TECHNISCHE
UNIVERSITÄT
WIEN

Reproducing TinyBERT

Anni Chen, Anton Martinovic, Paul von Hirschhausen, Leonardo Hrgic
Group 13

TECHNISCHE UNIVERSITÄT WIEN

Agenda

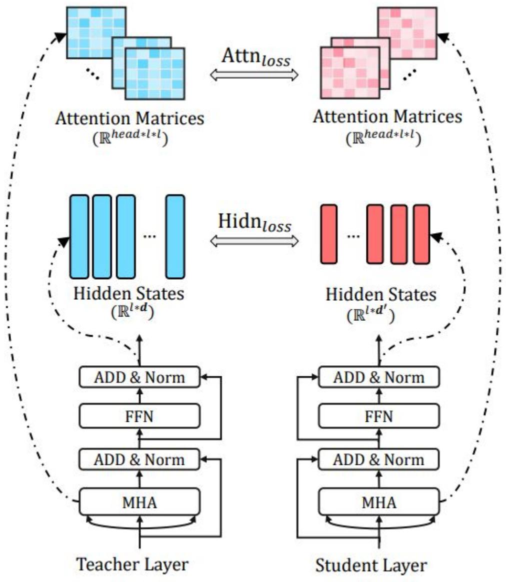
- Introduction and Motivation
- Transformer Distillation Objectives
- Two-Stage Distillation Pipeline
- Data Augmentation
- Layer Mapping
- Extensions
- Reproduction Results

Introduction & Motivation

- BERT does well in many NLP tasks
- Problem: BERT-Base has 109M parameters
  - Too slow/large for mobile devices and real-time inference.
- Knowledge Distillation.
- Goal: Transfer attention, hidden states and prediction layers to a smaller student model

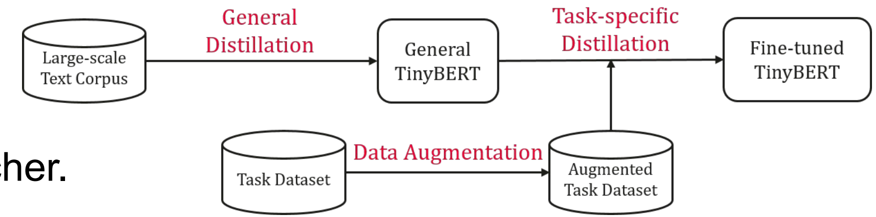
Transformer Distillation Objective

- Attention-based: Student mimics teacher's attention matrices
  - Captures syntax & coreference
  - Loss: Mean Squared Error (MSE) between teacher and student attention maps.
- Hidden State-based: Student mimics teacher's hidden outputs
  - Captures semantics
  - Learnable weight matrix W_h performs a linear transformation to align dimensions
- Embedding-layer Distillation: similar to hidden state based distillation
- Prediction-layer distillation fits the teacher's final output behavior



Two-Stage Distillation Pipeline

- Stage 1: General Distillation (Foundation):
  - Uses the original $\text{BERT}_{\text{Base}}$ (not fine-tuned) as a teacher.
  - Distills on a large, general corpus (like Wikipedia).
  - Goal: Transfer general linguistic knowledge to the student.
- Stage 2: Task-Specific Distillation (Fine-Tuning):
  - Uses a Fine-tuned BERT as a teacher.
  - Uses Augmented Data
  - Goal: Transfer specialized "behavior" (e.g., sentiment or grammar) to the student.



TECHNISCHE UNIVERSITÄT WIEN

Data Augmentation

- Problem: Small datasets (like CoLA or RTE) are too small to learn complex behaviors.
- Method:
  - World-level replacement: Identify keywords in the original sentence.
  - Contextual Prediction: Use BERT to suggest synonyms that fit the sentence structure.
  - Semantic Similarity: Use GloVe embeddings to find word-level synonyms.
- Result: Larger dataset that provides more examples.

Layer Mapping

- Problem: Teacher has more layers than the student so a mapping function is required.
- Different Mapping Strategies:
  - Uniform-Strategy: $g(m) = 3 \times m$ (student learns from every 3rd Mapping)
  - Top-Strategy: $g(m) = m + (N-M)$ (focuses only on the final layers)
  - Bottom-Strategy: $g(m) = m$ (focuses only on initial layers)

TECHNISCHE UNIVERSITÄT WIEN

$$\mathcal{L}_{\text{model}} = \sum_{x \in \mathcal{X}} \sum_{m=0}^{M+1} \lambda_m \mathcal{L}_{\text{layer}}(f_m^S(x), f_{g(m)}^T(x)), \quad (6)$$

TinyBERT Learning

- Teacher: BERT-Base (12 layers, 768 hidden size).
- Student: TinyBERT (4 layers, 312 hidden size).
- Loss Functions:
  - L_{Emb}: aligns the student's embeddings with the teacher's token and positional encodings.
  - L_{attn}: Learns linguistic structure (syntax).
  - L_{hidn}: Learns semantic representations.
  - L_{pred}: Learns final classification logic (Logits).

$$\mathcal{L}_{\text{embd}} = \text{MSE}(\boldsymbol{E}^S \boldsymbol{W}_e, \boldsymbol{E}^T),$$

$$\mathcal{L}_{\text{attn}} = \frac{1}{h} \sum_{i=1}^{h} \text{MSE}(\boldsymbol{A}_i^S, \boldsymbol{A}_i^T),$$

$$\mathcal{L}_{\text{hidn}} = \text{MSE}(\boldsymbol{H}^S \boldsymbol{W}_h, \boldsymbol{H}^T),$$

$$\mathcal{L}_{\text{pred}} = \text{CE}(\boldsymbol{z}^T / t, \boldsymbol{z}^S / t),$$

TECHNISCHE UNIVERSITÄT WIEN

Results Tinybert

- Size: 109M Params → 14,5M Params (13,3%)
- Speedup: 9,4x
- Avg. Glue Score 79,5 → 77 (96,8%)

System	#Params	#FLOPs	MNLI	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	AVG
BERT	109M	22.5B	83,9	71,1	90,9	93,4	52,8	85,2	87,5	67,0	79,5
TinyBert	14,5M	1.2B	82.5	71.3	87.7	92.6	44.1	80.4	86.4	66.6	77.0

Reproduction – Test Environment

- Tasks: Subset of GLUE (SST-2, MRPC, RTE, CoLA).
- Teacher: $\text{BERT}_{\text{BASE}}$ (Uncased).
- Student: TinyBERT_4 (4-layers, 312-hidden).
- Goal: Validate the author's claim that Uniform mapping is the superior distillation strategy.

Fine Tuning - Tasks

- **CoLA** (Corpus of Linguistic Acceptability): given a sentence, predict whether it is acceptable/grammatical (acceptable) or unacceptable/ungrammatical (unacceptable).
- **RTE** (Recognizing Textual Entailment): given a premise sentence and a hypothesis sentence, predict whether the premise entails the hypothesis (entailment) or does not entail it.
- **SST-2** (Stanford Sentiment Treebank, binary): given a sentence (from movie reviews), predict its sentiment: positive or negative.
- **MRPC** (Microsoft Research Paraphrase Corpus): given two sentences, predict whether they are paraphrases / semantically equivalent (equivalent) or not (not_equivalent).

Extension 1: Flexible Layer Mapping

- Mapping Strategies:
  - Uniform: Standard sampling; covers full teacher depth
  - Top: Learns from teacher’s final layers
  - Bottom: Learns from teacher’s initial layers
- Allows direct performance benchmarking across different mapping approaches

```
if args.layer_map == "uniform":
    assert teacher_layer_num % student_layer_num == 0
    layers_per_block = teacher_layer_num // student_layer_num
    att_idxs = [i * layers_per_block + (layers_per_block - 1) for i in range(student_layer_num)]
    rep_idxs = [i * layers_per_block for i in range(student_layer_num + 1)]

elif args.layer_map == "top":
    # last M layers
    att_idxs = list(range(teacher_layer_num - student_layer_num, teacher_layer_num))
    # keep embedding (0), then take the last M layer outputs
    # teacher layer outputs correspond to indices 1..teacher_layer_num in teacher_reps
    rep_idxs = [0] + list(range((teacher_layer_num - student_layer_num) + 1, teacher_layer_num + 1))
elif args.layer_map == "bottom":
    # first M layers
    att_idxs = list(range(student_layer_num))
    # embedding + first M layer outputs
    rep_idxs = list(range(student_layer_num + 1))
```

Extension 2: Quantization

- Reduces numeric precision from FP32 → INT8
- Smaller model size and faster CPU inference
- Static Quantization:
  - No retraining required - uses calibration data to estimate activation ranges (no retraining).
  - Done for the task specific BERT, not for the distilled models
- Two Quantization Variants:
  - Weights: INT8; Activations: INT8 (QDQ) / UINT8 (QOperator)

- Quant 1 – QDQ
  - Insert Quantize/DeQuantize nodes
  - Higher compatibility, often better accuracy preservation
- Quant 2 – QOperator
  - Uses native INT8 operators
  - Sometimes faster execution, more aggressive optimization

Extension 3: Weight Pruning

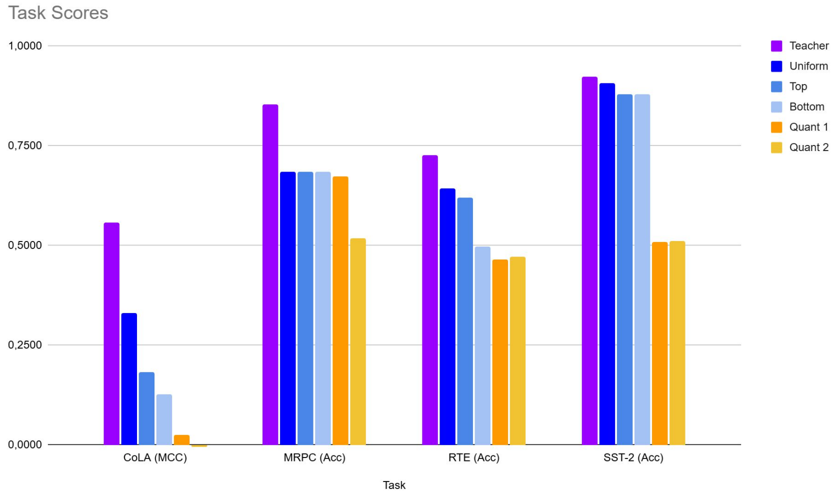
- Iterative unstructured pruning (global L1 == prune by absolute weight magnitude across all linear layers)
  - Applies to all nn.Linear.weight tensors (att + FFN, classifier)
- Gradually add sparsity over first 5–6 epochs
- Pruning amount 20% - applies in increments
- Preserve sparsity during training:
  - Gradient masking → zero gradients at pruned positions
  - Weight re-masking after optimizer.step() (forces pruned weights back to 0)
  - Finalize pruning by baking masks into weight tensors before saving

```python
if args.prune and pruning_masks is not None:
    for name, param in student_model.named_parameters():
        if name in pruning_masks:
            if param.grad is not None:
                param.grad.mul_(pruning_masks[name].to(device))
```

```python
for module in model.modules():
    if isinstance(module, torch.nn.Linear):
        # target the 'weight' of every Linear layer in the entire model
        parameters_to_prune.append((module, 'weight'))

if parameters_to_prune:
    prune.global_unstructured(
        parameters_to_prune,
        pruning_method=prune.L1Unstructured,
        amount=amount,
    )
```

Reproduction Results (no DA)

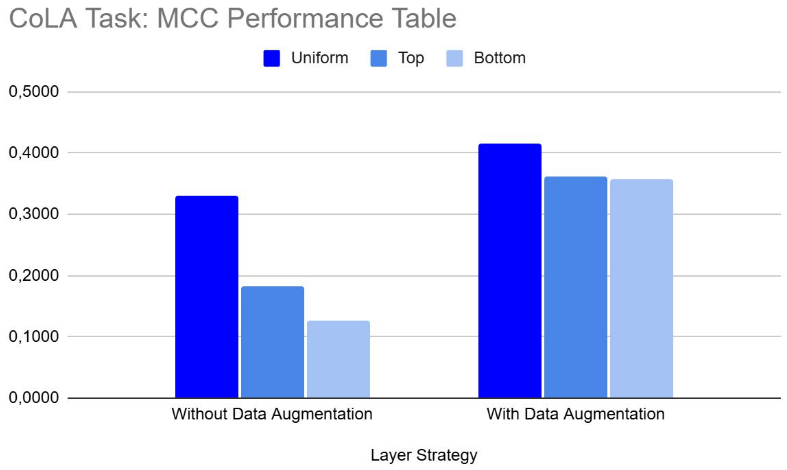


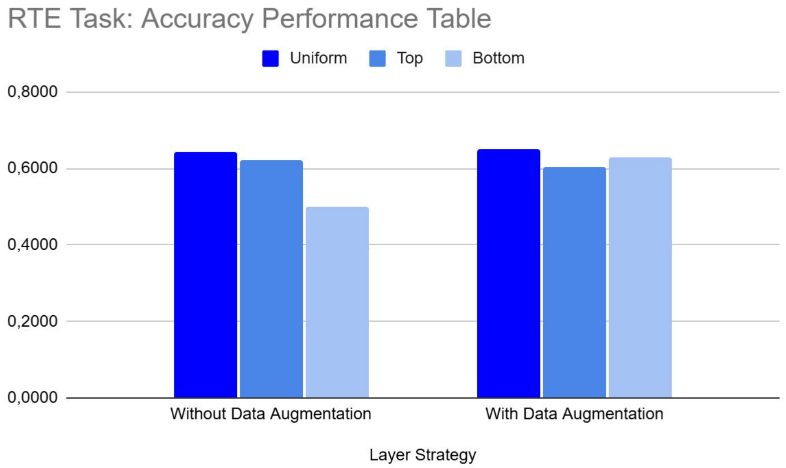
Task	Teacher Score	Student Score	Paper Score	Retention Rate (%)	Epochs inter/pred
SST-2	0,924	0,906	0,926	**98,02 %**	**20/3**
RTE	0,725	0,643	0,666	**89,3 %**	**20/3**
MRPC	0,852	0,683	0,864	**80,17 %**	**20/3**
CoLA	0,557	0,331	0,441	**59,4 %**	**20/3**

Reproduction Results

Modell	Size (MB)	Size (%)	Latency (ms)	Speedup
BERT-Base	417,85	100%	2.814,70	1,0x
TinyBERT	**54,82**	**13,11%**	**209,86**	**13,4x**
Quantized 1 (QDQ)	105,23	64,63%	3.436,46	0,8x
Quantized 2 (QOperator)	105,03	64,63%	1.436,28	1,9x

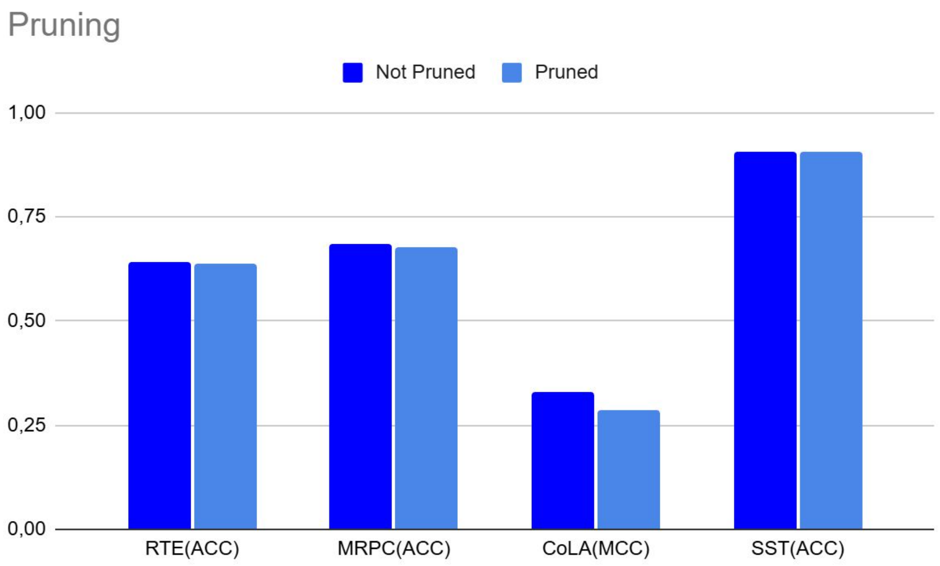
Reproduction Results - Data Augmentation (x30)





Pruning Comparison (no DA, uniform mapping)

- Sparsity = 6%
- Compressed File Size
  52MB → 50MB



Conclusion & Challenges

- **Reproduced the TinyBERT** task-specific distillation pipeline on a GLUE subset (SST-2, MRPC, RTE, CoLA).
- **Verified strong compression benefits** (BERT-Base → TinyBERT), with clear speedups and moderate, task-dependent performance drops.
- **Layer-mapping trends were reproducible.**
- **Quantization** can be a practical **alternative** for some tasks (e.g., MRPC), requiring less development effort than a full knowledge-distillation pipeline.
- **Weight pruning did not show a substantial impact** in our setup.

- Data augmentation is **computationally expensive** to run at scale.
- Distillation on large augmented datasets significantly increases training time and resource requirements.
- Matching the paper's results exactly is difficult due to **sensitivity to training setup**.

TECHNISCHE
UNIVERSITÄT
WIEN

Thank you for your attention!

Are there any questions?