2022 Large Language Model Bootcamp

Johns Hopkins Whiting School of Engineering

Practical Tips on BERT Applications

Xuan Zhang

June 30, 2022

Improve BERT by

I. Optimizing BERT pre-training

II. Optimizing BERT fine-tuning

III. Hyperparameter Search

A Primer in BERTology: What We Know About How BERT Works

Anna Rogers
Center for Social Data Science
University of Copenhagen
arogers@sodas.ku.dk

Olga Kovaleva
Dept. of Computer Science
University of Massachusetts Lowell
okovalev@cs.uml.edu

Anna Rumshisky
Dept. of Computer Science
University of Massachusetts Lowell
arum@cs.uml.edu

Optimizing BERT Pre-Training

Alternative training objectives — MLM alternatives

- **How to mask**

  Static masking vs. dynamic masking (Liu et al., 2019b)

  Replace MASK token w/ [UNK] (Clinchant et al., 2019)

- **What to mask**

  Full words vs. word-pieces (Devlin et al., 2019; Cui et al., 2019)

  Spans vs. single tokens (Joshi et al., 2020)

  Phrases & named entities (Sun et al., 2019b)

Optimizing BERT Pre-Training

Alternative training objectives — MLM alternatives (continued)

- **Where to mask**

  Arbitrary text streams vs. Sentence pairs (Lample and Conneau, 2019)

- **Alternatives to masking**

  Deletion, infilling, sentence permutation, document rotation (Lewis et al., 2019)

  Predict whether a token is capitalized and whether it occurs in other segments (Sun et al., 2019c)

  Train on different permutations of word order, maximizing the prob of original order (Yang et al., 2019)

Optimizing BERT Pre-Training

Alternative training objectives — NSP alternatives

Removing NSL does not hurt or slightly improves performance.

Predict both the next and previous sentences (Wang et al., 2019a; Cheng et al., 2019)

Sentence reordering and sentence distance prediction (ERNIE 2.0)

Optimizing BERT Pre-Training

Incorporate External Knowledge

- **Incorporate explicit linguistic information**

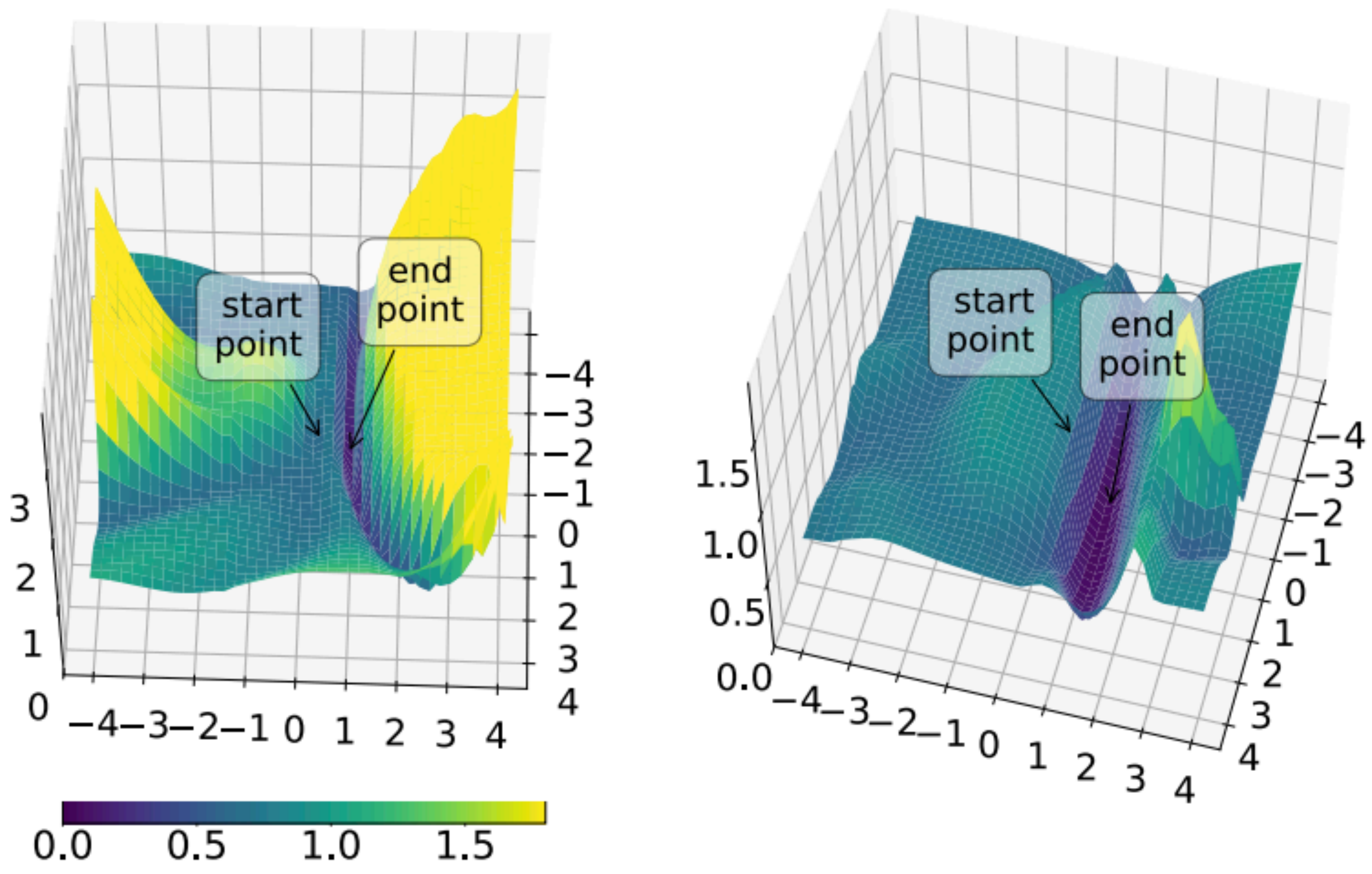
- **Explicitly supply structured knowledge**

  Include entity embeddings as input for training BERT (Peters et al., 2019a; Zhang et al., 2019)

  Mask named entities rather than random words (Sun et al., 2019b, c)

Optimizing BERT Pre-Training

Why does it help?



Pre-trained weights help BERT find wider optima in fine-tuning on MRPC (right) than training from scratch (left).

Optimizing BERT Fine-Tuning

Taking more layers into account

Kovaleva et al., 2019: During fine-tuning, the most changes occur in the last two layers, and those changes cause self-attention to focus on [SEP] rather than linguistically interpretable patterns.

Learn a complementary representation of the information in deep & output layers (Yang and Zhao, 2019)

Use a weighted combination of all layers instead of the final one (Su and Cheng, 2019; Kondratyuk and Straka, 2019)

Optimizing BERT Fine-Tuning

Two-stage fine-tuning

Introduce an intermediate supervised training stage between pre-training and fine-tuning.

Optimizing BERT Fine-Tuning

Regularization

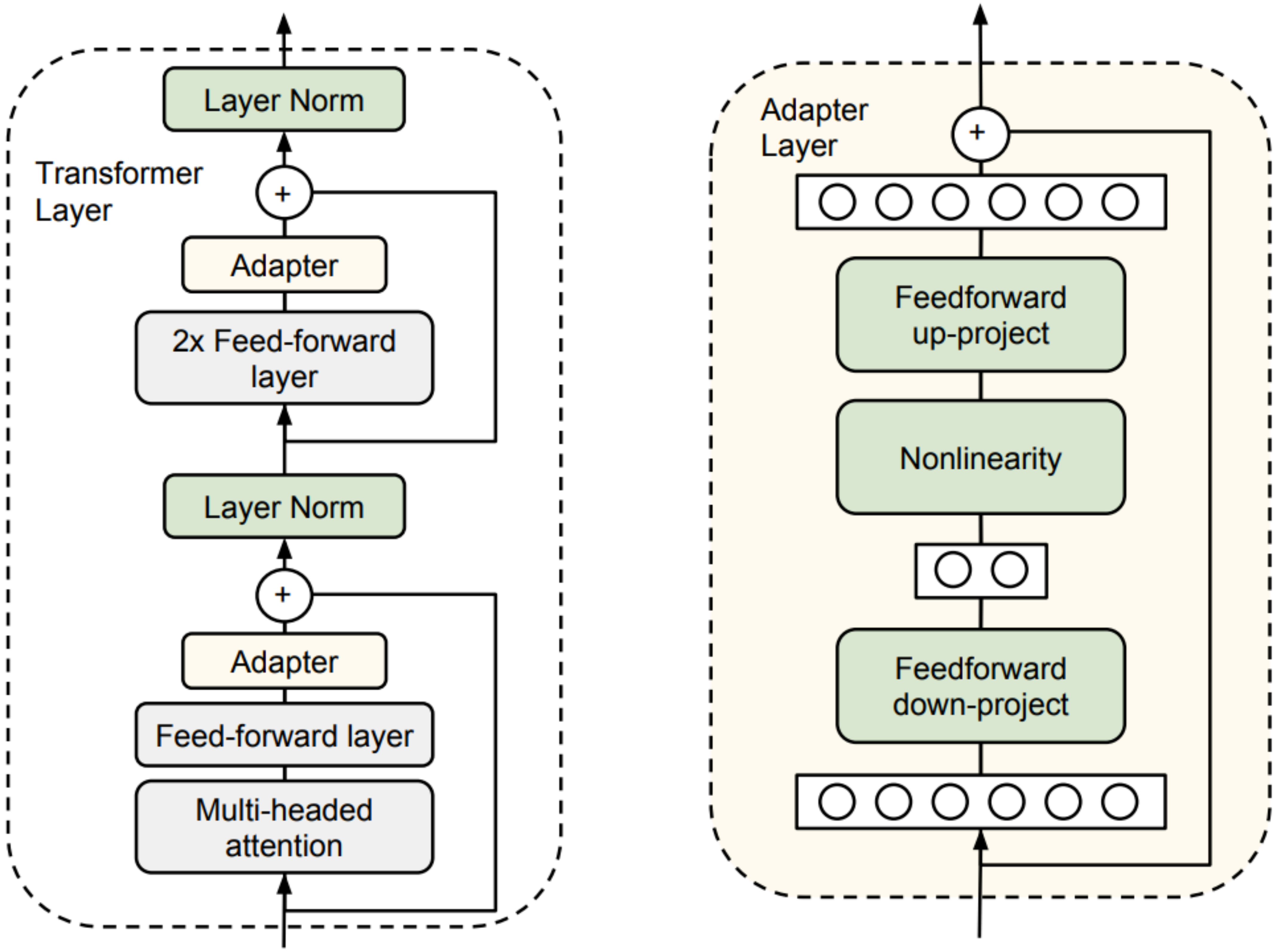
Jiang et al., 2019:

Encourage output of the model not to change much, when injecting a small perturbation to the input.

Update the model only within a small neighborhood of the previous iterate.

Optimizing BERT Fine-Tuning

Adapter



Hyperparameter Search

Architecture Choices

- **Larger hidden representation size** is consistently better
- **#attention heads** is not as significant as **#layers**
- Information flow through **layer**s: task-invariant at initial layers -> task-specific at higher layers; a deeper model has more capacity to encode task-invariant info
- Many **self-attention heads** learn the same patterns
- Benefits can be obtained with more attention sublayers at the bottom, and more feedforward sublayers at the top (Press et al., 2020)

Hyperparameter Search

Training Regime

- Large batch training (8k, 32k)
- Normalization of the trained [CLS] (Zhou et al., 2019)
- Recursive training: shallow layers are trained first and then copied to deeper layers

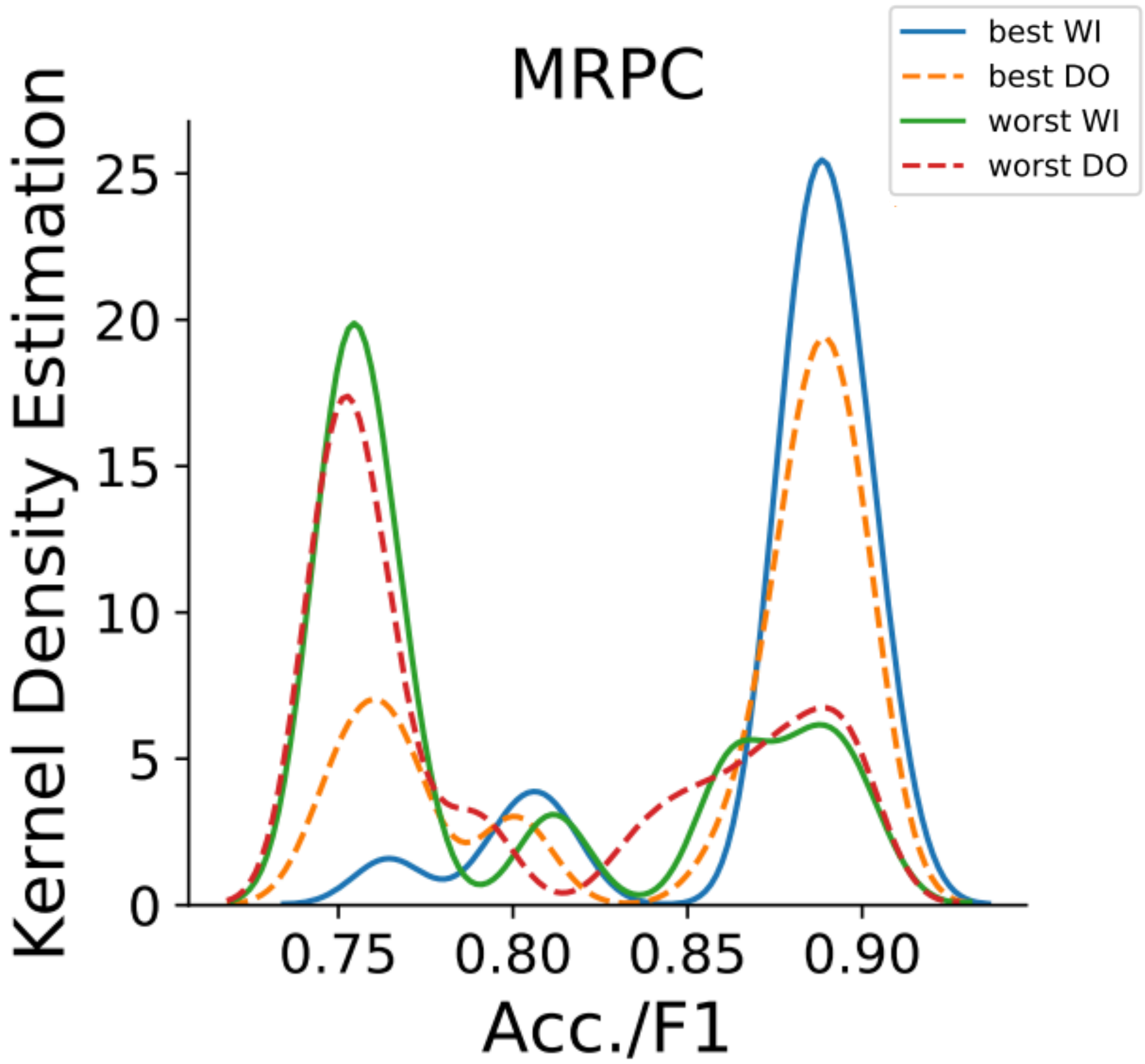
  -> 25% faster (Gong et al., 2019)

Hyperparameter Search

Random Seeds (weight initialization, data order)

	MRPC	RTE	CoLA	SST
BERT (Phang et al., 2018)	90.7	70.0	62.1	92.5
BERT (Liu et al., 2019)	88.0	70.4	60.6	93.2
BERT (ours)	**91.4**	**77.3**	**67.6**	**95.1**
STILTs (Phang et al., 2018)	90.9	83.4	62.1	93.2
XLNet (Yang et al., 2019)	89.2	83.8	63.6	95.6
RoBERTa (Liu et al., 2019)	90.9	86.6	68.0	96.4
ALBERT (Lan et al., 2019)	90.9	89.2	71.4	96.9

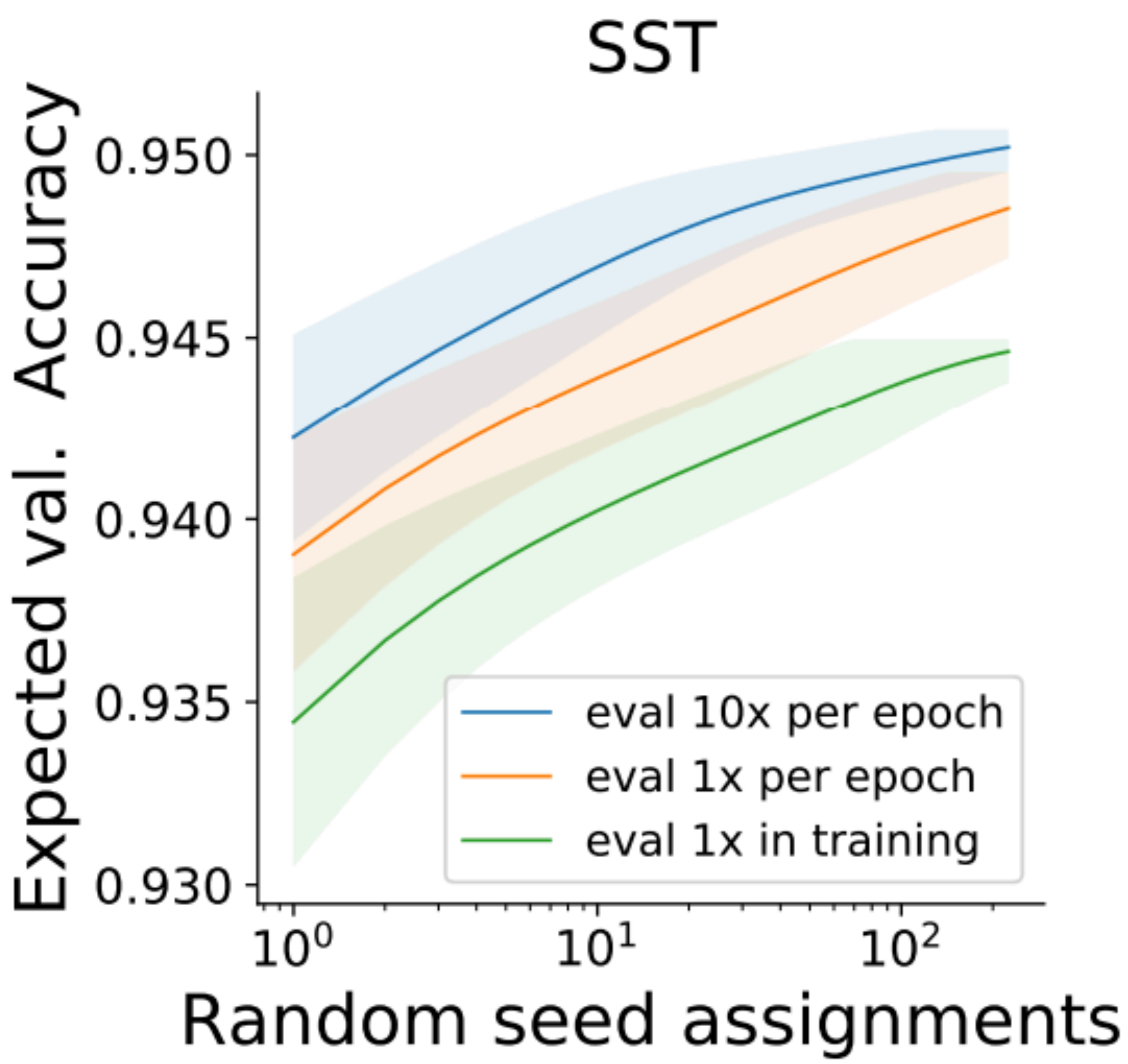
** Ours: Tuning only the random seeds*



Hyperparameter Search

Random Seeds (weight initialization, data order)

Frequently evaluating the model on validation data leads to higher expected validation values.



Hyperparameter Search

Early Stopping (save computations in hyperparameter search)

Start many, stop early, continue some

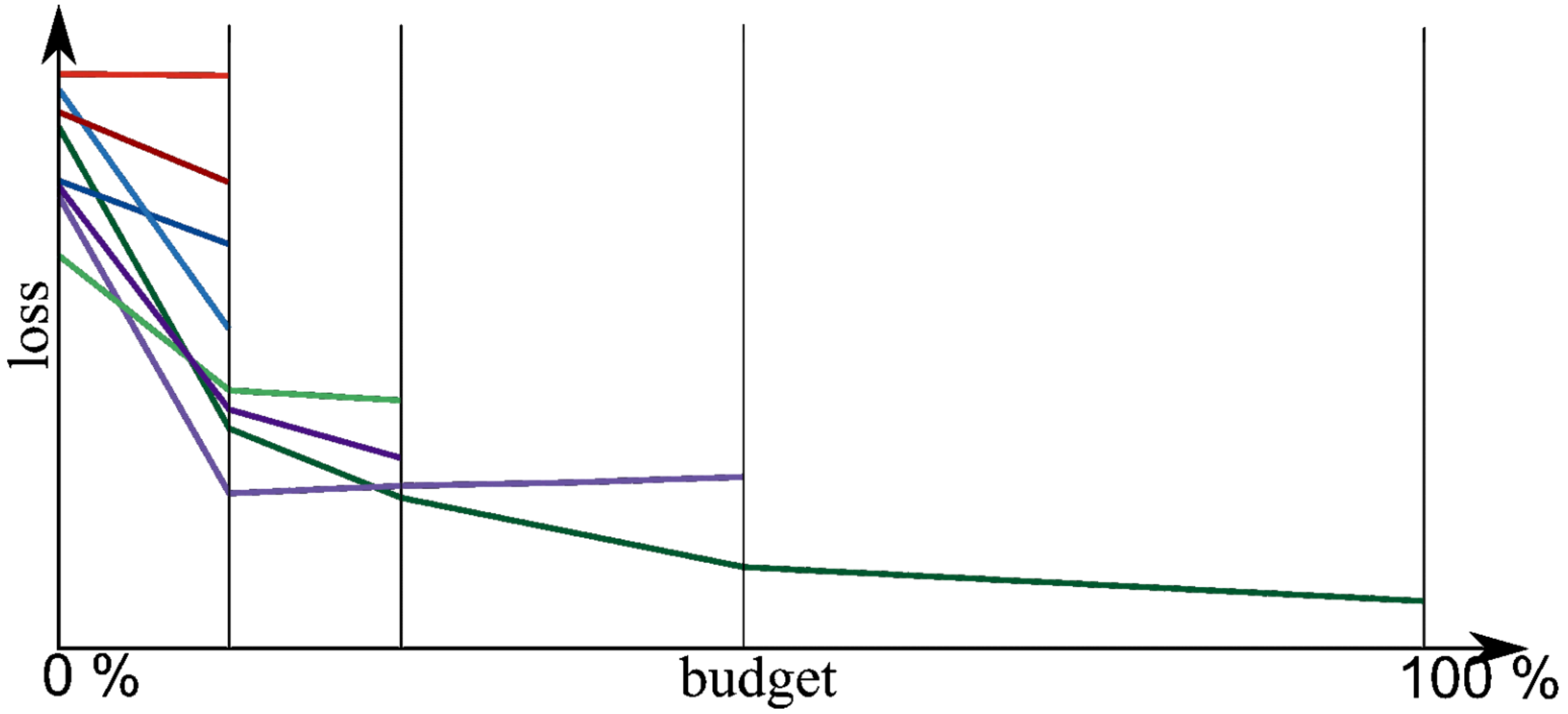


Figure from automl.org

Takeaways

1. Though there exist a large number of BERT modifications, gains are often marginal, significant testing are rare.

2. Performance improvements of new models may be within variation induced by environment factors & random seeds.

3. It is nontrivial to tune random seeds.