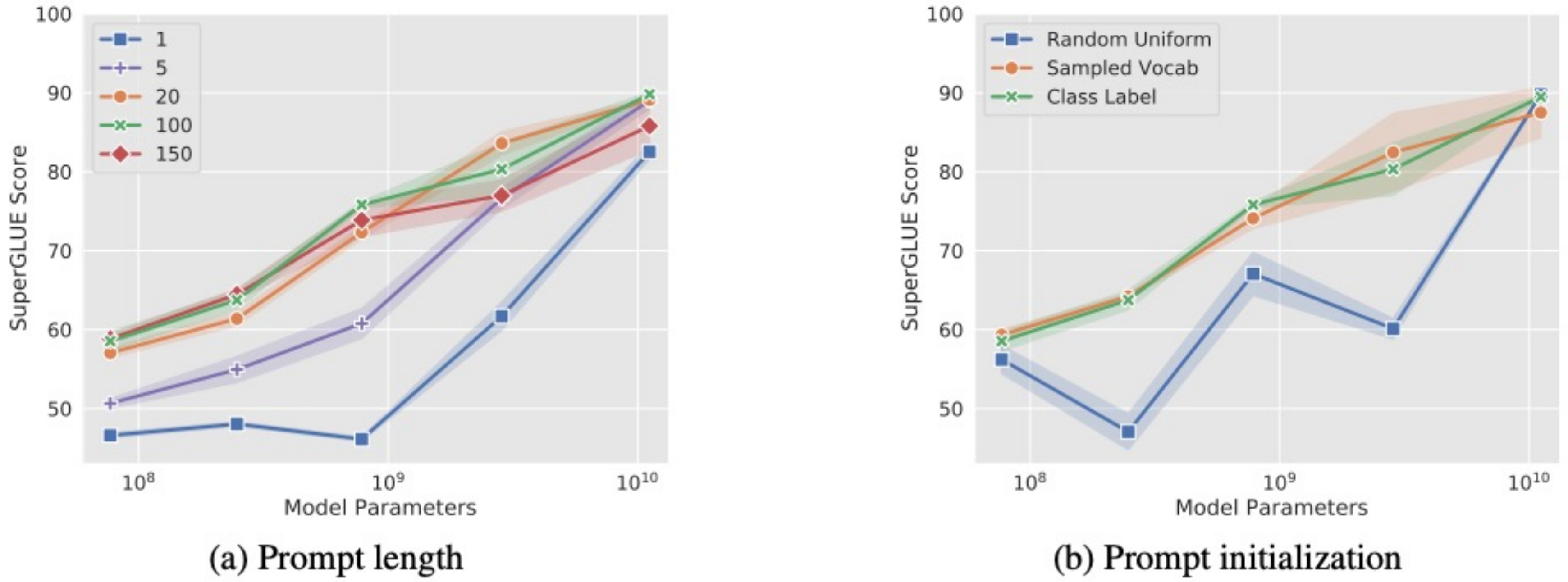
Results



- Default configuration
  - LM-adapted version of T5
  - initialize using class labels
  - use a prompt length of 100 tokens
- SuperGLUE benchmark
- Prompt Tuning matches the baselines

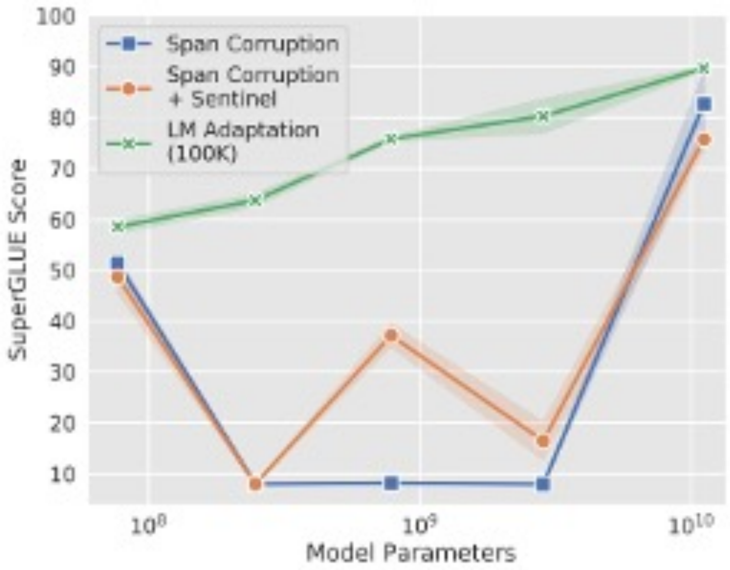
Results

- Ablation Study

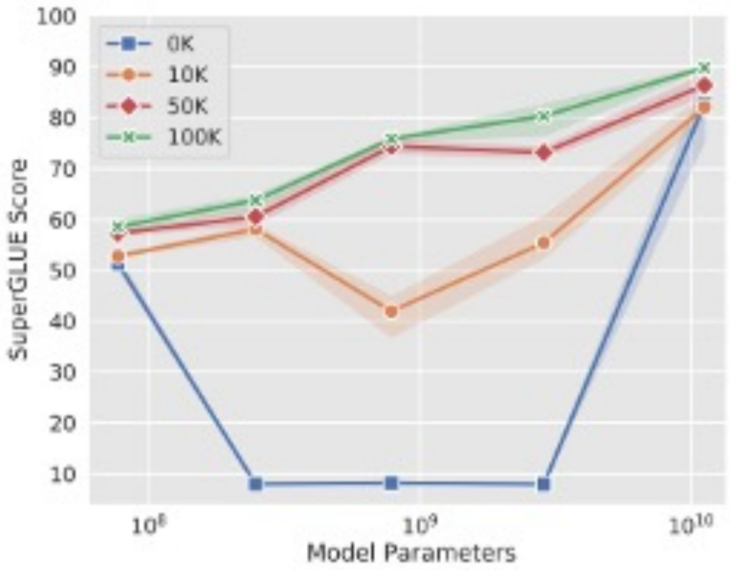


Results

- Ablation Study

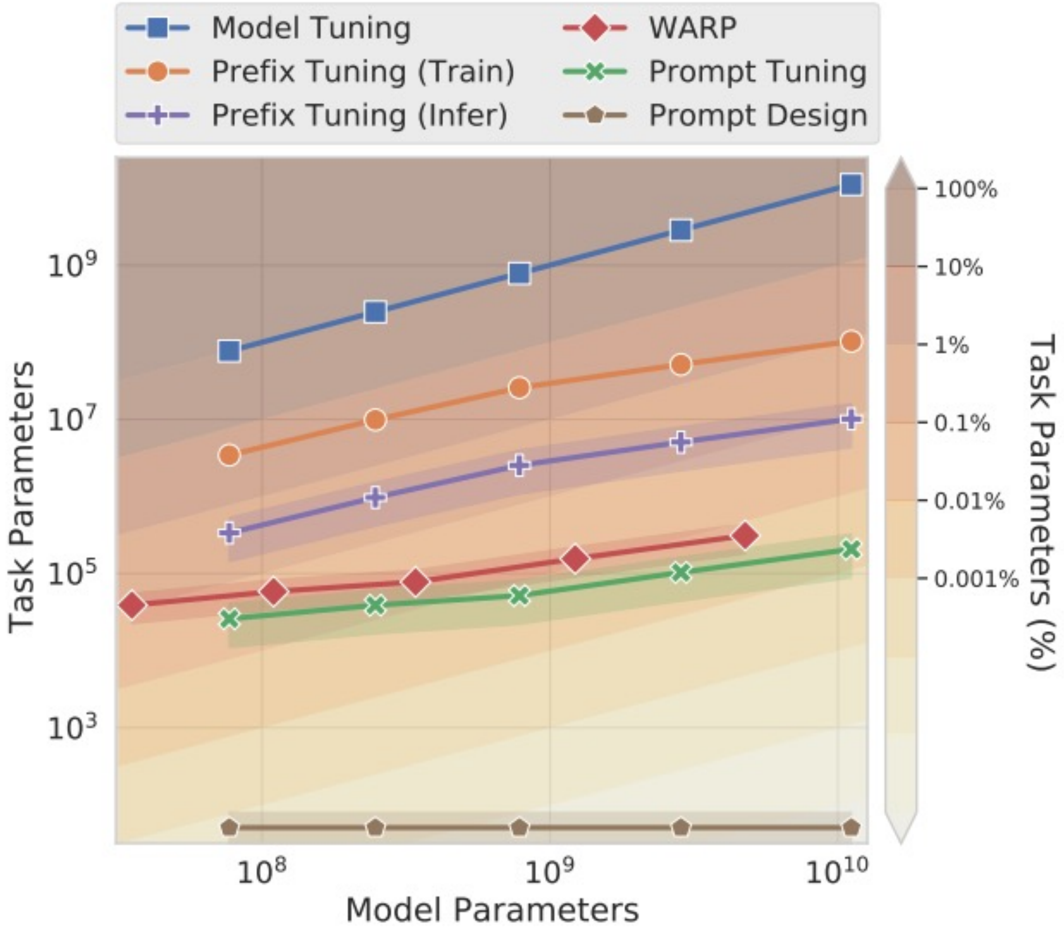


(c) Pre-training method



(d) LM adaptation steps

Comparison to Similar Approaches



- Prompt Tuning is
  the most parameter-efficient

Resilience to Domain Shift

- Prompt tuning prevents the model from modifying its general understanding
- This suggests that prompt tuning may improve robustness to domain shifts
- Zero-shot domain transfer: QA, paraphrase detection

Dataset	Domain	Model	Prompt	Δ
SQuAD	Wiki	94.9 ±0.2	94.8 ±0.1	−0.1
TextbookQA	Book	54.3 ±3.7	**66.8** ±2.9	+12.5
BioASQ	Bio	77.9 ±0.4	**79.1** ±0.3	+1.2
RACE	Exam	59.8 ±0.6	**60.7** ±0.5	+0.9
RE	Wiki	88.4 ±0.1	**88.8** ±0.2	+0.4
DuoRC	Movie	**68.9** ±0.7	67.7 ±1.1	−1.2
DROP	Wiki	**68.9** ±1.7	67.1 ±1.9	−1.8

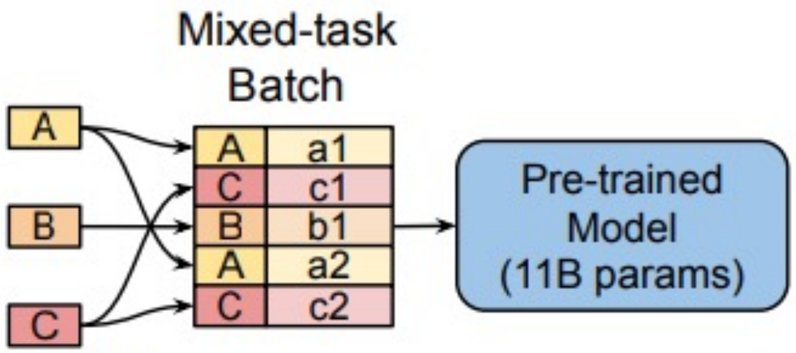
Table 1: F1 mean and stddev for models trained on SQuAD and evaluated on out-of-domain datasets from the MRQA 2019 shared task. Prompt tuning tends to give stronger zero-shot performance than model tuning, especially on datasets with large domain shifts like TextbookQA.

Train	Eval	Tuning	Accuracy	F1
QQP	MRPC	Model	73.1 ±0.9	81.2 ±2.1
		Prompt	**76.3** ±0.1	**84.3** ±0.3
MRPC	QQP	Model	74.9 ±1.3	**70.9** ±1.2
		Prompt	**75.4** ±0.8	69.7 ±0.3

Table 2: Mean and stddev of zero-shot domain transfer between two paraphrase detection tasks.

Prompt Ensembling

- By training N prompts on the same tasks, we create N separate “models” for a task, while still sharing the core language modeling parameters throughout



Dataset	Metric	Average	Best	Ensemble
BoolQ	acc.	91.1	91.3	**91.7**
CB	acc./F1	99.3 / 99.0	100.00 / 100.00	**100.0 / 100.0**
COPA	acc.	98.8	100.0	**100.0**
MultiRC	EM/F1$_a$	65.7 / 88.7	66.3 / 89.0	**67.1 / 89.4**
ReCoRD	EM/F1	92.7 / 93.4	92.9 / 93.5	**93.2 / 93.9**
RTE	acc.	92.6	**93.5**	**93.5**
WiC	acc.	76.2	76.6	**77.4**
WSC	acc.	95.8	**96.2**	**96.2**
SuperGLUE (dev)		90.5	91.0	**91.3**

Table 3: Performance of a five-prompt ensemble built from a single frozen T5-XXL model exceeds both the average and the best among the five prompts.

Conclusion

- On SuperGLUE benchmark
  - the task performance rivals that of traditional model tuning, with the gap vanishing as model size increases
- On zero-shot domain transfer
  - prompt tuning leads to improved generalization
- Efficiency
  - efficient storage and serving costs -> multi-task serving, prompt ensembling